Implementing Layer-3 Multicast Routing on Cisco IOS XR Software

This module describes how to implement Layer 3 multicast routing on Cisco ASR 9000 Series Routers running Cisco IOS XR Software.

Multicast routing is a bandwidth-conserving technology that reduces traffic by simultaneously delivering a single stream of information to potentially thousands of corporate recipients and homes. Applications that take advantage of multicast routing include video conferencing, corporate communications, distance learning, and distribution of software, stock quotes, and news.

This document assumes that you are familiar with IPv4 and IPv6 multicast routing configuration tasks and concepts for Cisco IOS XR Software .

Multicast routing allows a host to send packets to a subset of all hosts as a group transmission rather than to a single host, as in unicast transmission, or to all hosts, as in broadcast transmission. The subset of hosts is known as **group members** and are identified by a single multicast group address that falls under the IP Class D address range from 224.0.0.0 through 239.255.255.255.

For detailed conceptual information about multicast routing and complete descriptions of the multicast routing commands listed in this module, you can refer to the Related Documents, on page 275.

Feature History for Configuring Multicast Routing on the Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.9.0	Support was added for these features: • Flow-based multicast only fast reroute (MoFRR). • IGMP VRF override.
Release 3.9.1	Support was added for the Multicast VPN feature. (For IPv4 address family)
Release 4.0.0	Support was added for IPv4 Multicast routing, Multicast VPN basic and InterAS option A on Cisco ASR 9000 Series SPA Interface Processor-700 linecard and MVPN Hub and Spoke Topology.
Release 4.0.1	Support was added for IPv6 Multicast routing.

Release	Modification
Release 4.1.0	Support was added for Label Switched Multicast using Point-to-Multipoint Traffic Engineering in global context only (not in VRF).
Release 4.2.1	Support was added for these features: • Label Switched Multicast using MLDP (Multicast Label Distribution Protocol). • Multicast VPN for IPv6 address family. • Support for Satellite nV. • InterAS Support on Multicast VPN.
Release 4.3.2	Support was added for these features: • Support for IPv4 traffic on Multicast over unicast GRE was introduced. • Support was added for TI (Topology Independent) MoFRR.
Release 5.2.0	Support was introduced for Bidirectional Global Protocol Independent Multicast.
Release 5.3.2	Support for IPv6 traffic and ECMP on Multicast over unicast GRE was introduced.
Release 6.0.0	Support for MLDP Carrier Supporting Carrier based MVPN was introduced.
Release 6.1.2	Layer 3 Multicast Bundle Subinterface Load Balancing feature was introduced.
Release 6.1.2	Segmented Multicast Stitching with Inter AS and MLDP Carrier Supporting Carrier based MVPN feature support was extended to support Cisco IOS XR 64 bit.
Release 6.1.2	MVPN, MoGRE, MoFRR and Global Table Multicast feature support was extended to support Cisco IOS XR 64 bit.

- Prerequisites for Implementing Multicast Routing, on page 2
- Information About Implementing Multicast Routing, on page 3
- Layer 3 Multicast Bundle Subinterface Load Balancing, on page 122
- How to Implement Multicast Routing, on page 125
- Multicast only fast reroute (MoFRR), on page 193
- Enabling multicast on PW-HE interfaces, on page 200
- Configuring Route Policy for Static RPF, on page 202
- Point-to-Multipoint Traffic Engineering Label-Switched Multicast, on page 204
- Configuring IGMP VRF Override, on page 207
- MVPN GRE over PWHE with CSI, on page 211
- Multicast Traceroute v2, on page 212
- Configuration Examples for Implementing Multicast Routing on Software, on page 214
- Additional References, on page 275

Prerequisites for Implementing Multicast Routing

- You must install and activate the multicast pie.

- For detailed information about optional PIE installation, see *Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
- For MLDP, an MPLS PIE has to be installed.
- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be familiar with IPv4 and IPv6 multicast routing configuration tasks and concepts.
- Unicast routing must be operational.
- To enable multicast VPN, you must configure a VPN routing and forwarding (VRF) instance.

Information About Implementing Multicast Routing

Key Protocols and Features Supported in the Cisco IOS XR Software Multicast Routing Implementation

Table 1: Supported Features for IPv4 and IPv6

Feature	IPv4 Support	IPv6 Support
Dynamic host registration	Yes (IGMP v1/2/3)	Yes
Explicit tracking of hosts, groups, and channels	Yes (IGMP v3)	Yes
PIM-SM[1]	Yes	Yes
PIM-SSM	Yes	Yes
PIM-SSM Mapping	Yes	Yes
Auto-RP	Yes	No
Multicast VPN	Yes	Yes
InterAS Option A	Yes	Yes
BSR	Yes	Yes
BGP	Yes	Yes
MSDP	Yes	No
Multicast NSF	Yes	Yes
OOR handling	Yes	Yes
Global Protocol Independent Multicast Bidirectional support	Yes	Yes

1

Multicast Routing Functional Overview

Traditional IP communication allows a host to send packets to a single host (*unicast transmission*) or to all hosts (*broadcast transmission*). Multicast provides a third scheme, allowing a host to send a single data stream to a subset of all hosts (*group transmission*) at about the same time. IP hosts are known as group members.

Packets delivered to group members are identified by a single multicast group address. Multicast packets are delivered to a group using best-effort reliability, just like IP unicast packets.

The multicast environment consists of senders and receivers. Any host, regardless of whether it is a member of a group, can send to a group. However, only the members of a group receive the message.

A multicast address is chosen for the receivers in a multicast group. Senders use that group address as the destination address of a datagram to reach all members of the group.

Membership in a multicast group is dynamic; hosts can join and leave at any time. There is no restriction on the location or number of members in a multicast group. A host can be a member of more than one multicast group at a time.

How active a multicast group is and what members it has can vary from group to group and from time to time. A multicast group can be active for a long time, or it may be very short-lived. Membership in a group can change constantly. A group that has members may have no activity.

Routers use the Internet Group Management Protocol (IGMP) (IPv4) and Multicast Listener Discovery (MLD) (IPv6) to learn whether members of a group are present on their directly attached subnets. Hosts join multicast groups by sending IGMP or MLD report messages.

Many multimedia applications involve multiple participants. Multicast is naturally suitable for this communication paradigm.

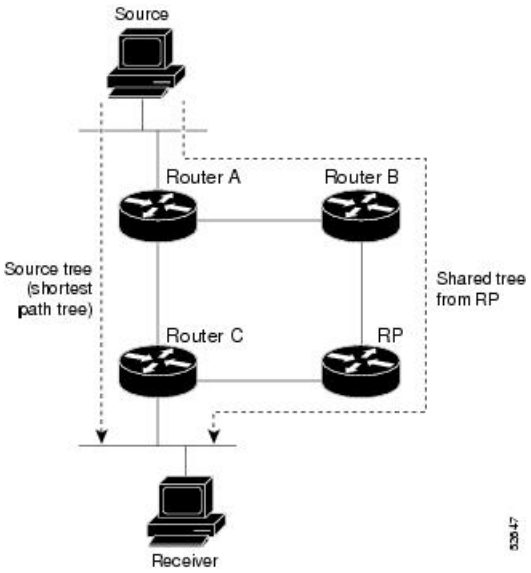
Multicast Routing Implementation

Cisco IOS XR Software supports the following protocols to implement multicast routing:

- IGMP is used between hosts on a LAN and the routers on that LAN to track the multicast groups of which hosts are members.
- Protocol Independent Multicast in sparse mode (PIM-SM) is used between routers so that they can track which multicast packets to forward to each other and to their directly connected LANs.
- Protocol Independent Multicast in Source-Specific Multicast (PIM-SSM) is similar to PIM-SM with the additional ability to report interest in receiving packets from specific source addresses (or from all but the specific source addresses), to an IP multicast address.
- PIM-SSM is made possible by IGMPv3 and MLDv2. Hosts can now indicate interest in specific sources using IGMPv3 and MLDv2. SSM does not require a rendezvous point (RP) to operate.
- PIM Bidirectional is a variant of the Protocol Independent Multicast suit of routing protocols for IP multicast. PIM-BIDIR is designed to be used for many-to-many applications within individual PIM domains.

This image shows IGMP and PIM-SM operating in a multicast environment.

Figure 1: Multicast Routing Protocols



PIM-SM, PIM-SSM, and PIM-BIDIR

Protocl Independent Multicast (PIM) is a multicast routing protocol used to create multicast distribution trees, which are used to forward multicast data packets. PIM is an efficient IP routing protocol that is "independent" of a routing table, unlike other multicast protocols such as Multicast Open Shortest Path First (MOSPF) or Distance Vector Multicast Routing Protocol (DVMRP).

Cisco IOS XR Software supports Protocol Independent Multicast in sparse mode (PIM-SM), Protocol Independent Multicast in Source-Specific Multicast (PIM-SSM), and Protocol Independent Multicast in Bi-directional mode (BIDIR) permitting these modes to operate on your router at the same time.

PIM-SM and PIM-SSM supports one-to-many applications by greatly simplifying the protocol mechanics for deployment ease. Bidir PIM helps deploy emerging communication and financial applications that rely on a many-to-many applications model. BIDIR PIM enables these applications by allowing them to easily scale to a very large number of groups and sources by eliminating the maintenance of source state.

PIM-SM Operations

PIM in sparse mode operation is used in a multicast network when relatively few routers are involved in each multicast and these routers do not forward multicast packets for a group, unless there is an explicit request for the traffic.

For more information about PIM-SM, see the PIM-Sparse Mode, on page 9.

PIM-SSM Operations

PIM in Source-Specific Multicast operation uses information found on source addresses for a multicast group provided by receivers and performs source filtering on traffic.

- By default, PIM-SSM operates in the 232.0.0.0/8 multicast group range for IPv4 and ff3x::/32 (where x is any valid scope) in IPv6. To configure these values, use the **ssm range** command.

- If SSM is deployed in a network already configured for PIM-SM, only the last-hop routers must be upgraded with Cisco IOS XR Software that supports the SSM feature.
- No MSDP SA messages within the SSM range are accepted, generated, or forwarded.

PIM-Bidirectional Operations

PIM Bidirectional (BIDIR) has one shared tree from sources to RP and from RP to receivers. This is unlike the PIM-SM, which is unidirectional by nature with multiple source trees - one per (S,G) or a shared tree from receiver to RP and multiple SG trees from RP to sources.

Benefits of PIM BIDIR are as follows:

- As many sources for the same group use one and only state (*, G), only minimal states are required in each router.
- No data triggered events.
- Rendezvous Point (RP) router not required. The RP address only needs to be a routable address and need not exist on a physical device.

Restrictions for PIM-SM and PIM-SSM, and PIM BIDIR

Interoperability with SSM

PIM-SM operations within the SSM range of addresses change to PIM-SSM. In this mode, only PIM (S,G) join and prune messages are generated by the router, and no (S,G) RP shared tree or (*,G) shared tree messages are generated.

IGMP Version

To report multicast memberships to neighboring multicast routers, hosts use IGMP, and all routers on the subnet must be configured with the same version of IGMP.

A router running Cisco IOS XR Software does not automatically detect Version 1 systems. You must use the **version** command in router IGMP configuration submode to configure the IGMP version.

PIM-Bidir Restrictions

PIM-Bidir is not supported on MVPN.

Internet Group Management Protocol

Cisco IOS XR Software provides support for Internet Group Management Protocol (IGMP) over IPv4.

IGMP provides a means for hosts to indicate which multicast traffic they are interested in and for routers to control and limit the flow of multicast traffic throughout the network. Routers build state by means of IGMP and MLD messages; that is, router queries and host reports.

A set of queries and hosts that receive multicast data streams from the same source is called a *multicast group*. Hosts use IGMP and MLD messages to join and leave multicast groups.

Note IGMP messages use group addresses, which are Class D IP addresses. The high-order four bits of a Class D address are 1110. Host group addresses can be in the range 224.0.0.0 to 239.255.255.255. The address 224.0.0.0 is guaranteed not to be assigned to any group. The address 224.0.0.1 is assigned to all systems on a subnet. The address 224.0.0.2 is assigned to all routers on a subnet.

IGMP Versions

The following points describe IGMP versions 1, 2, and 3:

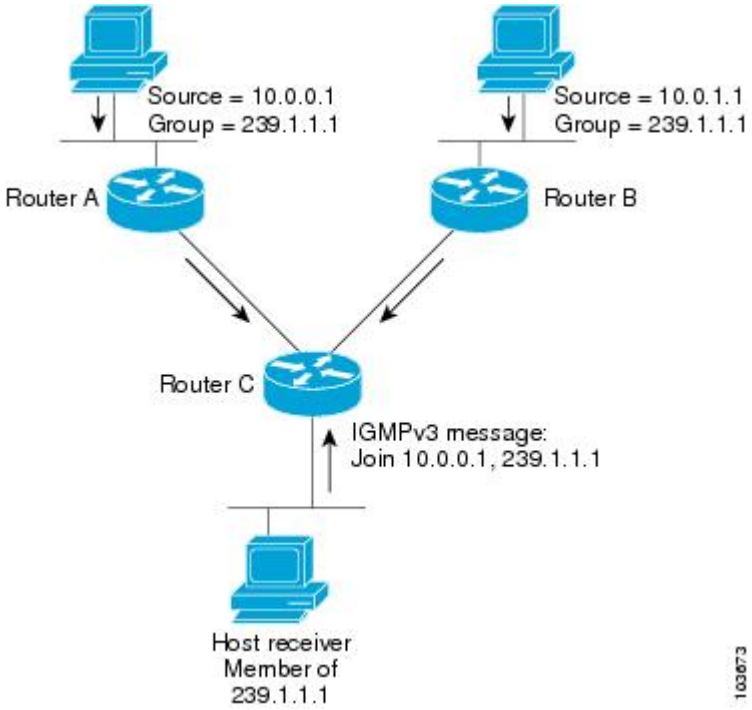
- IGMP Version 1 provides for the basic query-response mechanism that allows the multicast router to determine which multicast groups are active and for other processes that enable hosts to join and leave a multicast group.
- IGMP Version 2 extends IGMP allowing such features as the IGMP query timeout and the maximum query-response time. See RFC 2236.
- IGMP Version 3 permits joins and leaves for certain source and group pairs instead of requesting traffic from all sources in the multicast group.

IGMP Routing Example

Figure 2: IGMPv3 Signaling, on page 8 illustrates two sources, 10.0.0.1 and 10.0.1.1, that are multicasting to group 239.1.1.1. The receiver wants to receive traffic addressed to group 239.1.1.1 from source 10.0.0.1 but not from source 10.0.1.1. The host must send an IGMPv3 message containing a list of sources and groups (S, G) that it wants to join and a list of sources and groups (S, G) that it wants to leave. Router C can now use this information to prune traffic from Source 10.0.1.1 so that only Source 10.0.0.1 traffic is being delivered to

Router C.

Figure 2: IGMPv3 Signaling



Note When configuring IGMP, ensure that all systems on the subnet support the same IGMP version. The router does not automatically detect Version 1 systems. Configure the router for Version 2 if your hosts do not support Version 3.

Configuring IGMP Per Interface States Limit

The IGMP Per Interface States Limit sets a limit on creating OLEs for the IGMP interface. When the set limit is reached, the group is not accounted against this interface but the group can exist in IGMP context for some other interface.

The following configuration sets a limit on the number of group memberships created on an interface as a result of receiving IGMP or MLD membership reports.

```
router igmp | mld [vrf <vrfname>]
            interface <ifname>
                        (no) maximum groups-per-interface <max> [threshold <threshold>]
[<acl>]
            !
!
```

where,

`<ifname>` is the interface name

`<max>` is the maximum limit on the groups

`<threshold>` is the threshold number of groups at which point a syslog warning message will be issued

<acl> provides an option for selective accounting. If provided, only groups or (S,G)s that are permitted by the ACL is accounted against the limit. Groups or (S, G)s that are denied by the ACL are not accounted against the limit. If not provided, all the groups are accounted against the limit.

The following messages are displayed when the threshold limit is reached for IGMP:

```
igmp[1160]: %ROUTING-IPV4_IGMP-4-OOR_THRESHOLD_REACHED : Threshold for Maximum number of
group per interface has been reached 3: Groups joining will soon be throttled.
Config a higher max or take steps to reduce states

igmp[1160]: %ROUTING-IPV4_IGMP-4-OOR_LIMIT_REACHED : Maximum number of group per interface
 has been reached 6: Groups joining is throttled.
Config a higher max or take steps to reduce states
```

Limitations

- If a user has configured a maximum of 20 groups and has reached the maximum number of groups, then no more groups can be created. If the user reduces the maximum number of groups to 10, the 20 joins will remain and a message of reaching the maximum is displayed. No more joins can be added until the number of groups has reached less than 10.
- If a user already has configured a maximum of 30 joins and add a max of 20, the configuration occurs displaying a message that the maximum has been reached. No state change occurs and also no more joins can occur until the threshold number of groups is brought down below the maximum number of groups.

Protocol Independent Multicast

Protocol Independent Multicast (PIM) is a routing protocol designed to send and receive multicast routing updates. Proper operation of multicast depends on knowing the unicast paths towards a source or an RP. PIM relies on unicast routing protocols to derive this reverse-path forwarding (RPF) information. As the name PIM implies, it functions independently of the unicast protocols being used. PIM relies on the Routing Information Base (RIB) for RPF information.

If the multicast subsequent address family identifier (SAFI) is configured for Border Gateway Protocol (BGP), or if multicast intact is configured, a separate multicast unicast RIB is created and populated with the BGP multicast SAFI routes, the intact information, and any IGP information in the unicast RIB. Otherwise, PIM gets information directly from the unicast SAFI RIB. Both multicast unicast and unicast databases are outside of the scope of PIM.

The Cisco IOS XR implementation of PIM is based on RFC 4601 Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification. For more information, see RFC 4601 and the Protocol Independent Multicast (PIM): Motivation and Architecture Internet Engineering Task Force (IETF) Internet draft.

Note Cisco IOS XR Software supports PIM-SM, PIM-SSM, and PIM Version 2 only. PIM Version 1 hello messages that arrive from neighbors are rejected.

PIM-Sparse Mode

Typically, PIM in sparse mode (PIM-SM) operation is used in a multicast network when relatively few routers are involved in each multicast. Routers do not forward multicast packets for a group, unless there is an explicit

request for traffic. Requests are accomplished using PIM join messages, which are sent hop by hop toward the root node of the tree. The root node of a tree in PIM-SM is the rendezvous point (RP) in the case of a shared tree or the first-hop router that is directly connected to the multicast source in the case of a shortest path tree (SPT). The RP keeps track of multicast groups, and the sources that send multicast packets are registered with the RP by the first-hop router of the source.

As a PIM join travels up the tree, routers along the path set up the multicast forwarding state so that the requested multicast traffic is forwarded back down the tree. When multicast traffic is no longer needed, a router sends a PIM prune message up the tree toward the root node to prune (or remove) the unnecessary traffic. As this PIM prune travels hop by hop up the tree, each router updates its forwarding state appropriately. Ultimately, the forwarding state associated with a multicast group or source is removed. Additionally, if prunes are not explicitly sent, the PIM state will timeout and be removed in the absence of any further join messages.

PIM-SM is the best choice for multicast networks that have potential members at the end of WAN links.

PIM-Source Specific Multicast

In many multicast deployments where the source is known, protocol-independent multicast-source-specific multicast (PIM-SSM) mapping is the obvious multicast routing protocol choice to use because of its simplicity. Typical multicast deployments that benefit from PIM-SSM consist of entertainment-type solutions like the ETTH space, or financial deployments that completely rely on static forwarding.

PIM-SSM is derived from PIM-SM. However, whereas PIM-SM allows for the data transmission of all sources sending to a particular group in response to PIM join messages, the SSM feature forwards traffic to receivers only from those sources that the receivers have explicitly joined. Because PIM joins and prunes are sent directly towards the source sending traffic, an RP and shared trees are unnecessary and are disallowed. SSM is used to optimize bandwidth utilization and deny unwanted Internet broadcast traffic. The source is provided by interested receivers through IGMPv3 membership reports.

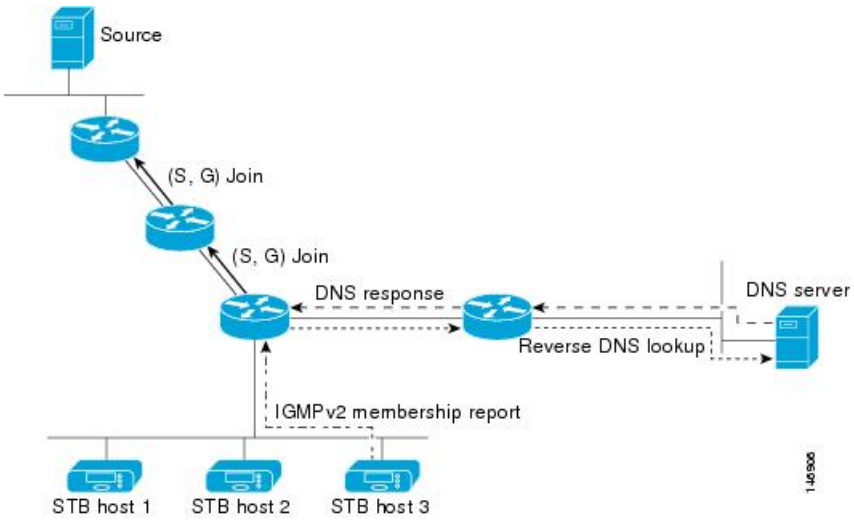
In SSM, delivery of datagrams is based on (S,G) channels. Traffic for one (S,G) channel consists of datagrams with an IP unicast source address S and the multicast group address G as the IP destination address. Systems receive traffic by becoming members of the (S,G) channel. Signaling is not required, but receivers must subscribe or unsubscribe to (S,G) channels to receive or not receive traffic from specific sources. Channel subscription signaling uses IGMP to include mode membership reports, which are supported only in Version 3 of IGMP (IGMPv3).

To run SSM with IGMPv3, SSM must be supported on the multicast router, the host where the application is running, and the application itself. Cisco IOS XR Software allows SSM configuration for an arbitrary subset of the IP multicast address range 224.0.0.0 through 239.255.255.255. When an SSM range is defined, existing IP multicast receiver applications do not receive any traffic when they try to use addresses in the SSM range, unless the application is modified to use explicit (S,G) channel subscription.

DNS-based SSM Mapping

DNS-based SSM mapping enables you to configure the last hop router to perform a reverse DNS lookup to determine sources sending to groups (see the figure below). When DNS-based SSM mapping is configured, the router constructs a domain name that includes the group address G and performs a reverse lookup into the DNS. The router looks up IP address resource records (IP A RRs) to be returned for this constructed domain name and uses the returned IP addresses as the source addresses associated with this group. SSM mapping supports up to 20 sources for each group. The router joins all sources configured for a group.

Figure 3: DNS-based SSM Mapping



The SSM mapping mechanism that enables the last hop router to join multiple sources for a group can be used to provide source redundancy for a TV broadcast. In this context, the redundancy is provided by the last hop router using SSM mapping to join two video sources simultaneously for the same TV channel. However, to prevent the last hop router from duplicating the video traffic, it is necessary that the video sources utilize a server-side switchover mechanism where one video source is active while the other backup video source is passive. The passive source waits until an active source failure is detected before sending the video traffic for the TV channel. The server-side switchover mechanism, thus, ensures that only one of the servers is actively sending the video traffic for the TV channel.

To look up one or more source addresses for a group G that includes G1, G2, G3, and G4, the following DNS resource records (RRs) must be configured on the DNS server:

G4.G3.G2.G1 [*multicast-domain*] [*timeout*]	IN A *source-address-1*
	IN A *source-address-2*
	IN A *source-address-n*

The *multicast-domain* argument is a configurable DNS prefix. The default DNS prefix is in-addr.arpa. You should only use the default prefix when your installation is either separate from the internet or if the group names that you map are global scope group addresses (RFC 2770 type addresses that you configure for SSM) that you own.

The *timeout* argument configures the length of time for which the router performing SSM mapping will cache the DNS lookup. This argument is optional and defaults to the timeout of the zone in which this entry is configured. The timeout indicates how long the router will keep the current mapping before querying the DNS server for this group. The timeout is derived from the cache time of the DNS RR entry and can be configured for each group/source entry on the DNS server. You can configure this time for larger values if you want to minimize the number of DNS queries generated by the router. Configure this time for a low value if you want to be able to quickly update all routers with new source addresses.

Note See your DNS server documentation for more information about configuring DNS RRs.

To configure DNS-based SSM mapping in the software, you must configure a few global commands but no per-channel specific configuration is needed. There is no change to the configuration for SSM mapping if additional channels are added. When DNS-based SSM mapping is configured, the mappings are handled entirely by one or more DNS servers. All DNS techniques for configuration and redundancy management can be applied to the entries needed for DNS-based SSM mapping.

PIM-Bidirectional Mode

PIM BIDIR is a variant of the Protocol Independent Multicast (PIM) suite of routing protocols for IP multicast. In PIM, packet traffic for a multicast group is routed according to the rules of the mode configured for that multicast group. In bidirectional mode, traffic is only routed along a bidirectional shared tree that is rooted at the rendezvous point (RP) for the group. In PIM-BIDIR, the IP address of the RP acts as the key to having all routers establish a loop-free spanning tree topology rooted in that IP address. This IP address does not need to be a router, but can be any unassigned IP address on a network that is reachable throughout the PIM domain. Using this technique is the preferred configuration for establishing a redundant RP configuration for PIM-BIDIR.

Note In Cisco IOS XR Release 4.2.1, Anycast RP is not supported on PIM Bidirectional mode.

PIM-BIDIR is designed to be used for many-to-many applications within individual PIM domains. Multicast groups in bidirectional mode can scale to an arbitrary number of sources without incurring overhead due to the number of sources. PIM-BIDIR is derived from the mechanisms of PIM-sparse mode (PIM-SM) and shares many SPT operations. PIM-BIDIR also has unconditional forwarding of source traffic toward the RP upstream on the shared tree, but no registering process for sources as in PIM-SM. These modifications are necessary and sufficient to allow forwarding of traffic in all routers solely based on the (*, G) multicast routing entries. This feature eliminates any source-specific state and allows scaling capability to an arbitrary number of sources.

The traditional PIM protocols (dense-mode and sparse-mode) provided two models for forwarding multicast packets, source trees and shared trees. Source trees are rooted at the source of the traffic while shared trees are rooted at the rendezvous point. Source trees achieve the optimum path between each receiver and the source at the expense of additional routing information: an (S,G) routing entry per source in the multicast routing table. The shared tree provides a single distribution tree for all of the active sources. This means that traffic from different sources traverse the same distribution tree to reach the interested receivers, therefore reducing the amount of routing state in the network. This shared tree needs to be rooted somewhere, and the location of this root is the rendezvous point. PIM BIDIR uses shared trees as their main forwarding mechanism.

The algorithm to elect the designated forwarder is straightforward, all the PIM neighbors in a subnet advertise their unicast route to the rendezvous point and the router with the best route is elected. This effectively builds a shortest path between every subnet and the rendezvous point without consuming any multicast routing state (no (S,G) entries are generated). The designated forwarder election mechanism expects all of the PIM neighbors to be BIDIR enabled. In the case where one of more of the neighbors is not a BIDIR capable router, the election fails and BIDIR is disabled in that subnet.

Configuring PIM Per Interface States Limit

The PIM Per Interface States Limit sets a limit on creating OLEs for the PIM interface. When the set limit is reached, the group is not accounted against this interface but the group can exist in PIM context for some other interface.

The following configuration sets a limit on the number of routes for which the given interface may be an outgoing interface as a result of receiving a PIM J/P message.

```
router pim | pim6 [vrf <vrfname>]
interface <ifname>
          maximum route-interfaces <max> [threshold <threshold>] [<acl>]
!
!
```

where,

`<ifname>` is the interface name

`<max>` is the maximum limit on the groups

`<threshold>` is the threshold number of groups at which point a syslog warning message will be issued

`<acl>` provides an option for selective accounting. If provided, only groups or (S,G)s that are permitted by the ACL is accounted against the limit. Groups or (S, G)s that are denied by the ACL are not accounted against the limit. If not provided, all the groups are accounted against the limit.

The following messages are displayed when the threshold limit is reached for PIM:

```
pim[1157]: %ROUTING-IPV4_PIM-4-CAC_STATE_THRESHOLD : The interface GigabitEthernet0_2_0_0
threshold number (4) allowed states has been reached.
State creation will soon be throttled. Configure a higher state limit value or take steps
to reduce the number of states.

pim[1157]: %ROUTING-IPV4_PIM-3-CAC_STATE_LIMIT : The interface GigabitEthernet0_2_0_0 maximum
 number (5) of allowed states has been reached.
State creation will not be allowed from here on. Configure a higher maximum value or take
steps to reduce the number of states
```

Limitations

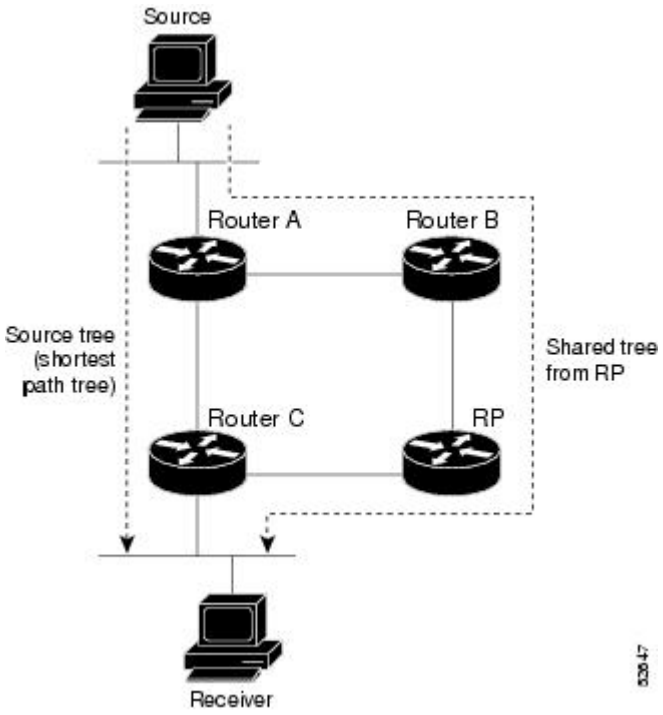
- If a user has configured a maximum of 20 groups and has reached the maximum number of groups, then no more groups/OLEs can be created. If the user now decreases the maximum number to 10, the 20 joins/OLE will remain and a message of reaching the max is displayed. No more joins/OLE can be added at this point until it has reached less than 10.

- If a user already has configured a maximum of 30 joins/OLEs and add a max of 20, the configuration occurs displaying a message that the max has been reached. No states will change but no more joins/OLEs can happen until the number is brought down below the maximum number of groups.
- Local interest joins are added, even if the limit has reached and is accounted for it.

PIM Shared Tree and Source Tree (Shortest Path Tree)

In PIM-SM, the rendezvous point (RP) is used to bridge sources sending data to a particular group with receivers sending joins for that group. In the initial setup of state, interested receivers receive data from senders to the group across a single data distribution tree rooted at the RP. This type of distribution tree is called a shared tree or rendezvous point tree (RPT) as illustrated in Figure 4: Shared Tree and Source Tree (Shortest Path Tree), on page 14 . Data from senders is delivered to the RP for distribution to group members joined to the shared tree.

Figure 4: Shared Tree and Source Tree (Shortest Path Tree)



Unless the **spt-threshold infinity** command is configured, this initial state gives way as soon as traffic is received on the leaf routers (designated router closest to the host receivers). When the leaf router receives traffic from the RP on the RPT, the router initiates a switch to a data distribution tree rooted at the source sending traffic. This type of distribution tree is called a **shortest path tree** or **source tree**. By default, the Cisco IOS XR Software switches to a source tree when it receives the first data packet from a source.

The following process describes the move from shared tree to source tree in more detail:

1. Receiver joins a group; leaf Router C sends a join message toward RP.
2. RP puts link to Router C in its outgoing interface list.
3. Source sends data; Router A encapsulates data in Register and sends it to RP.
4. RP forwards data down the shared tree to Router C and sends a join message toward Source. At this point, data may arrive twice at the RP, once encapsulated and once natively.
5. When data arrives natively (unencapsulated) at RP, RP sends a register-stop message to Router A.
6. By default, receipt of the first data packet prompts Router C to send a join message toward Source.
7. When Router C receives data on (S,G), it sends a prune message for Source up the shared tree.
8. RP deletes the link to Router C from outgoing interface of (S,G). RP triggers a prune message toward Source.

Join and prune messages are sent for sources and RPs. They are sent hop by hop and are processed by each PIM router along the path to the source or RP. Register and register-stop messages are not sent hop by hop. They are exchanged using direct unicast communication between the designated router that is directly connected to a source and the RP for the group.

Tip The **spt-threshold infinity** command lets you configure the router so that it never switches to the shortest path tree (SPT).

Multicast-Intact

The multicast-intact feature provides the ability to run multicast routing (PIM) when Interior Gateway Protocol (IGP) shortcuts are configured and active on the router. Both Open Shortest Path First, version 2 (OSPFv2), and Intermediate System-to-Intermediate System (IS-IS) support the multicast-intact feature. Multiprotocol Label Switching Traffic Engineering (MPLS-TE) and IP multicast coexistence is supported in Cisco IOS XR Software by using the **mpls traffic-eng multicast-intact** IS-IS or OSPF router command. See *Routing Configuration Guide for Cisco ASR 9000 Series Routers* for information on configuring multicast intact using IS-IS and OSPF commands.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGPs route the IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel (from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next-hops for use by PIM. These next-hops are called **mcast-intact next-hops**. The mcast-intact next-hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next hop to a PIM source.
- They are not published to the Forwarding Information Base (FIB).
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost mcast-intact next-hops to the RIB. This attribute applies even when the native next-hops have no IGP shortcuts.
- In IS-IS, the max-paths limit is applied by counting both the native and mcast-intact next-hops together. (In OSPFv2, the behavior is slightly different.)

Designated Routers

Cisco routers use PIM-SM to forward multicast traffic and follow an election process to select a designated router (DR) when there is more than one router on a LAN segment.

The designated router is responsible for sending PIM register and PIM join and prune messages toward the RP to inform it about host group membership.

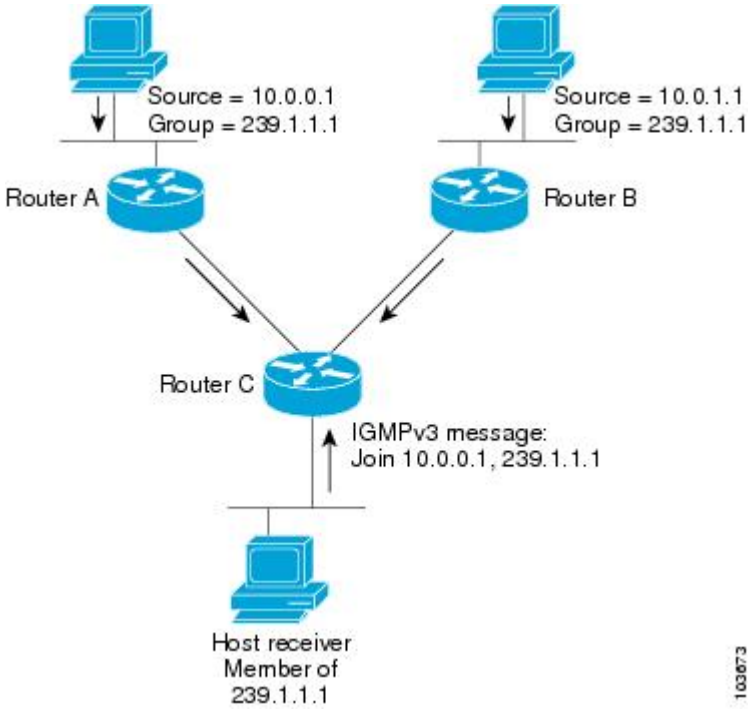
If there are multiple PIM-SM routers on a LAN, a designated router must be elected to avoid duplicating multicast traffic for connected hosts. The PIM router with the highest IP address becomes the DR for the LAN unless you choose to force the DR election by use of the **dr-priority** command. The DR priority option allows you to specify the DR priority of each router on the LAN segment (default priority = 1) so that the router with the highest priority is elected as the DR. If all routers on the LAN segment have the same priority, the highest IP address is again used as the tiebreaker.

Figure 5: Designated Router Election on a Multiaccess Segment, on page 16 illustrates what happens on a multiaccess segment. Router A (10.0.0.253) and Router B (10.0.0.251) are connected to a common multiaccess Ethernet segment with Host A (10.0.0.1) as an active receiver for Group A. As the Explicit Join model is used, only Router A, operating as the DR, sends joins to the RP to construct the shared tree for Group A. If Router B were also permitted to send (*, G) joins to the RP, parallel paths would be created and Host A would receive duplicate multicast traffic. When Host A begins to source multicast traffic to the group, the DR's responsibility is to send register messages to the RP. Again, if both routers were assigned the responsibility, the RP would receive duplicate multicast packets.

If the DR fails, the PIM-SM provides a way to detect the failure of Router A and to elect a failover DR. If the DR (Router A) were to become inoperable, Router B would detect this situation when its neighbor adjacency with Router A timed out. Because Router B has been hearing IGMP membership reports from Host A, it already has IGMP state for Group A on this interface and immediately sends a join to the RP when it becomes the new DR. This step reestablishes traffic flow down a new branch of the shared tree using Router B. Additionally, if Host A were sourcing traffic, Router B would initiate a new register process immediately after receiving the next multicast packet from Host A. This action would trigger the RP to join the SPT to Host A, using a new branch through Router B.

Tip Two PIM routers are neighbors if there is a direct connection between them. To display your PIM neighbors, use the **show pim neighbor** command in EXEC mode.

Figure 5: Designated Router Election on a Multiaccess Segment



Note DR election process is required only on multiaccess LANs. The last-hop router directly connected to the host is the DR.

Designated Router Election Using StickyDR

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Designated Router Election Using StickyDR	Release 7.4.1	With this feature, the router sends a PIM *hello* message with a special PIM DR priority value on a multi-access LAN. The router with this special DR priority value is always elected as the designated router. The traffic now flows in the same path even when a new router is added. This feature introduces the **sticky-dr** command.

When you enable PIM on an interface or reload a router, router periodically sends the PIM Hello messages on each interface. PIM Hello messages allow a router to learn neighboring PIM routers on each interface and elects a Designated Router (DR) based on the DR Priority. The DR election avoids duplicating multicast traffic for connected hosts.

Each time the DR is reelected, the multicast control tree sets up a new path and the multicast traffic flows in different direction.

With Sticky DR feature, the designated router remains the same and doesn't allow any other router to become the designated router. The multicast control tree does not set up a new path and the multicast traffic flows in same direction, thus avoids traffic loss. DR election isn't based on DR priority.

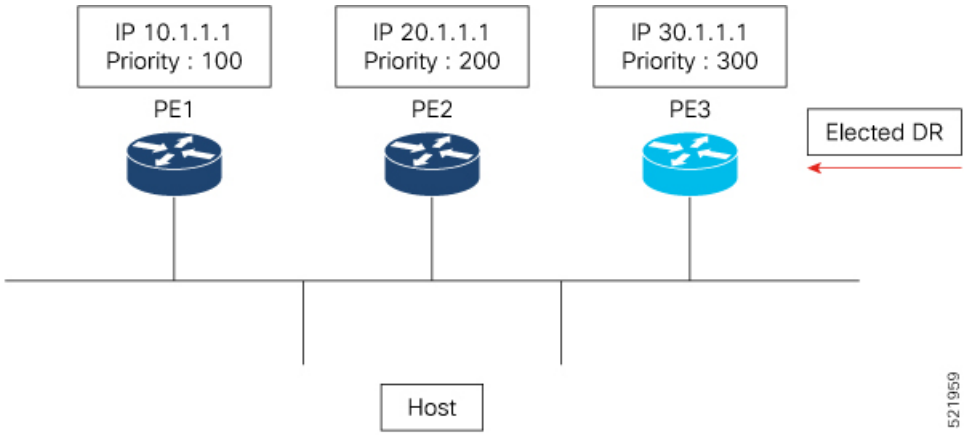
After you enable the sticky DR feature, the elected DR no longer advertises configured DR. Instead the router sends PIM Hello message with special PIM DR priority value which is reserved for Sticky PIM DR.

Restrictions

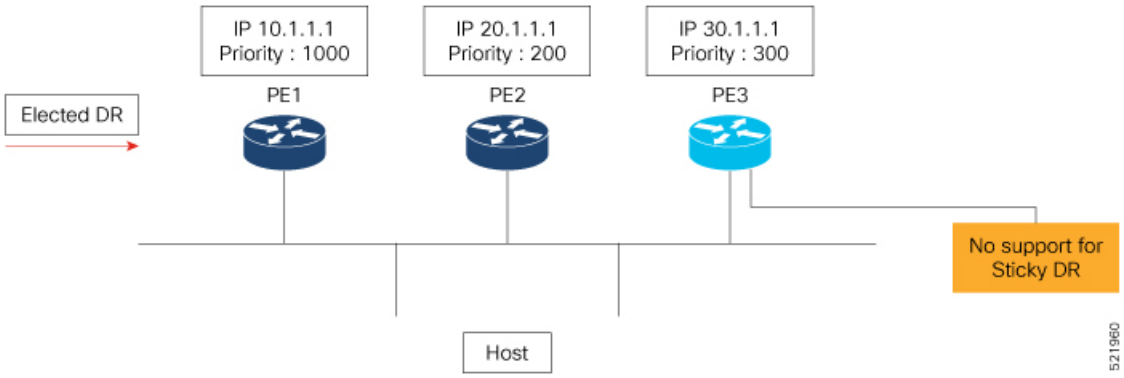
- The Sticky DR priority value is 4294967294. You must not configure DR priority with the value 4294967294 or any number greater than this value.

Topology

In this topology, PE1, PE2, PE3 are three PIM routers connected on a LAN. PE3 has the maximum priority and hence PE3 is elected as DR.

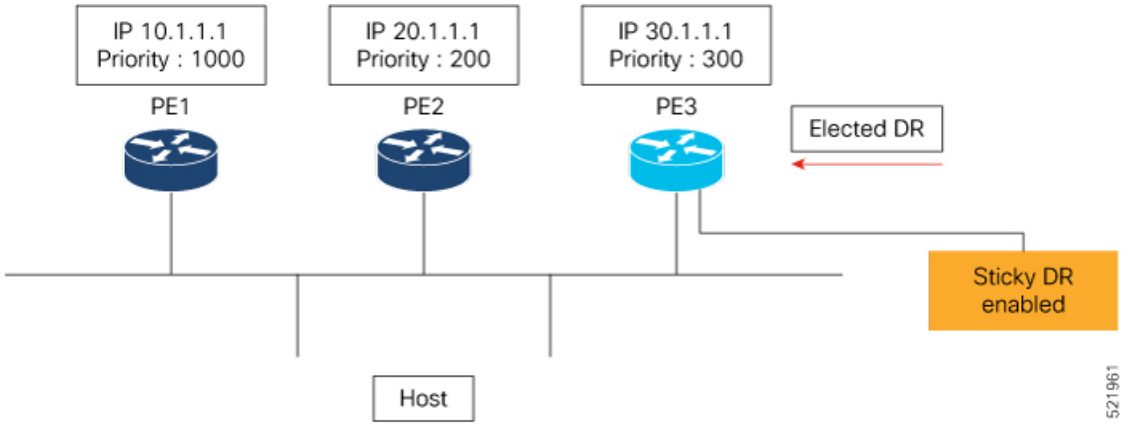


Now, when you configure PE1 with DR priority 1000, DR election process is re-initiated and PE1 becomes the new DR.



Every time a new DR is elected, the control tree computes a new path for traffic flow.

Now if you enable sticky DR on PE3, the PE3 remains the designated router irrespective of the DR priority of the PE devices.



In this example, the sticky DR is configured on PE3 and PE3 always remains as the DR.

Configuration

Let's configure sticky DR on PE3. To configure sticky DR on an interface, perform the following task:

```
Router# configure
Router(config)# router pim
Router(config-pim-default)# address-family ipv4
Router(config-pim-default-ipv4)# interface bundle-ether 72.1
Router(config-pim-ipv4-if)# sticky-dr
Router(config-ipv4-acl)# commit
```

Verification

The following output specifies that the Sticky DR is enabled on the interface and active:

```
Router# show pim interface bundle-ether 72.1 detail

PIM interfaces in VRF default
IP PIM Multicast Interface State
Flag: B - Bidir enabled, NB - Bidir disabled
      P - PIM Proxy enabled, NP - PIM Proxy disabled
      V - Virtual Interface, S - Sticky DR enabled
BFD State - State/Interval/Multiplier

Interface                  PIM  Nbr   Hello  DR
                                Count Intvl  Prior

Bundle-Ether72.1            on   2     30     100000
    Primary Address : 200.1.72.1
              Flags : B NP S V
                BFD : On/150 ms/3
                 DR : this system
  Propagation delay : 500
  Override Interval : 2500
        Hello Timer : 00:00:24
    Neighbor Filter : -
          Sticky DR : Configured, Active since Mon Jul 26 16:53:01 2021

-----------------------
Sticky DR Event History
-----------------------
Event                     State        Time
-----                     -----        ----
Dynamic Batch             Active       (null)
```

The following output specifies that the Sticky DR is enabled on the interface and is inactive:

```
Router#  show pim interface bundle-ether 72.1 detail

PIM interfaces in VRF default
IP PIM Multicast Interface State
Flag: B - Bidir enabled, NB - Bidir disabled
      P - PIM Proxy enabled, NP - PIM Proxy disabled
      V - Virtual Interface, S - Sticky DR enabled
BFD State - State/Interval/Multiplier

Interface                  PIM  Nbr   Hello  DR
                                Count Intvl  Prior

Bundle-Ether72.1            on   2     30     1
    Primary Address : 200.1.72.1
              Flags : B NP S V
                BFD : On/150 ms/3
```

```
               DR : 200.1.72.2
  Propagation delay : 500
  Override Interval : 2500
        Hello Timer : 00:00:18
    Neighbor Filter : -
          Sticky DR : Configured, Inactive
```

```
Router# show pim neighor detail

PIM neighbors in VRF default
Flag: B - Bidir capable, P - Proxy capable, DR - Designated Router,
      E - ECMP Redirect capable, S - Sticky DR Neighbor
      * indicates the neighbor created for this router

Neighbor Address             Interface              Uptime    Expires  DR pri      Flags

201.7.7.7*                   tunnel-mte1019         2d17h     00:01:36 1           (DR) B
E
  Expiry Timer: 00:01:05
201.7.7.7*                   tunnel-mte1001         2d17h     00:01:36 1           (DR) B
E
  Expiry Timer: 00:01:12
200.1.71.1*                  Bundle-Ether71.1       2d17h     00:01:31 99          (DR) B
  Expiry Timer: 00:00:02
200.1.71.2                   Bundle-Ether71.1       2d17h     00:01:19 1           B
BFD State: enabled
201.7.7.7*                   Loopback0              2d17h     00:01:41 1           (DR) B
E
  Expiry Timer: 00:01:12
201.202.7.7*                 Loopback1              2d17h     00:01:40 1           (DR) B
E
  Expiry Timer: 00:01:11
200.1.72.1*                  Bundle-Ether72.1       2d17h     00:01:15 -           (DR) B
S
  Expiry Timer: 00:01:21
```

Disable Sticky DR

To disable the sticky DR feature, perform the following task:

```
Router# configure
Router(config)# router pim
Router(config-pim-default)# address-family ipv4
Router(config-pim-default-ipv4)# interface bundle-ether 72.1
Router(config-pim-ipv4-if)# no sticky-dr
Router(config-ipv4-acl)# commit
```

To clear the DR stickiness and force the DR reelection, use the following command:

```
Router# clear pim interface bundle-ether 72.1 sticky-dr
```

Rendezvous Points

When PIM is configured in sparse mode, you must choose one or more routers to operate as a rendezvous point (RP). A rendezvous point is a single common root placed at a chosen point of a shared distribution tree, as illustrated in Figure 4: Shared Tree and Source Tree (Shortest Path Tree), on page 14. A rendezvous point can be either configured statically in each box or learned through a dynamic mechanism.

PIM DRs forward data from directly connected multicast sources to the rendezvous point for distribution down the shared tree. Data is forwarded to the rendezvous point in one of two ways:

- Encapsulated in register packets and unicast directly to the rendezvous point by the first-hop router operating as the DR
- Multicast forwarded by the RPF forwarding algorithm, described in the Reverse-Path Forwarding, on page 22, if the rendezvous point has itself joined the source tree.

The rendezvous point address is used by first-hop routers to send PIM register messages on behalf of a host sending a packet to the group. The rendezvous point address is also used by last-hop routers to send PIM join and prune messages to the rendezvous point to inform it about group membership. You must configure the rendezvous point address on all routers (including the rendezvous point router).

A PIM router can be a rendezvous point for more than one group. Only one rendezvous point address can be used at a time within a PIM domain. The conditions specified by the access list determine for which groups the router is a rendezvous point.

You can either manually configure a PIM router to function as a rendezvous point or allow the rendezvous point to learn group-to-RP mappings automatically by configuring Auto-RP or BSR. (For more information, see the Auto-RP, on page 21 section that follows and PIM Bootstrap Router, on page 22.)

Auto-RP

Automatic route processing (Auto-RP) is a feature that automates the distribution of group-to-RP mappings in a PIM network. This feature has these benefits:

- It is easy to use multiple RPs within a network to serve different group ranges.
- It allows load splitting among different RPs.
- It facilitates the arrangement of RPs according to the location of group participants.
- It avoids inconsistent, manual RP configurations that might cause connectivity problems.

Multiple RPs can be used to serve different group ranges or to serve as hot backups for each other. To ensure that Auto-RP functions, configure routers as candidate RPs so that they can announce their interest in operating as an RP for certain group ranges. Additionally, a router must be designated as an RP-mapping agent that receives the RP-announcement messages from the candidate RPs, and arbitrates conflicts. The RP-mapping agent sends the consistent group-to-RP mappings to all remaining routers. Thus, all routers automatically determine which RP to use for the groups they support.

Tip By default, if a given group address is covered by group-to-RP mappings from both static RP configuration, and is discovered using Auto-RP or PIM BSR, the Auto-RP or PIM BSR range is preferred. To override the default, and use only the RP mapping, use the **rp-address override** keyword.

Note If you configure PIM in sparse mode and do not configure Auto-RP, you must statically configure an RP as described in the Configuring a Static RP and Allowing Backward Compatibility, on page 130. When router interfaces are configured in sparse mode, Auto-RP can still be used if all routers are configured with a static RP address for the Auto-RP groups.

Note Auto-RP is not supported on VRF interfaces. Auto-RP Lite allows you to configure auto-RP on the CE router. It allows the PE router that has the VRF interface to relay auto-RP discovery, and announce messages across the core and eventually to the remote CE. Auto-RP is supported in only the IPv4 address family.

PIM Bootstrap Router

The PIM bootstrap router (BSR) provides a fault-tolerant, automated RP discovery and distribution mechanism that simplifies the Auto-RP process. This feature is enabled by default allowing routers to dynamically learn the group-to-RP mappings.

PIM uses the BSR to discover and announce RP-set information for each group prefix to all the routers in a PIM domain. This is the same function accomplished by Auto-RP, but the BSR is part of the PIM Version 2 specification. The BSR mechanism interoperates with Auto-RP on Cisco routers.

To avoid a single point of failure, you can configure several candidate BSRs in a PIM domain. A BSR is elected among the candidate BSRs automatically. Candidates use bootstrap messages to discover which BSR has the highest priority. The candidate with the highest priority sends an announcement to all PIM routers in the PIM domain that it is the BSR.

Routers that are configured as candidate RPs unicast to the BSR the group range for which they are responsible. The BSR includes this information in its bootstrap messages and disseminates it to all PIM routers in the domain. Based on this information, all routers are able to map multicast groups to specific RPs. As long as a router is receiving the bootstrap message, it has a current RP map.

Reverse-Path Forwarding

Reverse-path forwarding (RPF) is an algorithm used for forwarding multicast datagrams. It functions as follows:

- If a router receives a datagram on an interface it uses to send unicast packets to the source, the packet has arrived on the RPF interface.
- If the packet arrives on the RPF interface, a router forwards the packet out the interfaces present in the outgoing interface list of a multicast routing table entry.
- If the packet does not arrive on the RPF interface, the packet is silently discarded to prevent loops.

PIM uses both source trees and RP-rooted shared trees to forward datagrams; the RPF check is performed differently for each, as follows:

- If a PIM router has an (S,G) entry present in the multicast routing table (a source-tree state), the router performs the RPF check against the IP address of the source for the multicast packet.
- If a PIM router has no explicit source-tree state, this is considered a shared-tree state. The router performs the RPF check on the address of the RP, which is known when members join the group.

Sparse-mode PIM uses the RPF lookup function to determine where it needs to send joins and prunes. (S,G) joins (which are source-tree states) are sent toward the source. (*,G) joins (which are shared-tree states) are sent toward the RP.

Multicast Non-Stop Routing

Multicast Non-Stop Routing (NSR) enables the router to synchronize the multicast routing tables on both the active and standby RSPs so that during an HA scenario like an RSP failover there is no loss of multicast data. Multicast NSR is enabled through the multicast processes being hot standby. Multicast NSR supports both Zero Packet Loss (ZPL) and Zero Topology Loss (ZTL). With Multicast NSR, there is less CPU churn and no multicast session flaps during a failover event.

Multicast NSR is enabled by default, however, if any unsupported features like BNG or Snooping are configured, Multicast performs Non-Stop Forwarding (NSF) functionality during failover events. When Multicast NSR is enabled, multicast routing state is synchronized between the active and standby RSPs. Once the synchronization occurs, each of the multicast processes signal the NSR readiness to the system. For the multicast processes to support NSR, the processes must be hot standby compliant. That is, the processes on active and standby RSPs both have to be in synchronization at all times. The active RSP receives packets from the network and makes local decisions while the standby receives packet from the network and synchronizes it with the active RSPs for all the local decisions. Once the state is determined, a check is performed to verify if the states are synchronized. If the states are synchronized, a signal in the form NSR_READY is conveyed to the NSR system.

With NSR, in the case of a failover event, routing changes are updated to the forwarding plane immediately. With NSF, there is an NSF hold time delay before routing changes can be updated.

Non-Supported Features

The following features are unsupported on NG NSR:

- IGMP and MLD Snooping
- BNG

Restriction

NSF is enabled by default. You can’t configure **nsr** under multicast-routing manually.

Failure Scenarios in NSR

If a switchover occurs before all multicast processes issue an NSR_READY signal, the proceedings revert back to the existing NSF behavior. Also, on receiving the GO_ACTIVE signal from the multicast processes, the following events occur in processes that have not signaled NSR_READY:

1. IGMP starts the NSF timer for one minute.
2. PIM starts the NSF timer for two minutes.
3. MSDP resets all peer sessions that are not synchronized.

Multicast VPN

Multicast VPN (MVPN) provides the ability to dynamically provide multicast support over MPLS networks. MVPN introduces an additional set of protocols and procedures that help enable a provider to support multicast traffic in a VPN.

Note PIM-Bidir is not supported on MVPN.

There are two ways MCAST VPN traffic can be transported over the core network:

- Rosen GRE (native): MVPN uses GRE with unique multicast distribution tree (MDT) forwarding to enable scalability of native IP Multicast in the core network. MVPN introduces multicast routing information to the VPN routing and forwarding table (VRF), creating a Multicast VRF. In Rosen GRE, the MCAST customer packets (c-packets) are encapsulated into the provider MCAST packets (p-packets), so that the PIM protocol is enabled in the provider core, and mrib/mfib is used for forwarding p-packets in the core.
- MLDP ones (Rosen, partition): MVPN allows a service provider to configure and support multicast traffic in an MPLS VPN environment. This type supports routing and forwarding of multicast packets for each individual VPN routing and forwarding (VRF) instance, and it also provides a mechanism to transport VPN multicast packets across the service provider backbone. In the MLDP case, the regular label switch path forwarding is used, so core does not need to run PIM protocol. In this scenario, the c-packets are encapsulated in the MPLS labels and forwarding is based on the MPLS Label Switched Paths (LSPs) ,similar to the unicast case.

In both the above types, the MVPN service allows you to build a Protocol Independent Multicast (PIM) domain that has sources and receivers located in different sites.

To provide Layer 3 multicast services to customers with multiple distributed sites, service providers look for a secure and scalable mechanism to transmit customer multicast traffic across the provider network. Multicast VPN (MVPN) provides such services over a shared service provider backbone, using native multicast technology similar to BGP/MPLS VPN.

In addition to all the ethernet based line cards, Multicast VPN is also supported on the Cisco ASR 9000 Series SPA Interface Processor-700 card from the Cisco IOS XR Software Release 4.0 onwards. Cisco ASR 9000 Series SPA Interface Processor-700 enables the Cisco ASR 9000 Series Routers to support multiple legacy services (such as TDM and ATM) on a router that is primarily designed for Ethernet networks. Cisco ASR 9000 Series SPA Interface Processor-700 is QFP-based and therefore has the flexibility and service scale offered by Cisco ASIC and the reliability of Cisco IOS XR Software.

MVPN emulates MPLS VPN technology in its adoption of the multicast domain (MD) concept, in which provider edge (PE) routers establish virtual PIM neighbor connections with other PE routers that are connected to the same customer VPN. These PE routers thereby form a secure, virtual multicast domain over the provider network. Multicast traffic is then transmitted across the core network from one site to another, as if the traffic were going through a dedicated provider network.

Multi-instance BGP is supported on multicast and MVPN. Multicast-related SAFIs can be configured on multiple BGP instances.

Multicast VPN Routing and Forwarding

Dedicated multicast routing and forwarding tables are created for each VPN to separate traffic in one VPN from traffic in another.

The VPN-specific multicast routing and forwarding database is referred to as **MVRF**. On a PE router, an MVRF is created when multicast is enabled for a VRF. Protocol Independent Multicast (PIM), and Internet Group Management Protocol (IGMP) protocols run in the context of MVRF, and all routes created by an MVRF protocol instance are associated with the corresponding MVRF. In addition to VRFs, which hold

VPN-specific protocol states, a PE router always has a global VRF instance, containing all routing and forwarding information for the provider network.

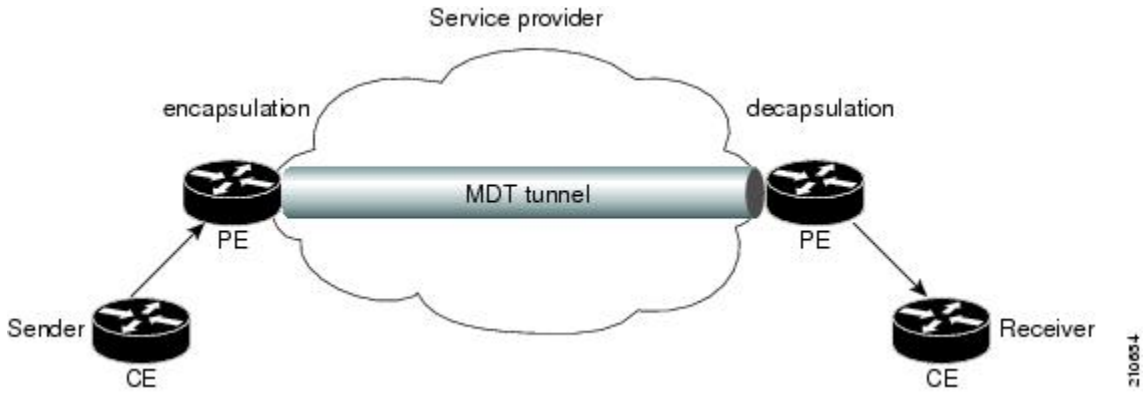
Multicast Distribution Tree Tunnels

The multicast distribution tree (MDT) can span multiple customer sites through provider networks, allowing traffic to flow from one source to multiple receivers. For MLDP, the MDT tunnel are called Labeled MDT (LMDT).

Secure data transmission of multicast packets sent from the customer edge (CE) router at the ingress PE router is achieved by encapsulating the packets in a provider header and transmitting the packets across the core. At the egress PE router, the encapsulated packets are decapsulated and then sent to the CE receiving routers.

Multicast distribution tree (MDT) tunnels are point-to-multipoint. A MDT tunnel interface is an interface that MVRF uses to access the multicast domain. It can be deemed as a passage that connects an MVRF and the global MVRF. Packets sent to an MDT tunnel interface are received by multiple receiving routers. Packets sent to an MDT tunnel interface are encapsulated, and packets received from a MDT tunnel interface are decapsulated.

Figure 6: Virtual PIM Peer Connection over an MDT Tunnel Interface



Encapsulating multicast packets in a provider header allows PE routers to be kept unaware of the packets' origin—all VPN packets passing through the provider network are viewed as native multicast packets and are routed based on the routing information in the core network. To support MVPN, PE routers only need to support native multicast routing.

MVPN also supports optimized VPN traffic forwarding for high-bandwidth applications that have sparsely distributed receivers. A dedicated multicast group can be used to encapsulate packets from a specific source, and an optimized MDT can be created to send traffic only to PE routers connected to interested receivers. This is referred to **data MDT**.

InterAS Support on Multicast VPN

The Multicast VPN Inter-AS Support feature enables service providers to provide multicast connectivity to VPN sites that span across multiple autonomous systems. This feature was added to MLDP profile that enables Multicast Distribution Trees (MDTs), used for Multicast VPNs (MVPNs), to span multiple autonomous systems.

There are two types of MVPN inter-AS deployment scenarios:

- Single-Provider Inter-AS—A service provider whose internal network consists of multiple autonomous systems.

- Intra-Provider Inter-AS—Multiple service providers that need to coordinate their networks to provide inter-AS support.

To establish a Multicast VPN between two autonomous systems, a MDT-default tunnel must be setup between the two PE routers. The PE routers accomplish this by joining the configured MDT-default group. This MDT-default group is configured on the PE router and is unique for each VPN. The PIM sends the join based on the mode of the groups, which can be PIM SSM, or sparse mode.

Note PIM-Bidir is not supported on MVPN.

Benefits of MVPN Inter-AS Support

The MVPN Inter-AS Support feature provides these benefits to service providers:

- Increased multicast coverage to customers that require multicast to span multiple services providers in an MPLS Layer 3 VPN service.
- The ability to consolidate an existing MVPN service with another MVPN service, as in the case of a company merger or acquisition.

InterAS Option A

InterAS Option A is the basic Multicast VPN configuration option. In this option, the PE router partially plays the Autonomous System Border Router (ASBR) role in each Autonomous System (AS). Such a PE router in each AS is directly connected through multiple VRF bearing subinterfaces. MPLS label distribution protocol need not run between these InterAS peering PE routers. However, an IGP or BGP protocol can be used for route distribution under the VRF.

The Option A model assumes direct connectivity between PE routers of different autonomous systems. The PE routers are attached by multiple physical or logical interfaces, each of which is associated with a given VPN (through a VRF instance). Each PE router, therefore, treats the adjacent PE router like a customer edge (CE) router. The standard Layer 3 MPLS VPN mechanisms are used for route redistribution with each autonomous system; that is, the PEs use exterior BGP (eBGP) to distribute unlabeled IPv4 addresses to each other.

Note Option A allows service providers to isolate each autonomous system from the other. This provides better control over routing exchanges and security between the two networks. However, Option A is considered the least scalable of all the inter-AS connectivity options.

InterAS Option B

InterAS Option B is a model that enables VPNv4 route exchanges between the ASBRs. This model also distributes BGP MVPN address family. In this model, the PE routers use internal BGP (iBGP) to redistribute labeled VPNv4 routes either to an ASBR or to route reflector of which an ASBR is a client. These ASBRs use multiprotocol eBGP (MP-eBGP) to advertise VPNv4 routes into the local autonomous systems. The MP-eBGP advertises VPNv4 prefix and label information across the service provider boundaries. The advertising ASBR router replaces the two-level label stack, which it uses to reach the originating PE router and VPN destination in the local autonomous system, with a locally allocated label before advertising the

VPNv4 route. This replacement happens because the next-hop attribute of all routes advertised between the two service providers is reset to the ASBR router's peering address, thus making the ASBR router the termination point of the label-switched path (LSP) for the advertised routes. To preserve the LSP between ingress and egress PE routers, the ASBR router allocates a local label that is used to identify the label stack of the route within the local VPN network. This newly allocated label is set on packets sent towards the prefix from the adjacent service provider.

Note Option B enables service providers to isolate both autonomous systems with the added advantage that it scales to a higher degree than Option A.

In the InterAS Option B model, only BGP-AD profiles are supported:

- MLDP MS-PMSI MP2MP with BGP-AD (profile 4)
- Rosen GRE with or without BGP-AD (profile 9)

Note Profile 9 is only supported with leaking root address into IGP.

Note MLDP MS-PMSI MP2MP with BGP-AD (profile 5) is not supported.

InterAS Option C

InterAS Option C allows exchange of VPNv4 routes between router reflectors (RRs) using multihop eBGP peering sessions. In this model, the MP-eBGP exchange of VPNv4 routes between the RRs of different autonomous systems is combied with the next hops for these routes exchanges between corresponding ASBR routers. This model also distributes BGP MVPN address family along with VPNv4. This model neither allows the VPNv4 routes to be maintained nor distributes by the ASBRs. ASBRs maintains labeled IPv4 routes to the PE routers within its autonomous system and uses eBGP to distribute these routes to other autonomous systems. In any transit autonomous systems, the ASBRs uses eBGP to pass along the labeled IPv4 routes, resulting in the creation of a LSP from the ingress PE router to the egress PE router.

Option C model uses the multihop functionality to allow the establishment for MP-eBGP peering sessions as the RRs of different autonomous systems are not directly connected. The RRs also do not reset the next-hop attribute of the VPNv4 routes when advertising them to adjacent autonomous systems as these do not attract the traffic for the destinations that they advertise, making it mandatory to enable the exchange of next hops. These are just a relay station between the source and receiver PEs. The PE router next-hop addresses for the VPNv4 routes, thus, are exchanged between ASBR routers. The exchange of these addresses between autonomous systems is accomplished by redistributing the PE router /32 addresses between the autonomous systems or by using BGP label distribution.

Note Option C normally is deployed only when each autonomous system belongs to the same overall authority, such as a global Layer 3 MPLS VPN service provider with global autonomous systems.

Supported profiles for Inter-AS models

The table below displays supported profiles for each Inter-AS models:

Profiles	InterAS Option A model	InterAS Option B model	InterAS Option C model
Profile 0	Supported	Not supported	Supported
Profile 1	Supported	Supported	Supported
Profile 2	Supported	Supported	Supported
Profile 3	Supported	Not supported	Supported
Profile 4	Supported	Supported	Supported
Profile 5	Supported	Supported	Supported
Profile 6	Supported	Supported	Supported
Profile 7	Supported	Supported	Supported
Profile 8	Supported	Not supported	Not supported
Profile 9	Supported	Supported	Supported
Profile 10	Supported	Not supported	Not supported
Profile 11	Supported	Not supported	Supported
Profile 12	Supported	Supported	Supported
Profile 13	Supported	Supported	Supported
Profile 14	Supported	Supported	Supported
Profile 15	Supported	Supported	Supported
Profile 16	Supported	Not supported	Not supported
Profile 17	Supported	Supported	Supported
Profile 18	Supported	Not supported	Not supported
Profile 19	Supported	Not supported	Not supported

Profiles	InterAS Option A model	InterAS Option B model	InterAS Option C model
Profile 20	Supported	Not supported	Not supported
Profile 21	Supported	Not supported	Not supported
Profile 22	Supported	Not supported	Not supported
Profile 24	Supported	Not supported	Not supported
Profile 26	Supported	Not supported	Not supported

IPv6 Connectivity over MVPN

On the Cisco ASR 9000 Series Routers, in Cisco IOS XR Software starting Release 4.2.1, IPv6 connectivity is supported between customer sites over an IPv4-only core network with a default VRF. VPN PE routers interoperate between the two address families, with control and forwarding actions between IPv4-encapsulated MDTs and IPv6 customer routes. IPv6 users can configure IPv6-over-IPv4 multicast VPN support through BGP.

In Cisco IOS XR Software, MVPNv6 can have a separate data mdt group configured, which can be different from MVPNv4. But both MVPNv6 and MVPNv4 must have the same default mdt group configured.

The configuration example below shows MVPNv6 data mdt :

```
vrf cisco-sjc1
  address-family ipv4
   mdt data 226.8.3.0/24 threshold 5
   mdt default ipv4 226.8.0.1
  !
  address-family ipv6
   mdt data 226.8.4.0/24 threshold 5
   mdt default ipv4 226.8.0.1
  !
```

BGP Requirements

PE routers are the only routers that need to be MVPN-aware and able to signal remote PEs with information regarding the MVPN. It is fundamental that all PE routers have a BGP relationship with each other, either directly or through a route reflector, because the PE routers use the BGP peering address information to derive the RPF PE peer within a given VRF.

PIM-SSM MDT tunnels cannot be set up without a configured BGP MDT address-family, because you establish the tunnels, using the BGP connector attribute.

Note A router being RR and PE at that same time for BGP mVPN implementation is not supported, a type 7 and type 6 IPv4 mVPN route is not advertised by a RR, which is also a PE router, if the PE router has the VRF locally configured and when there is a local receiver.

Use full mesh for iBGP mVPN address-family or elect any core (P) router to be the RR.

See the Implementing BGP on Cisco IOS XR Software module of the *Routing Configuration Guide for Cisco ASR 9000 Series Routers* for information on BGP support for Multicast VPN.

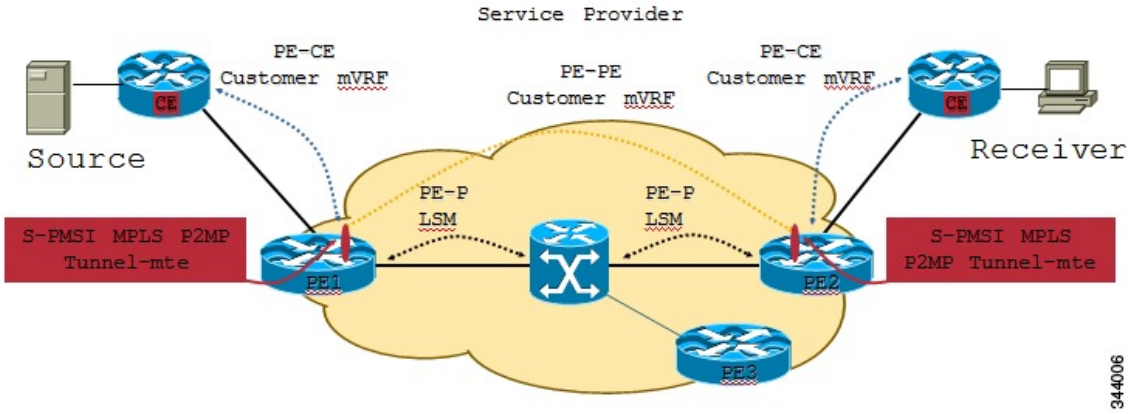
MVPN Static P2MP TE

This feature describes the Multicast VPN (MVPN) support for Multicast over Point-to-Multipoint -Traffic Engineering (P2MP-TE). Currently, Cisco IOS-XR Software supports P2MP-TE only in the Global table and the (S,G) route in the global table can be mapped to P2MP-TE tunnels. However, this feature now enables service providers to use P2MP-TE tunnels to carry VRF multicast traffic. Static mapping is used to map VRF (S, G) traffic to P2MP-TE tunnels, and BGP-AD is used to send P2MP BGP opaque that includes VRF-based P2MP FEC as MDT Selective Provider Multicast Service Interface (S-PMSI).

The advantages of the MVPN support for Multicast over P2MP-TE are:

- Supports traffic engineering such as bandwidth reservation, bandwidth sharing, forwarding replication, explicit routing, and Fast ReRoute (FRR).
- Supports the mapping of multiple multicast streams onto tunnels.

Figure 7: Multicast VRF



On PE1 router, multicast S,G (video) traffic is received on a VRF interface. The multicast S,G routes are statically mapped to P2MP-TE tunnels. The head-end then originates an S-PMSI (Type-3) BGP-AD route, for each of the S,Gs, with a PMSI Tunnel Attribute (PTA) specifying the P2MP-TE tunnel as the core-tree. The type of the PTA is set to RSVP-TE P2MP LSP and the format of the PTA Tunnel-identifier <Extended Tunnel ID, Reserved, Tunnel ID, P2MP ID>, as carried in the RSVP-TE P2MP LSP SESSION Object. Multiple S,G A-D routes can have the same PMSI Tunnel Attribute.

The tail-end PEs (PE2, PE3) receive and cache these S-PMSI updates (sent by all head-end PEs). If there is an S,G Join present in the VRF, with the Upstream Multicast Hop (UMH) across the core, then the PE looks for an S-PMSI announcement from the UMH. If an S-PMSI route is found with a P2MP-TE PTA, then the

PE associates the tail label(s) of the Tunnel, with that VRF. When a packet arrives on the P2MP-TE tunnel, the tail-end removes the label and does an S,G lookup in the 'associated' VRF. If a match is found, the packet is forwarded as per its outgoing information.

Multitopology Routing

Multitopology routing allows you to manipulate network traffic flow when desirable (for example, to broadcast duplicate video streams) to flow over non-overlapping paths.

At the core of multitopology routing technology is router space infrastructure (RSI). RSI manages the global configuration of routing tables. These tables are hierarchically organized into VRF tables under logical routers. By default, RSI creates tables for unicast and multicast for both IPv4 and IPv6 under the default VRF. Using multitopology routing, you can configure named topologies for the default VRF.

PIM uses a routing policy that supports matching on source or group address to select the topology in which to look up the reverse-path forwarding (RPF) path to the source. If you do not configure a policy, the existing behavior (to select a default table) remains in force.

Currently, IS-IS and PIM routing protocols alone support multitopology-enabled network.

Multicast VPN Extranet Routing

Multicast VPN (MVPN) extranet routing lets service providers distribute IP multicast content from one enterprise site to another across a multicast VRF. In other words, this feature provides capability to seamlessly hop VRF boundaries to distribute multicast content end to end.

Unicast extranet can be achieved simply by configuring matching route targets across VRFs. However, multicast extranet requires such configuration to resolve route lookups across VRFs in addition to the following:

- Maintain multicast topology maps across VRFs.
- Maintain multicast distribution trees to forward traffic across VRFs.

Information About Extranets

An extranet can be viewed as part of an enterprise intranet that is extended to users outside the enterprise. A VPN is used as a way to do business with other enterprises and with customers, such as selling products and maintaining strong business partnerships. An extranet is a VPN that connects to one or more corporate sites to external business partners or suppliers to securely share a designated part of the enterprise’s business information or operations.

MVPN extranet routing can be used to solve such business problems as:

- Inefficient content distribution between enterprises.
- Inefficient content distribution from service providers or content providers to their enterprise VPN customers.

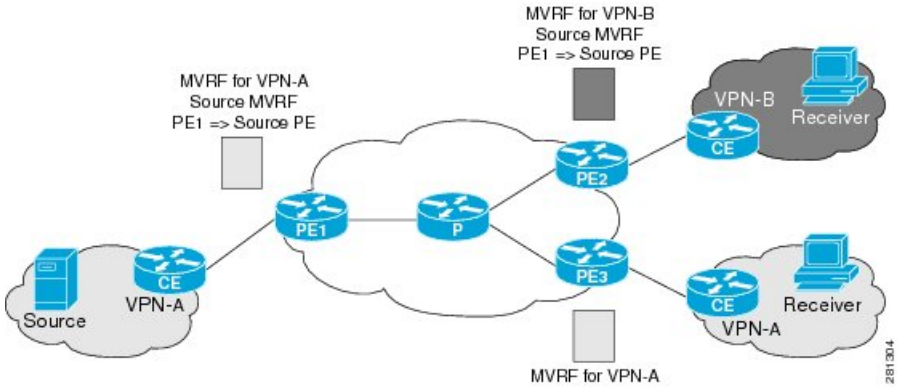
MVPN extranet routing provides support for IPv4 and IPv6 address family.

An extranet network requires the PE routers to pass traffic across VRFs (labeled “P” in Figure 8: Components of an Extranet MVPN, on page 32). Extranet networks can run either IPv4 or IPv6, but the core network always runs only IPv4 active multicast.

Note Multicast extranet routing is not supported on BVI interfaces.

Extranet Components

Figure 8: Components of an Extranet MVPN



MVRF—Multicast VPN routing and forwarding (VRF) instance. An MVRF is a multicast-enabled VRF. A VRF consists of an IP routing table, a derived forwarding table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols that determine what goes into the forwarding table. In general, a VRF includes the routing information that defines a customer VPN site that is attached to a provider edge (PE) router.

Source MVRF—An MVRF that can reach the source through a directly connected customer edge (CE) router.

Receiver MVRF—An MVRF to which receivers are connected through one or more CE devices.

Source PE—A PE router that has a multicast source behind a directly connected CE router.

Receiver PE—A PE router that has one or more interested receivers behind a directly connected CE router.

Information About the Extranet MVPN Routing Topology

In unicast routing of peer-to-peer VPNs, BGP routing protocol is used to advertise VPN IPv4 and IPv6 customer routes between provider edge (PE) routers. However, in an MVPN extranet peer-to-peer network, PIM RPF is used to determine whether the RPF next hop is in the same or a different VRF and whether that source VRF is local or remote to the PE.

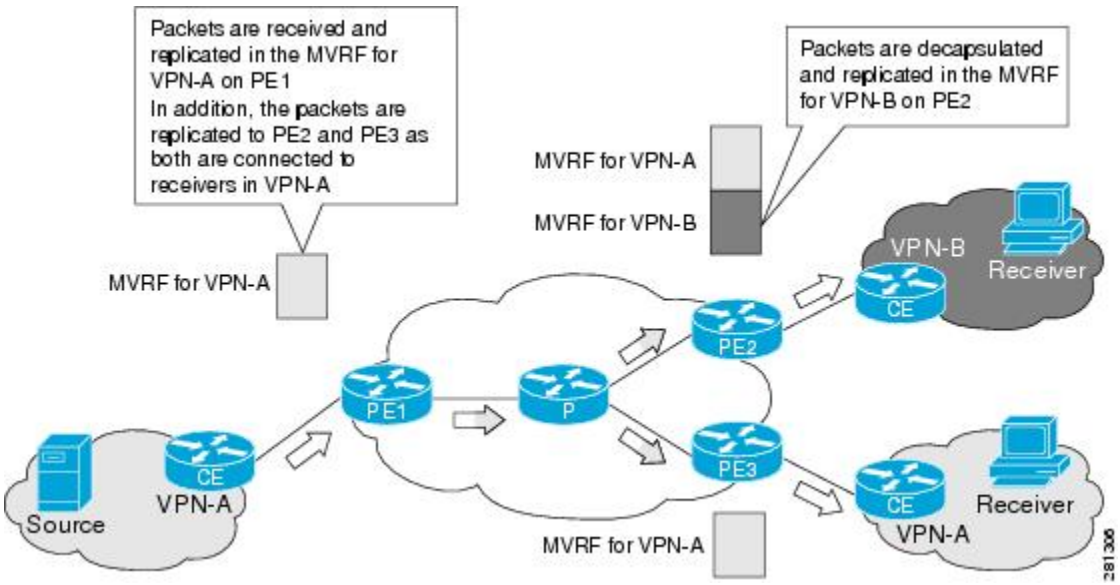
Source MVRF on a Receiver PE Router

To provide extranet MVPN services to enterprise VPN customers by configuring a source MVRF on a receiver PE router, you would complete the following procedure:

- On a receiver PE router that has one or more interested receivers in an extranet site behind a directly connected CE router, configure an MVRF that has the same default MDT group as the site connected to the multicast source.
- On the receiver PE router, configure the same unicast routing policy to import routes from the source MVRF to the receiver MVRF.

If the originating MVRF of the RPF next hop is local (source MVRF at receiver PE router), the join state of the receiver VRFs propagates over the core by using the default multicast distribution tree (MDT) of the source VRF. Figure 9: Source MVRF at the Receiver PE Router, on page 33 illustrates the flow of multicast traffic in an extranet MVPN topology where the source MVRF is configured on a receiver PE router (source at receiver MVRF topology). An MVRF is configured for VPN-A and VPN-B on PE2, a receiver PE router. A multicast source behind PE1, the source PE router, is sending out a multicast stream to the MVRF for VPN-A, and there are interested receivers behind PE2, the receiver PE router for VPN-B, and also behind PE3, the receiver PE router for VPN-A. After PE1 receives the packets from the source in the MVRF for VPN-A, it replicates and forwards the packets to PE2 and PE3. The packets received at PE2 in VPN-A are decapsulated and replicated to receivers in VPN-B.

Figure 9: Source MVRF at the Receiver PE Router



Receiver MVRF on the Source PE Router

To provide extranet MVPN services to enterprise VPN customers by configuring the receiver MVRF on the source PE router, complete the following procedure:

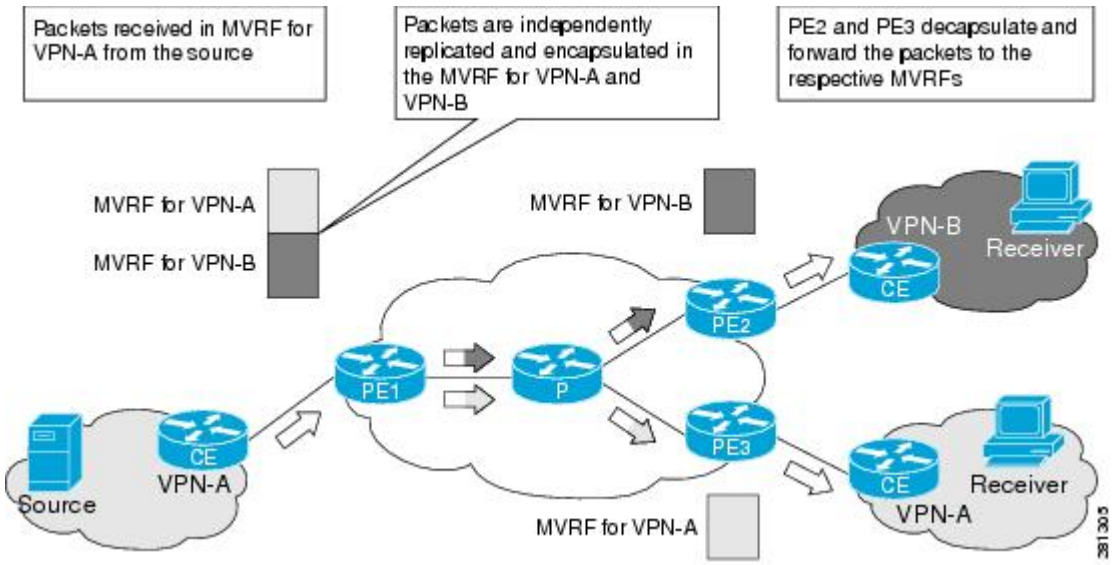
- For each extranet site, you would configure an additional MVRF on the source PE router, which has the same default MDT group as the receiver MVRF, if the MVRF is not already configured on the source PE.
- In the receiver MVRF configuration, you would configure the same unicast routing policy on the source and receiver PE routers to import routes from the source MVRF to the receiver MVRF.

If the originating MVRF of the RPF next-hop is remote (receiver MVRF on the source PE router), then the join state of receiver VRFs propagates over the core through the MDT of each receiver.

Figure 10: Receiver MVRF at the Source PE Router Receiver, on page 34 illustrates the flow of multicast traffic in an extranet MVPN topology where a receiver MVRF is configured on the source PE router. An MVRF is configured for VPN-A and VPN-B on PE1, the source PE router. A multicast source behind PE1 is sending out a multicast stream to the MVRF for VPN-A, and there are interested receivers behind PE2 and PE3, the receiver PE routers for VPN-B and VPN-A, respectively. After PE1 receives the packets from the source in the MVRF for VPN-A, it independently replicates and encapsulates the packets in the MVRF for

VPN-A and VPN-B and forwards the packets. After receiving the packets from this source, PE2 and PE3 decapsulate and forward the packets to the respective MVRFs.

Figure 10: Receiver MVRF at the Source PE Router Receiver



For more information, see also Configuring MVPN Extranet Routing, on page 184 and Configuring MVPN Extranet Routing: Example, on page 241.

RPF Policies in an Extranet

RPF policies can be configured in receiver VRFs to bypass RPF lookup in receiver VRFs and statically propagate join states to specified source VRF. Such policies can be configured to pick a source VRF based on either multicast group range, multicast source range, or RP address.

For more information about configuration of RFP policies in extranets, see Configuring RPL Policies in Receiver VRFs to Propagate Joins to a Source VRF: Example, on page 243 and Configuring RPL Policies in Receiver VRFs on Source PE Routers to Propagate Joins to a Source VRF: Example, on page 246.

Multicast VPN Hub and Spoke Topology

Hub and spoke topology is an interconnection of two categories of sites — Hub sites and Spoke sites. The routes advertised across sites are such that they achieve connectivity in a restricted hub and spoke fashion. A spoke can interact only with its hub because the rest of the network (that is, other hubs and spokes) appears hidden behind the hub.

The hub and spoke topology can be adopted for these reasons:

- Spoke sites of a VPN customer receives all their traffic from a central (or Hub) site hosting services such as server farms.
- Spoke sites of a VPN customer requires all the connectivity between its spoke sites through a central site. This means that the hub site becomes a transit point for interspoke connectivity.
- Spoke sites of a VPN customer do not need any connectivity between spoke sites. Hubs can send and receive traffic from all sites but spoke sites can send or receive traffic only to or from Hub sites.

Realizing the Hub and Spoke Topology

Hub and Spoke implementation leverages the infrastructure built for MVPN Extranet. The regular MVPN follows the model in which packets can flow from any site to the other sites. But Hub and Spoke MVPN will restrict traffic flows based on their subscription.

A site can be considered to be a geographic location with a group of CE routers and other devices, such as server farms, connected to PE routers by PE-CE links for VPN access. Either every site can be placed in a separate VRF, or multiple sites can be combined in one VRF on the PE router.

By provisioning every site in a separate VRF, you can simplify the unicast and multicast Hub and Spoke implementation. Such a configuration brings natural protection from traffic leakage - from one spoke site to another. Cisco IOS XR Software implementation of hub and spoke follows the one- site-to-one VRF model. Any site can be designated as either a hub or spoke site, based on how the import or export of routes is setup. Multiple hub and spoke sites can be collated on a given PE router.

Unicast Hub and Spoke connectivity is achieved by the spoke sites importing routes from only Hub sites, and Hub sites importing routes from all sites. As the spoke sites do not exchange routes, spoke to spoke site traffic cannot flow. If interspoke connectivity is required, hubs can choose to re-inject routes learned from one spoke site into other spoke site.

MVPN Hub and Spoke is achieved by separating core tunnels, for traffic sourced from hub sites, and spoke sites. MDT hub is the tunnel carrying traffic sourced from all Hub sites, and MDT spoke carries traffic sourced from all spoke sites. Such tunnel end-points are configured on all PEs participating in hub and spoke topology. If spoke sites do not host any multicast sources or RPs, provisioning of MDT Spoke can be completely avoided at all such routers.

Once these tunnels are provisioned, multicast traffic path will be policy routed in this manner:

1. Hub sites will send traffic to only MDT Hub.
2. Spoke sites will send traffic to only MDT Spoke.
3. Hub sites will receive traffic from both tunnels.
4. Spoke sites will receive traffic from only MDT Hub.

These rules ensure that hubs and spokes can send and receive traffic to or from each other, but direct spoke to spoke communication does not exist. If required, interspoke multicast can flow by turning around the traffic at Hub sites.

These enhancements are made to the Multicast Hub and Spoke topology in Cisco IOS XR Software Release 4.0:

- Auto-RP and BSR are supported across VRFs that are connected through extranet. It is no longer restricted to using static RP only.
- MP-BGP can publish matching import route-targets while passing prefix nexthop information to RIB.
- Route policies can use extended community route targets instead of IP address ranges.
- Support for extranet v4 data mdt was included so that data mdt in hub and spoke can be implemented.

Flexible Algorithm for MLDP

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Algorithm for Multicast VPN profiles	Release 7.5.2	Flexible Algorithm is now available for the following profiles: • Profile 12: Default MDT - MLDP - P2MP - BGP-AD - BGP C-Mcast Signaling • Profile 14: Partitioned MDT - MLDP P2MP - BGP-AD - BGP C-Mcast Signaling
Flexible Algorithm for MLDP	Release 7.5.1	This feature gives you the flexibility to customize the metrics that IGP uses to route traffic for MLDP tunnels. With this feature, your router can generate two multicast streams for the same feed, thus ensuring low latency and high availability of multicast traffic. This feature introduces the keyword.

IGP determines the shortest path to send traffic through MLDP tunnels. However, at times, you may require to choose a path other than the shortest one. For instance, when you want to achieve low latency or want to send the traffic on a specific path to avoid a set of links or build totally two or more disjoint paths.

Multipoint LDP (mLDP) flexible algorithm allows you to customize the IGP path computation based on the business needs.

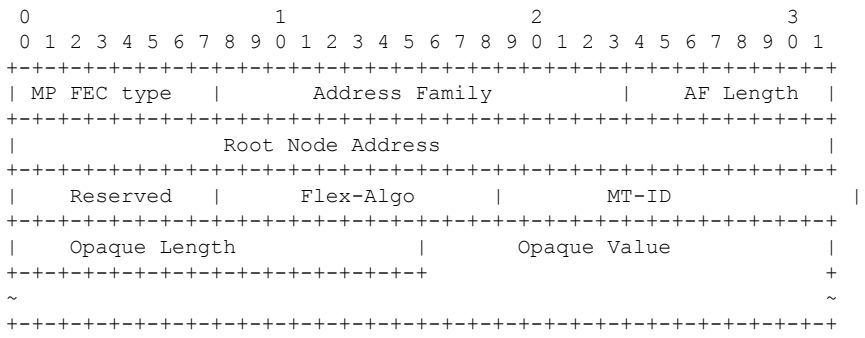
With this feature, you can get disjoint paths for two multicast streams. Each stream carries traffic within a separate network and helps reduce a connection loss or delay ensuring low latency and high availability of multicast traffic. This feature allows you to segregate multicast traffic to specific regions.

mLDP flexible algorithm is based on Segment Routing flexible algorithm that allows operators to customize IGP shortest path computation according to their own needs. For more information, see Enabling Segment Routing Flexible Algorithm.

To compute two different paths, the mLDP flexible algorithm uses a 2-tuple hash algorithm, which includes MPLS Multi-Topology Identifier (MT-ID) and IGP algorithm.

MPLS multi topology Identifier (MT-ID) is a unique identifier that is used to associate an LSP with multi topology. This identifier is part of the mLDP FEC encoding so that LDP peers are able to set up an MP LSP through their own defined policy and avoid conflicting policies for the same mLDP FEC. The MT-ID and IGP Algorithm must be part of the FEC so that different values result in unique MP-LSP FEC elements.

mLDP flexible algorithm is stored in IGP Algorithm (IPA) Registry field. A 16-bit reserved field is included and out of which 8 bits are used for flexible algorithm. The root is an IP address identifying the Root of an MLDP-P2MP tree. Only IPv4 address is supported.



For example, two flexible algorithms are required to implement a disjoint-paths scenario.

Leafs belonging to the first plane are associated with flexible algorithm 130 and leafs belonging another plane are associated with flexible algorithm 128.

mLDP with flexible algorithm also provides the following support:

- Profile 12: Default MDT - MLDP - P2MP - BGP-AD - BGP C-Mcast Signaling
- Profile 14: Partitioned MDT - MLDP P2MP - BGP-AD - BGP C-Mcast Signaling
- MLDP Loop-Free Alternative Fast Reroute
- Data Multicast Distribution Tree (MDT) route policy supports `none` keyword.

Restrictions

The following features are not supported with flexible algorithm:

- PIM BiDir in the core
- Carrier supporting carrier (CSC)
- Inter-AS
- Extranet
- Default MDT
- MVPN profiles except profile-14

How to configure mLDP Flexible Algorithm?

Configure partitioned MDT with flexible algorithm MVPN profile:

```
Router #configure
Router(config)# multicast-routing
Router(config-mcast)# vrf red
Router(config-mcast-red)# address-family ipv4
Router(config-mcast-red-ipv4)# mdt partitioned mldp ipv4 p2mp flex-algo 128
```

Configure data MDT with flexible algorithm:

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

```
Router(config-mcast)# vrf red
Router(config-mcast-red)# address-family ipv4
Router(config-mcast-red-ipv4)# mdt data mldp 10 flex-algo 130
```

Configure route policy for Data MDT with flexible algorithm:

```
Router # configure
Router(config)# multicast-routing vrf red address-family ipv4
Router(config-mcast-red-ipv4)# mdt data mldp 10 route-policy rpl-for-red flex-algo 130
```

Note We recommend having 1:1 allocation of Data MDTs with number of flows. However, you can configure less Data MDTs than the number of flows. But you may not see the expected results. You cannot modify the flexible algorithm configuration without removing the MDT configuration. You must reconfigure MDT with the new flexible algorithm. Other combinations like no flexible algorithm to flexible algorithm are not supported.

Configuration Example

The route-policy and flexible algorithm are configured on Data MDT CLI.

In this example,

- Multicast groups 232.1.1.1 and 232.1.1.2 match the route policy it uses the Data MDT created with flexible algorithm 128.
- Multicast groups 232.1.1.3 and 232.1.1.4 also match the route-policy it uses the Data MDT created with flexible algorithm 129.

  All other groups do not match the route-policy so it uses the Data MDT created with flexible algorithm 130 which is configured in Data MDT CLI.

```
route-policy c1-data-mdt
  if destination in (232.1.1.1, 232.1.1.2) then
    set flex-algo 128
  elseif destination in (232.1.1.3, 232.1.1.4) then
    set flex-algo 129
  endif
end-policy
!
multicast-routing
 vrf red
  address-family ipv4
   mdt partitioned mldp ipv4 p2mp flex-algo 131
   mdt data mldp 100 route-policy c1-data-mdt flex-algo 130
  !
 !
!
```

In the following example, route-policy is configured on Data MDT CLI but without flexible algorithm.

```
route-policy c1-data-mdt
  if destination in (232.1.1.1, 232.1.1.2) then
    set flex-algo 128
  elseif destination in (232.1.1.3, 232.1.1.4) then
    set flex-algo 129
```

```
  endif
end-policy
!
multicast-routing
 vrf red
  address-family ipv4
   mdt partitioned mldp ipv4 p2mp flex-algo 131
   mdt data mldp 100 route-policy c1-data-mdt
  !
 !
!
```

Flexible algorithm is configured on Data MDT CLI and all groups uses the Data MDT created with flexible algorithm 130.

```
multicast-routing
 vrf red
  address-family ipv4
   mdt partitioned mldp ipv4 p2mp flex-algo 131
   mdt data mldp 100 flex-algo 130
```

In the following example, flexible algorithms are configured in Partitioned MDT. All groups use the Data MDT created with Flexible Algorithm 131 which is configured in Partitioned MDT CLI.

```
multicast-routing
 vrf red
  address-family ipv4
   mdt partitioned mldp ipv4 p2mp flex-algo 131
   mdt data mldp 100
```

Verification

```
Router# show mvpn vrf red context private

MVPN context information for VRF red (0xa99dbf8)

RD: 1:1 (Valid, IID 0x1), VPN-ID: 0:0
Import Route-targets : 2
  RT:10.0.0.4:0, BGP-AD
  RT:10.0.0.4:17, BGP-AD
BGP Auto-Discovery Enabled (I-PMSI added) , MS-PMSI sent
MLDP Core-tree data:
  MDT Name: Lmdtred, Handle: 0x8041b0, idb: 0xa9b1c18
  MTU: 1376, MaxAggr: 255, SW_Int: 30, AN_Int: 60
  RPF-ID: 9/0, C:0, O:1, D:0, CP:0
  MLDP Number of Roots: 0 (Local: 0), HLI: 0x00000, Rem HLI: 0x00000
  Discovery Type,Value: Flex-Algo,131
  Data MDT Discovery Type,Value: Flex-Algo,130
  Partitioned MDT: Configured, P2MP (RD:Not added, ID:Added), HLI: 0x00005, Loc Label:
24017, Remote: None
    ID: 2 (0xa79ce90), Ctrl Trees : 0/0/0, Ctrl ID: 0 (0x0), IR Ctrl ID: 0 (0x0), Ctrl HLI:
 0x00000
  P2MP Def MDT ID: 0 (0x0), added: 0, HLI: 0x00000, Cfg: 0/0


Router# show mpls mldp database root 10.0.0.21 opaquetype global-id 5

mLDP database
LSM-ID: 0x00020  Type: P2MP  Uptime: 2d17h
  FEC Root           : 10.0.0.21
  IGP Algorithm      : Flex-Algo 129
   Opaque decoded     : [global-id 5]
```

```
  Upstream neighbor(s) :
      10.0.0.4:0 [Active] Uptime: 1d22h
      Local Label (D) : 24040
  Downstream  client(s):
    PIM MDT            Uptime: 2d17h
      Egress intf     : LmdtWHITE
      Table ID        : IPv4: 0xe0000012 IPv6: 0xe0800012
      RPF ID          : 15
      RD              : 257:18154475
```

```
Router# show pim vrf vpn1 mdt cache
Core Source      Cust (Source, Group)              Core Data        Expires
5.15.15.15       (100.0.1.1, 233.1.1.1)            [global-id 17, Flex-Algo 129]  never

5.15.15.15       (100.0.1.1, 233.1.1.2)            [global-id 18, Flex-Algo 129]  never

5.15.15.15       (100.0.1.1, 233.1.1.3)            [global-id 19, Flex-Algo 129]  never

5.15.15.15       (100.0.1.1, 233.1.1.4)            [global-id 20, Flex-Algo 129]  never

5.15.15.15       (100.0.1.1, 233.1.1.5)            [global-id 21, Flex-Algo 129]  never
```

Label Switched Multicast (LSM) Multicast Label Distribution Protocol (mLDP) based Multicast VPN (mVPN) Support

Label Switch Multicast (LSM) is MPLS technology extensions to support multicast using label encapsulation. Next-generation MVPN is based on Multicast Label Distribution Protocol (mLDP), which can be used to build P2MP and MP2MP LSPs through a MPLS network. These LSPs can be used for transporting both IPv4 and IPv6 multicast packets, either in the global table or VPN context.

Benefits of LSM MLDP based MVPN

LSM provides these benefits when compared to GRE core tunnels that are currently used to transport customer traffic in the core:

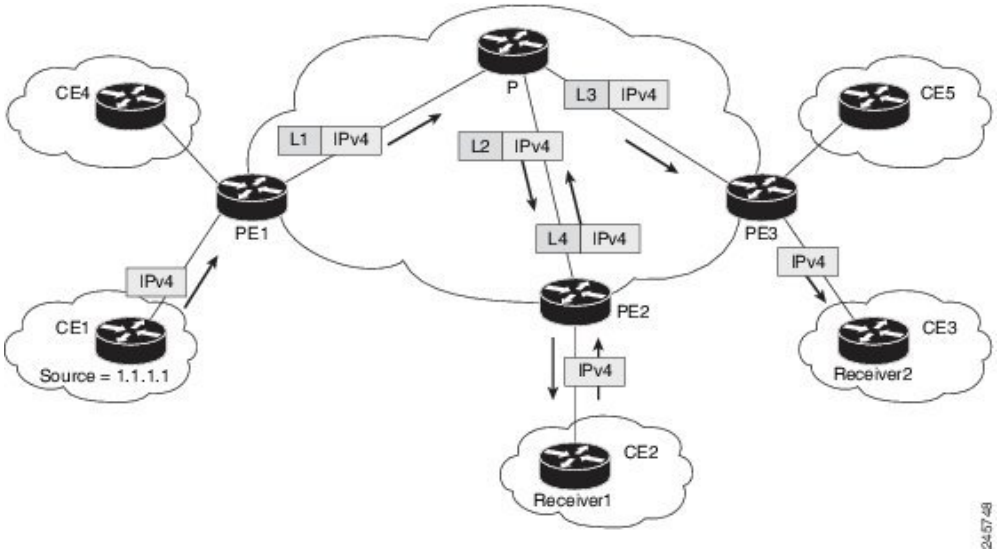
- It leverages the MPLS infrastructure for transporting IP multicast packets, providing a common data plane for unicast and multicast.
- It applies the benefits of MPLS to IP multicast such as Fast ReRoute (FRR) and
- It eliminates the complexity associated PIM.

Configuring MLDP MVPN

The MLDP MVPN configuration enables IPv4 multicast packet delivery using MPLS. This configuration uses MPLS labels to construct default and data Multicast Distribution Trees (MDTs). The MPLS replication is used as a forwarding mechanism in the core network. For MLDP MVPN configuration to work, ensure that the global MPLS MLDP configuration is enabled. To configure MVPN extranet support, configure the source multicast VPN Routing and Forwarding (mVRF) on the receiver Provider Edge (PE) router or configure the receiver mVRF on the source PE. MLDP MVPN is supported for both intranet and extranet.

Note If a tail-end router has multiple VRFs with the same import Route Target (RT), issues can occur. These issues arise when the intended Label Switched Multicast (LSM) is not an extranet setup. The router may perform incorrect signaling. It may also fail to build core trees.

Figure 11: MLDP based MPLS Network



P2MP and MP2MP Label Switched Paths

mLDP is an application that sets up Multipoint Label Switched Paths (MP LSPs) in MPLS networks without requiring multicast routing protocols in the MPLS core. mLDP constructs the P2MP or MP2MP LSPs without interacting with or relying upon any other multicast tree construction protocol. Using LDP extensions for MP LSPs and Unicast IP routing, mLDP can setup MP LSPs. The two types of MP LSPs that can be setup are Point-to-Multipoint (P2MP) and Multipoint-to-Multipoint (MP2MP) type LSPs.

A P2MP LSP allows traffic from a single root (ingress node) to be delivered to a number of leaves (egress nodes), where each P2MP tree is uniquely identified with a 2-tuple (root node address, P2MP LSP identifier). A P2MP LSP consists of a single root node, zero or more transit nodes, and one or more leaf nodes, where typically root and leaf nodes are PEs and transit nodes are P routers. A P2MP LSP setup is receiver-driven and is signaled using mLDP P2MP FEC, where LSP identifier is represented by the MP Opaque Value element. MP Opaque Value carries information that is known to ingress LSRs and Leaf LSRs, but need not be interpreted by transit LSRs. There can be several MP LSPs rooted at a given ingress node, each with its own identifier.

A MP2MP LSP allows traffic from multiple ingress nodes to be delivered to multiple egress nodes, where a MP2MP tree is uniquely identified with a 2-tuple (root node address, MP2MP LSP identifier). For a MP2MP LSP, all egress nodes, except the sending node, receive a packet sent from an ingress node.

A MP2MP LSP is similar to a P2MP LSP, but each leaf node acts as both an ingress and egress node. To build an MP2MP LSP, you can setup a downstream path and an upstream path so that:

- Downstream path is setup just like a normal P2MP LSP
- Upstream path is setup like a P2P LSP towards the upstream router, but inherits the downstream labels from the downstream P2MP LSP.

Packet Flow in mLDP-based Multicast VPN

For each packet coming in, MPLS creates multiple out-labels. Packets from the source network are replicated along the path to the receiver network. The CE1 router sends out the native IP multicast traffic. The Provider Edge1 (PE1) router imposes a label on the incoming multicast packet and replicates the labeled packet towards the MPLS core network. When the packet reaches the core router (P), the packet is replicated with the appropriate labels for the MP2MP default MDT or the P2MP data MDT and transported to all the egress PEs. Once the packet reaches the egress PE , the label is removed and the IP multicast packet is replicated onto the VRF interface.

Realizing a mLDP-based Multicast VPN

There are different ways a Label Switched Path (LSP) built by mLDP can be used depending on the requirement and nature of application such as:

- P2MP LSPs for global table transit Multicast using in-band signaling.
- P2MP/MP2MP LSPs for MVPN based on MI-PMSI or Multidirectional Inclusive Provider Multicast Service Instance (Rosen Draft).
- P2MP/MP2MP LSPs for MVPN based on MS-PMSI or Multidirectional Selective Provider Multicast Service Instance (Partitioned E-LAN).

The router performs the following important functions for the implementation of MLDP:

1. Encapsulating VRF multicast IP packet with GRE/Label and replicating to core interfaces (imposition node).
2. Replicating multicast label packets to different interfaces with different labels (Mid node).
3. Decapsulate and replicate label packets into VRF interfaces (Disposition node).

Note In native multicast and draft-rosen mVPN, load-balancing between bundle members is performed based on outer IP/User Datagram Protocol (UDP) header fields. However, in all MPLS based mVPN design solution, a single tunnel is pinned down to a bundle member.

Characteristics of mLDP Profiles

The characteristics of various mLDP profiles are listed in this section.

Profile 1:Rosen-mLDP (with no BGP-AD)

These are the characteristics of this profile:

- MP2MP mLDP trees are used in the core.
- VPN-ID is used as the VRF distinguisher.
- Configuration based on Default MDTs.
- Same Default-MDT core-tree used for IPv4 and IPv6 traffic.
- Data-MDT announcements sent by PIM (over Default-MDT).

- The multicast traffic can either be SM or SSM.
- Inter-AS Options A, B, and C are supported. Connector Attribute is announced in VPN-IP routes.

Profile 2:MS-PMSI-mLDP-MP2MP (No BGP-AD)

These are the characteristics of this profile:

- MP2MP mLDP trees are used in the core.
- Different MS-PMSI core-trees for IPv4 and IPv6 traffic.
- The multicast traffic can be SM or SSM.
- Extranet, Hub and Spoke are supported.
- Inter-AS Options A, B, and C are supported. Connector Attribute is announced in VPN-IP routes.

Profile 3:Rosen-GRE with BGP-AD

These are the characteristics of this profile:

- PIM-trees are used in the core. The data encapsulation method used is GRE.
- SM or SSM used in the core.
- Configuration is based on Default-MDTs.
- The multicast traffic can be SM or SSM.
- MoFRR in the core is supported.
- Extranet, Hub and Spoke, CsC, Customer-RP-discovery (Embedded-RP, AutoRP and BSR) are supported.
- Inter-AS Options A, and C are supported. VRF-Route-Import EC is announced in VPN-IP routes.

Profile 4: MS-PMSI-mLDP-MP2MP with BGP-AD

These are the characteristics of this profile:

- MP2MP mLDP trees are used in the core.
- The multicast traffic can be SM or SSM.
- Extranet, Hub and Spoke, CsC, Customer-RP-discovery (Embedded-RP, AutoRP, and BSR) are supported.
- Inter-AS Options A, B, and C are supported. VRF-Route-Import EC is announced in VPN-IP routes.

Profile 5: MS-PMSI-mLDP-P2MP with BGP-AD

These are the characteristics of this profile:

- P2MP mLDP trees are used in the core.
- The multicast traffic can be SM or SSM.
- Extranet, Hub and Spoke, CsC, Customer-RP-discovery (Embedded-RP, AutoRP and BSR) are supported.
- Inter-AS Options A, B, and C are supported. VRF-Route-Import EC is announced in VPN-IP routes.

Profile 6: VRF In-band Signaling (No BGP-AD)

These are the characteristics of this profile:

- P2MP mLDP trees are used in the core.
- MoFRR in the core is supported.
- There is one core tree built per VRF-S,G route. There can be no (*,G) routes in VRF, with RPF reachability over the core.
- The multicast traffic can be SM S,G or SSM.
- Inter-AS Options A, B, and C are supported.

Profile 7: Global Inband Signalling

These are the characteristics of this profile:

- P2MP mLDP inband tree in the core; no C-multicast Routing.
- Customer traffic can be SM S,G or SSM.
- Support for global table S,Gs on PEs.
- Inter-AS Options A, B, and C are supported.

For more information on MLDP implementation and OAM concepts, see the Cisco IOS XR MPLS Configuration Guide for the Cisco ASR 9000 Series Router

Profile 8: Global P2MP-TE

These are the characteristics of this profile:

- P2MP-TE tree, with static Destination list, in the core; no C-multicast Routing.
- Static config of (S,G) required on Head-end PE.
- Only C-SSM support on PEs.
- Support for global table S,Gs on PEs.
- Inter-AS Options A is supported.

Profile 9: Rosen-mLDP with BGP-AD

These are the characteristics of this profile:

- Single MP2MP mLDP core-tree as the Default-MDT, with PIM C-multicast Routing.
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RIB-Extranet, RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.

- Inter-AS Options A, B, and C are supported.

Profile 10 : VRF Static-P2MP-TE with BGP AD

These are the characteristics of this profile:

- P2MP-TE tree, with static Destination list, in the core; no C-multicast Routing.
- Static config of (S,G) required on Head-end PE.
- Only C-SSM support on PEs.
- Support for IPv4 MVPN S,Gs on PEs. No support for IPv6 MVPN routes.
- Inter-AS Options A is supported.

Profile 11 : Rosen-PIM/GRE with BGP C-multicast Routing

These are the characteristics of this profile:

- PIM-trees in the core, data encapsulation is GRE, BGP C-multicast Routing.
- Static config of (S,G) required on Head-end PE.
- For PIM-SSM core-tree and PIM-SM core-tree with no spt-infinity, all UMH options are supported.
- For PIM-SM core-tree with spt-infinity case, only SFS (Highest PE or Hash-of-BGP-paths) is supported. Hash of installed-paths method is not supported.
- Default and Data MDTs supported.
- Customer traffic can be SM or SSM .
- Inter-AS Options A, and C are supported. Options B is not supported.
- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Profile 13 : Rosen-mLDP-MP2MP with BGP C-multicast Routing

These are the characteristics of this profile:

- Single MP2MP mLDP core-tree as the Default-MDT, with BGP C-multicast Routing.
- Only SFS (Highest PE or Hash-of-BGP-paths) is supported. Hash of Installed-paths method is not supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RIB-Tail-end-Extranet, RPL-Tail-end-Extranet supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A, B and C supported. For Options B and C, Root has to be on a PE or the roor-address reachability has to be leaked across all autonomous systems.

- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Profile 15 : MP2MP-mLDP-MP2MP with BGP C-multicast Routing

These are the characteristics of this profile:

- Full mesh of MP2MP mLDP core-tree as the Default-MDT, with BGP C-multicast Routing.
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RPL-Tail-end-Extranet supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A, B and C supported.
- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Profile 16 : Rosen-Static-P2MP-TE with BGP C-multicast Routing

These are the characteristics of this profile:

- Full mesh of Static-P2MP-TE core-trees, as the Default-MDT, with BGP C-multicast Routing.
- All UMH options supported.
- Support for Data MDT, Default MDT.
- Customer traffic can be SM, SSM .
- RPL-Tail-end-Extranet supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.
- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Note Whenever multicast stream crosses configured threshold on encap PE(Head PE), S-PMSI is announced. Core tunnel is static P2MP-TE tunnel configured under route-policy for the stream. Static P2MP-TE data mdt is implemented in such a way that it can work with dynamic data mdt, dynamic default mdtand default static P2MP.

Profile 17: Rosen-mLDP-P2MP with BGP AD/PIM C-multicast Routing

These are the characteristics of this profile:

- Full mesh of P2MP mLDP core-tree as the Default-MDT, with PIM C-multicast Routing.

- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A, B and C supported.

Profile 18 : Rosen-Static-P2MP-TE with BGP AD/PIM C-multicast Routing

These are the characteristics of this profile:

- Full mesh of Static-P2MP-TE core-trees, as the Default-MDT, with PIM C-multicast Routing.
- All UMH options supported.
- Default MDT supported; Data MDT is not supported.
- Customer traffic can be SM, SSM .
- RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.

Profile 19: Rosen-mLDP-P2MP with BGP AD/PIM C-multicast Routing

These are the characteristics of this profile:

- IR tunnels as the Core-tree protocol.
- Only SFS (Highest PE or Hash-of-BGP-paths) is supported. Hash of Installed-paths method is not supported.
- Default and Data MDT supported.
- Customer traffic can be SM, SSM .
- RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.

Profile 20 : Rosen-P2MP-TE with BGP AD/PIM C-multicast Routing

These are the characteristics of this profile:

- Dynamic P2MP-TE tunnels setup on demand, with PIM C-multicast Routing
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM, SSM .

- RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.

Profile 22 : Rosen-P2MP-TE with BGP C-multicast Routing

These are the characteristics of this profile:

- Dynamic P2MP-TE tunnels with BGP C-multicast Routing
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RIB-Tail-end-Extranet, RPL-Tail-end-Extranet supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.
- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Profile 24: Partitioned-P2MP-TE with BGP AD/PIM C-multicast Routing

These are the characteristics of this profile:

- Dynamic P2MP-TE tunnels setup on demand, with PIM C-multicast Routing
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM or SSM .
- RPL-Extranet, Hub & Spoke supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.
- Inter-AS Option A supported. Options B and C not supported.

Profile 26 : Partitioned-P2MP-TE with BGP C-multicast Routing

These are the characteristics of this profile:

- Dynamic P2MP-TE tunnels with BGP C-multicast Routing
- All UMH options supported.
- Default and Data MDT supported.
- Customer traffic can be SM, SSM.
- RIB-Tail-end-Extranet, RPL-Tail-end-Extranet supported.
- Customer-RP-discovery (Embedded-RP, AutoRP & BSR) is supported.

- Inter-AS Option A supported. Options B and C not supported.
- All PEs must have a unique BGP Route Distinguisher (RD) value. To configure BGP RD value, refer *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

Configuration rules for profiles

Rules for Rosen-mGRE profiles (profiles- 0, 3, 11)

- All profiles require VPNv4 or v6 unicast reachability.
- By default, encap 1400-byte size c-multicast IP packet is supported. To support decap or encap larger packet size, **mdt mtu** command.
- Loopback configuration is required. Use the **mdt source loopback0** command. Other loopbacks can be used for different VRFs, but this is not recommended.

Rules for Rosen-mLDP profiles (profiles- 1, 9, 12, 13, 17)

- mLDP must be globally enabled.
- VPN-id is mandatory for Rosen-mLDP MP2MP profiles.
- Root node must be specified manually. Multiple root nodes can be configured for Root Node Redundancy.
- If only profile 1 is configured, MVPN must be enabled under bgp.
- For BGP-AD profiles, the remote PE address is required.

Rules for mLDP profiles (profiles- 2, 4, 5, 14, 15)

- MVPN must be enabled under bgp, if only profile 2 is configured.
- Support only for static RP for customer RP.

Rules for inband mLDP profiles (profiles- 6, 7)

- MVPN must be enabled under bgp for vrf-inband profiles.
- Data MDT is not supported.
- Backbone facing interface (BFI) must be enabled on tail PE.
- Source route of SSM must be advertise to tail PE by iBGP.

MLDP inband signaling

MLDP Inband signaling allows the core to create (S,G) or (*,G) state without using out-of-band signaling such as BGP or PIM. It is supported in VRF (and in the global context). Both IPv4 and IPv6 multicast groups are supported.

In MLDP Inband signaling, one can configure an ACL range of multicast (S,G). This (S,G) can be transported in MLDP LSP. Each multicast channel (S,G), is 1 to 1 mapped to each tree in the inband tree. The (S,G) join, through IGMP/MLD/PIM, will be registered in MRIB, which is the client of MLDP.

MLDP In-band signalling supports transiting PIM (S,G) or (*,G) trees across a MPLS core without the need for an out-of-band protocol. In-band signaling is only supported for shared-tree-only forwarding (also known as sparse-mode threshold infinity). PIM Sparse-mode behavior is not supported (switching from (*,G) to (S,G).

The details of the MLDP profiles are discussed in the *Multicast Configuration Guide for Cisco ASR 9000 Series Routers*

Summary of Supported MVPN Profiles

This tables summarizes the supported MVPN profiles:

Profile Number	Name	Opaque-value	BGP-AD	Data-MDT
0	Rosen GRE	N/A	N/A	PIM TLVs over default MDT
1	Rosen MLDP	Type 2 - Root Address:VPN-ID:0-n	N/A	PIM TLVs over default MDT
2	MS- PMSI (Partition) MLDP MP2MP	Cisco proprietary - Source- PE:RD:0	N/A	N/A
3	Rosen GRE with BGP -AD	N/A	• Intra-AS MI- PMSI • S- PMSI for Data-MDT	PIM or BGP -AD (knob controlled)
4	MS- PMSI (Partition) MLDP MP2MP with BGP -AD	Type 1 - Source-PE:Global -ID	• I- PMSI with empty PTA • MS- PMSI for partition mdt • S- PMSI for data-mdt • S- PMSI cust RP-discovery trees	BGP-AD
5	MS- PMSI (Partition) MLDP P2MP with BGP -AD	Type 1 - Source-PE:Global -ID	• I- PMSI with empty PTA • MS- PMSI for partition mdt • S- PMSI for data-mdt • S- PMSI cust RP-discovery trees	BGP-AD
6	VRF Inband MLDP	RD:S,G	N/A	N/A
7	Global Inband	S,G	N/A	N/A
8	Global P2MP TE	N/A	N/A	N/A
9	Rosen MLDP with BGP -AD	Type 2 - RootAddresss:VPN - ID:0 -n	• Intra-AS MI- PMSI • S- PMSI for Data-MDT	PIM or BGP-AD (knob controlled)

LSP-switch for P2MP-TE

Turnaround for P2MP-TE can be handled by LSP-switch with a partitioned profile. For partitioned profiles, there is no core tree (where all the PEs join). When the traffic arrives at the ingress PE, it is forwarded to the RP-PE on a LSP. The RP-PE must then switch the traffic to a different LSP for all the non-RP PE receivers.

Configuration Process for MLDP MVPN (Intranet)

These steps provide a broad outline of the different configuration process of MLDP MVPN for intranet:

Note For detailed summary of the various MVPN profiles, see the *Summary of Supported MVPN Profiles.*

- Enabling MPLS MLDP
  - configure
  - mpls ldp mldp
- Configuring a VRF entry
  - configure
  - vrf vrf_name
  - address-family ipv4/ipv6 unicast
  - import route-target route-target-ext-community
  - export route-target route-target-ext-community
- Configuring VPN ID
  - configure
  - vrf *vrf_name*
  - vpn id *vpn_id*
- Configuring MVPN Routing and Forwarding instance
  - configure
  - multicast-routing vrf *vrf_name*
  - address-family ipv4
  - mdt default mldp ipv4 *root-node*
- Configuring the Route Distinguisher
  - configure
  - router bgp *AS Number*
  - vrf *vrf_name*
  - rd *rd_value*

- Configuring Data MDTs (optional)
  - configure
  - multicast-routing vrf *vrf_name*
  - address-family ipv4
  - mdt data <1-255>
- Configuring BGP MDT address family
  - configure
  - router bgp *AS Number*
  - address-family ipv4 mdt
- Configuring BGP vpnv4 address family
  - configure
  - router bgp *AS Number*
  - address-family vpnv4 unicast
- Configuring BGP IPv4 VRF address family
  - configure
  - router bgp *AS Number*
  - vrf *vrf_name*
  - address-family ipv4 unicast
- Configuring PIM SM/SSM Mode for the VRFs
  - configure
  - router pim
  - vrf *vrf_name*
  - address-family ipv4
  - rpf topology route-policy *rosen_mvpn_mldp*

  For each profile, a different route-policy is configured.
- Configuring route-policy
  - route-policy *rosen_mvpn_mldp*
  - set core-tree *tree-type*
  - pass
  - end-policy

Note The configuration of the above procedures depends on the profile used for each configuration.

MLDP Loop-Free Alternative Fast Reroute

Generally, in a network, a network topology change, caused by a failure in a network, results in a loss of connectivity until the control plane convergence is complete. There can be various levels of loss of connectivity depending on the performance of the control plane, fast convergence tuning, and leveraged technologies of the control plane on each node in the network.

The amount of loss of connectivity impacts some loss-sensitive applications, which have severe fault tolerance (typically of the order of hundreds of milliseconds and up to a few seconds). To ensure that the loss of connectivity conforms to such applications, a technology implementation for data plane convergence is essential. **Fast Reroute (FRR)** is one of such technologies that is primarily applicable to the network core.

With the FRR solution, at each node, the backup path is precomputed, and the traffic is routed through this backup path. As a result, the reaction to failure is local; immediate propagation of the failure and subsequent processing on to other nodes is not required. With FRR, if the failure is detected quickly, a loss of connectivity as low as 10s of milliseconds is achieved.

Loop-Free Alternative Fast Reroute

IP Loop Free Alternative FRR is a mechanism that enables a router to rapidly switch traffic to a pre-computed or a pre-programmed **loop-free alternative (LFA)** path (Data Plane Convergence), following either an adjacent link and node failure, or an adjacent link or node failure in both IP and LDP networks. The LFA path is used to switch traffic till the router installs the new primary next-hops based on the changed network topology (Control Plane Convergence).

The goal of LFA FRR is to reduce the loss of connectivity to tens of milliseconds by using a pre-computed alternative next-hop, in the case where the selected primary next-hop fails.

There are two approaches to computing LFA paths:

- **Link-based (per-link):** In link-based LFA paths, all prefixes reachable through the primary (protected) link share the same backup information. This means that the whole set of prefixes sharing the same primary also shares the repair and FRR ability.
- **Prefix-based (per-prefix):** Prefix-based LFAs allow computing backup information for each prefix. This means that the repair and backup information computed for a given prefix using prefix-based LFA may be different from the one computed by link-based LFA.

  Node-protection support is available with per-prefix LFA FRR on ISIS currently. It uses a tie-breaker mechanism in the code to select node-protecting backup paths.

The per-prefix LFA approach is preferred to the per-link LFA approach for the following reasons:

- Better node failure resistance.
- Better coverage: Each prefix is analyzed independently.
- Better capacity planning: Each flow is backed up on its own optimized shortest path.

MLDP LFA FRR

The point-to-point physical or bundle interface FRR mechanism is supported on MLDP. FRR with LFA backup is also supported on MLDP. When there is a link failure, MLDP automatically sets up and chooses the backup path. With this implementation, you must configure the physical or bundle interface for unicast traffic, so that the MLDP can act as an MLDP FRR.

LFA FRR support on MLDP is a per-prefix backup mechanism. As part of computing the LFA backup for a remote IP, the LFA backup paths for the loopback address of the downstream intermediate nodes are also computed. MLDP uses this small subset of information, by using the loopback address of the peer to compute the LFA backup path.

Note Both IPv4 and IPv6 traffic is supported on the MLDP LFA FRR solution.

For information on use cases, see the MLDP with Flex-Algo in Service Provider Networks White Paper.

MLDP LFA FRR with Flexible Algorithm

The MLDP LFA FRR with Flexible Algorithm uses the segment routed (SR) LFA FRR-selected primary and backup paths to the peers and emulates a multicast distribution tree, instead of multicast label-switched paths (LSP). It helps in having a more efficient FRR with low-latency routing, live-live disjoint paths, or constraining multicast flows to a specific region. Interior Gateway Protocol (IGP) calculates LFA path for each learned node SID within the IGP domain.

Note All the following limitations of MLDP LFR FRR without Flexible Algorithm also apply to MLDP LFA FRR with Flexible Algorithm:

- Node protection is not supported.

Supported MLDP Profiles

The following MLDP profile is supported:

- Profile 14: Partitioned MDT - MLDP P2MP - BGP-AD - BGP C-Mcast Signaling
- MVPN-MLDP Inband Signaling
  - Global Inband Profile
  - VRF Inband Profile
- MVPN Rosen MLDP

Supported Line Cards And Interfaces

The supported line cards include Cisco ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 High Density 100GE Ethernet line cards; and the supported interface types include: Physical interface, Bundle interface, and the Bundle VLANs (Local Shut).

Advantages of LFA FRR

The following are the advantages of the LFA FRR solution:

- The backup path for the traffic flow is pre-computed.
- Reaction to failure is local, an immediate propagation and processing of failure on to other nodes is not required.
- If the failure is detected in time, the loss of connectivity of up to 10s of milliseconds can be achieved. Prefix independency is the key for a fast switchover in the forwarding table.
- The mechanism is locally significant and does not impact the Interior Gateway Protocol (IGP) communication channel.
- LFA next-hop can protect against:
  - a single link failure
  - failure of one of more links within a shared risk link group (SRLG)
  - any combination of the above

MLDP LFA FRR - Features

The following are the features of mLDP LFA FRR solution:

- Supports both IPv4 and IPv6 traffic
- Supports all the mLDP profiles
- Supports the LAG interfaces and sub-interfaces in the core
- Supports ECMP primary paths
- Supports both ISIS and OSPF routing protocols
- Supports switchover time of less than 50 milliseconds
- Supports switchover time to be independent of the number of multicast routes that has to be switched over

Limitations of LFA FRR

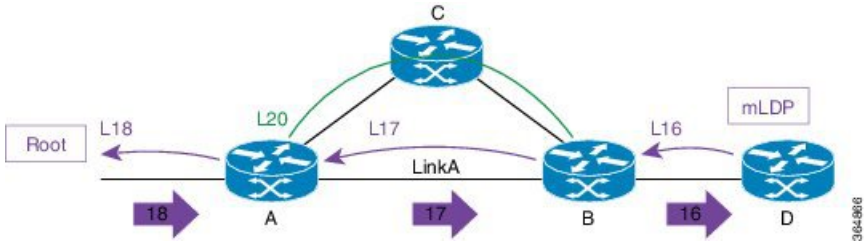
The following are some of the known limitations of the LFA FRR solution:

- When a failure that is more extensive than that which the alternate was intended to protect occurs, there is the possibility of temporarily looping traffic (micro looping until Control Plane Convergence).
- Topology dependent. For example, either MPLS or MLDP dependent.
- Complex implementation.
- The solution is currently not supported on all platforms.
- mLDP FRR with BGP labeled unicast (BGP-LU) is not supported as an underlay. The **forwarding recursive** configuration should not be enabled on PE routers when using MVPN with BGP-LU.

MLDP LFA FRR - Working

To enable FRR for mLDP over physical or bundle interfaces, LDP session-protection has to be configured. The sequence of events that occur in an mLDP LFA FRR scenario is explained with the following example:

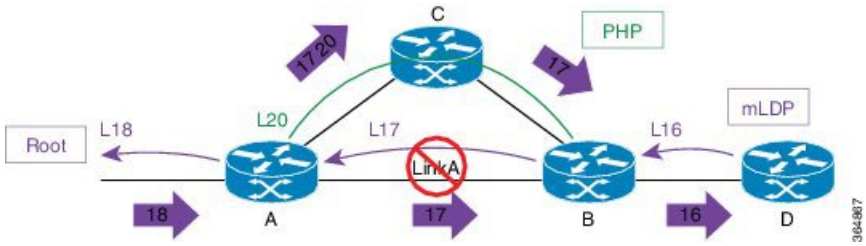
Figure 12: MLDP LFA FRR - Setup



In this figure:

1. Router A is the source provider edge router, and the next Hop is Router B.
2. The primary path is Router A -> Router B - > Router D, and the backup path is from Router A -> Router C -> Router B -> Router D. The backup path is pre-computed by IGP through LFA prefix-based selection.
3. Backup tunnels are configured for Link A or auto-tunnels are enabled.
4. MLDP LSP is build from D, B, and A towards the root.
5. Router A installs a downstream forwarding replication over link A to Router B. This entry has both the primary interface (Link A) and the backup tunnel programmed.

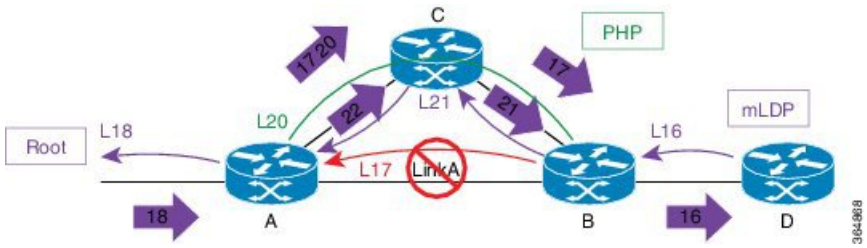
Figure 13: Link Failure



When a ink failure occurs on Link A:

1. Traffic over Link A is rerouted over the backup tunnel by imposing the traffic engineering (TE) label 20 towards mid Router C.
2. Router C performs penultimate hop popping (PHP) and removes the outer label 20.
3. Router B receives the mLDP packets with label 17 and forwards to Router D.

Figure 14: Re-optimization - Make-Before-Break



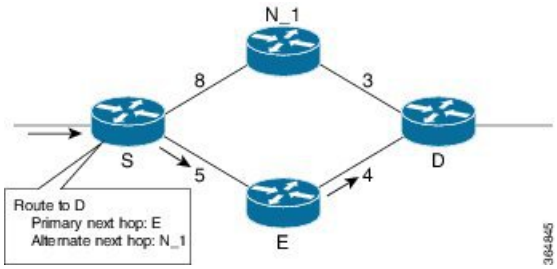
During re-optimization:

1. mLDP is notified that the root is reachable through Router C, and mLDP converges. With this, a new mLDP path is built to router A through Router C.
2. Router A forwards packets natively with old label 17 and also new label 22.
3. Router B drops traffic carried from new label 22 and forwards traffic with label 17.
4. Router B uses make-before-break (MBB) trigger to switch from either physical or bundle interface to native, label 17 to 21.
5. Router B prunes off the physical or bundle interface with a label withdraw to router A.

MLDP LFA FRR - Behavior

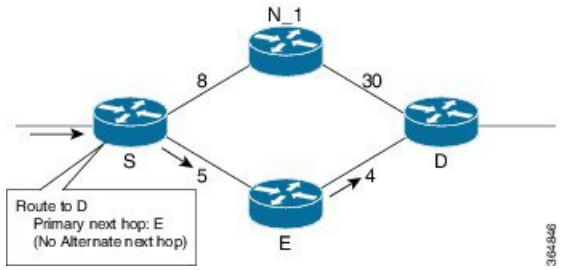
In the following scenarios, S is source router, D is the destination router, E is primary next hop, and N_1 is the alternative next hop.

Figure 15: LFA FRR Behavior - LFA Available



With LFA FRR, the source router S calculates an alternative next hop N_1 to forward traffic towards the destination router D through N_1, and installs N_1 as a the alternative next hop. On detecting the link failure between routers S and E, router S stops forwarding traffic destined for router D towards E through the failed link; instead it forwards the traffic to a pre-computed alternate next hop N_1, until a new SPF is run and the results are installed.

Figure 16: LFA FRR Behavior - LFA Not Available



In the above scenario, if the link cost between the next hop N_1 and the destination router D is increased to 30, then the next hop N_1 would no longer be a loop-free alternative. (The cost of the path, from the next hop N_1 to the destination D through the source S, would be 17, while the cost from the next hop N_1 directly to destination D would be 30). Thus, the existence of a LFA next hop is dependent on the topology and the nature of the failure, for which the alternative is calculated.

LFA Criteria

In the above example, the LFA criteria of whether N is to be the LFA next-hop is met, when:

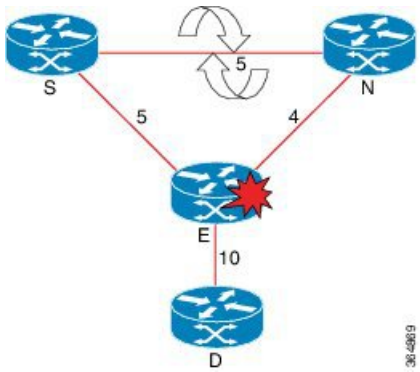
Cost of path (N_1, D) < Cost of path (N_1, S) + Cost of path (E, S) + Cost of path (D, E)

Downstream Path criteria, which is subset of LFA, is met when:

Cost of path (N_1, D) < Cost of path (E, S) + Cost of path (D, E)

Link Protecting LFA

Figure 17: Link Protecting LFA



In the above illustration, if router E fails, then both router S and router N detects a failure and switch to their alternates, causing a forwarding loop between both routers S and N. Thus, the Link Protecting LFA causes Loop on Node Failure; however, this can be avoided by using a down-stream path, which can limit the coverage of alternates. Router S will be able to use router N as a downstream alternate, however, router N cannot use S. Therefore, N would have no alternate and would discard the traffic, thus avoiding the micro-looping.

Node Protecting LFA

Link and node protecting LFA guarantees protection against either link or node failure. Depending on the protection available at the downstream node, the downstream path provides protection against a link failure; however, it does not provide protection against a node failure, thereby preventing micro looping.

The criteria for LFA selection priority is that: the Link and Node protecting LFA is greater than the Link Protecting Downstream is greater than the Link Protecting LFA.

Configure MLDP Route-Policy for Flexible Algorithm FRR

Using the MLDP route-policy option, you can enable the FRR for selected LSPs. You can enable FRR only for flexible algorithm-based LSPs, non-flexible algorithm-based LSPs, or for both. This route policy helps when you have a large number of flows and you have enabled FRR on all of them. If you have some flows that are critical and need FRR and other flows without FRR, you can apply the customization using the route policy.

If you do not configure any route policy, then the FRR is enabled on all LSPs.

The following example shows how to configure MLDP route-policy with flexible algorithm and apply the same:

```
Router#config
Router(config)#route-policy mldp-fa-frr
Router(config-rpl)#if mldp flex-algo?
<128-255>
Algorithm number
any
Any Algorithm
Router (config-rpi)#if mldp flex-algo 128 then
Router(config-rpl-if)#pass
Router(config-rpl-if)Hendif
Router(config-rpl)#if mldp flex-algo any then
Router(config-rpl-if)#pass
Router(config-rpl-if)Hendif
Router(config-rpl)Hend-policy

Router#config
Router(config)#mpls Idp mldp address-family ipv4
Router(config-ldp-mldp-af)#forwarding recursive route-policy mldp-fa-frr
Router(config-ldp-mldp-af)#
```

Configurations to Enable LFA FRR

Key Configurations To Enable LFA FRR

The key configurations to enable LFA FRR feature include:

- Router OSPF configuration

  The various configurations available under OSPF are:

  - Enabling Per-Prefix LFA
  - Excluding Interface from Using Backup
  - Adding Interfaces to LFA Candidate List
  - Restricting LFA Candidate List
  - Limiting Per-Prefix Calculation by Prefix Priority
  - Disabling Load Sharing of Backup Paths

- Router ISIS configuration
- Bidirectional Forwarding Detection (BFD) configuration
- MPLS configuration

  The various configurations available under MPLS are:

- MBB (MLDP) configuration
- Make Before Break (MBB) Delay X <sec> Delete Y <sec>
- Configure FRR Timer for Scale Number of MLDP LSPs

Configuring Router OSPF LFA FRR

In OSPF configuration, configure per-prefix link based LFA to enable the LFA FRR feature. The detailed configuration steps with an example follows:

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix
```

Enables the per-prefix mode of LFA calculation on the specified interface.

Step 6 **commit**

Example

Example: Configuration to Enable OSPF LFA FRR

```
!
router ospf {tag}
area {area-id}
interface {interface}
fast-reroute per-prefix enable
!
!
```

Enabling Per Prefix LFA

Lists the steps required to enable per-prefix LFA mode of LFA calculation using OSPF configuration.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area sub mode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface sub mode configuration, under OSPF area sub mode.

Step 5 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix
```

Enables the per-prefix mode of LFA backup path calculation on the specified interface.

Step 6 **commit**

Adding Interfaces to LFA Candidate List

Lists the steps required to add an interface to the LFA candidate list.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix lfa-candidate**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix lfa-candidate Bundle-Ether10
```

Adds the listed interface to the LFA candidate list to compute backup paths.

Note
By default, no interfaces are on the LFA candidate list.

Step 6 **commit**

Exclude Interface from Backup

Lists the steps required to exclude an interface from using backup paths for LFA calculation using OSPF configuration.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix exclude**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix exclude Bundle-Ether10
```

Excludes the specific listed interface while calculating the LFA backup paths.

Note
By default, no interfaces are excluded from the LFA backup path calculation.

Step 6 **commit**

Restricting the Backup Interfaces to the LFA Candidate List

Lists the steps required to restrict the backup interface to the LFA candidate list.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix use-candidate-only**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix use-candidate-only
```

Restricts the calculation of the backup paths to only the interfaces listed on the LFA candidate list.

Note
By default, the **fast-reroute per-prefix use-candidate-only** is disabled.

Step 6 **commit**

Limiting the Per-Prefix Calculation by Prefix-Priority

Lists the steps required to limit the per-prefix calculation by prefix-priority.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix prefix-limit {priority}**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix prefix-limit
{priority}
```

Limits the per-prefix LFA backup path calculation by prefix-priority.

Only prefixes with the same or higher priority as specified are subjected to the per-prefix backup paths calculation.

Note
By default, backup path is calculated for prefixes regardless of their priority.

Step 6 **commit**

Disabling Load Sharing of the Backup Paths

Lists the steps required to disable the load sharing of the backup paths.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF configuration mode.

Step 3 **area 0**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 4 **interface Bundle-Ether10**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether10
```

Enters the interface submode configuration, under OSPF area submode.

Step 5 **fast-reroute per-prefix load-sharing disable**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix load-sharing disable
```

Disables the load sharing of the backup paths.

It is used to control the load-balancing of the backup paths on a per-prefix basis.

Note
By default, load-balancing of per-prefixes across all backup paths is enabled.

Step 6 **commit**

Configuring Router ISIS LFA FRR

In ISIS configuration, configure fast-reroute per-prefix to enable the LFA FRR feature.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **router isis** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router isis MCAST
```

Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.

Step 3 **net** *network-entity-title*

Example:

```
RP/0/RSP0/CPU0:router(config-isis)# net 49.0001.0000.0000.0001.00
```

Configures network entity titles (NETs) for the routing instance.

- Specify a NET for each routing instance if you are configuring multi-instance IS-IS.
- This example, configures a router with area ID 49.0001.0000.0000 and system ID 0000.0001.0000.0000
- To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs for all of the configured items, the system ID portion of the NET must match exactly.

Step 4 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-isis)# nsr
```

Enables nonstop routing.

Step 5 **nsf** *cisco*

Example:

```
RP/0/RSP0/CPU0:router(config-isis)# nsr cisco
```

Specifies that nonstop forwarding.

Step 6 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast
```

Enters the address-family submode. This is supported only on unicast topologies.

Step 7 **commit**

Step 8 **interface***GigabitEthernet0/0/1/1*

Example:

```
RP/0/RSP0/CPU0:router(config-isis-af)# interface GigabitEthernet0/0/1/1
```

Enters the interface submode.

Step 9 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-isis-if-af)# address-family ipv4 unicast
```

Enters the address-family submode. This is supported on unicast topologies only.

Step 10 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
```

Enables LFA FRR.

Step 11 **interface***GigabitEthernet0/0/1/7*

Example:

```
RP/0/RSP0/CPU0:router(config-isis-af)# interface GigabitEthernet0/0/1/7
```

Enters the interface submode.

Step 12 **commit**

Configuring Bidirectional Forwarding Detection

When a local interface is down, that is, due to either a fiber cut or because of interface shutdown configuration is run, it can take a long delay in the order of tens of milliseconds for the remote peer to detect the link disconnection; so, to quickly detect the remote shut on physical port or on bundle interfaces, the physical port and bundle interfaces must be running Bidirectional Forwarding Detection (BFD) to ensure faster failure detection.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config)#router ospf 0
```

Enters the OSPF routing configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# nsr
```

Enables nonstop routing.

Step 4 **router-id** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# router-id 21.21.21.21
```

Specifies the router ID of the particular IPv4 address.

Step 5 **nsf** *instance name*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# interface cisco
```

Enters the interface submode configuration, under OSPF mode.

Step 6 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# address-family ipv4 unicast
```

Enters the address-family submode. This is supported only on unicast topologies.

Step 7 **area** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# area 0
```

Enters the area submode under the OSPF configuration mode.

Step 8 **bfd minimum-interval** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# bfd minimum-interval 3
```

Sets the bidirectional forwarding detection minimum-interval value to 3.

Step 9 **bfd fast-detect**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# bfd fast-detect
```

Configures bidirectional forwarding detection to fast detection.

Step 10 **bfd multiplier** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# bfd multiplier 2
```

Configures bidirectional forwarding detection to fast detection.

Step 11 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# fast-reroute per-prefix
```

Enables the per-prefix mode of LFA calculation on the specified interface.

Step 12 **mpls traffic-eng**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# mpls traffic-eng
```

Configures the MPLS TE under the OSPF area.

Step 13 **interface** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# interface Bundle-Ether100.1
```

Configures the specified interface.

Step 14 **bfd fast-detect**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af-if)# bfd fast-detect
```

Configures bidirectional forwarding detection to fast detection.

Step 15 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af-if)# fast-reroute per-prefix
```

Enables the per-prefix mode of LFA calculation on the specified interface.

Step 16 **commit**

Step 17 **interface** *instance id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af)# interface Bundle-Ether100.1
```

Configures the specified interface.

Step 18 **bfd fast-detect**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af-if)# bfd fast-detect
```

Configures bidirectional forwarding detection to fast detection.

Step 19 **fast-reroute per-prefix**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-af-if)# fast-reroute per-prefix
```

Enables the per-prefix mode of LFA calculation on the specified interface.

Step 20 **commit**

Step 21 **interface loopback0**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface loopback0
```

Example

```
router ospf 0
 nsr
 router-id 21.21.21.21
 nsf cisco
 address-family ipv4 unicast
 area 0
  bfd minimum-interval 3
  bfd fast-detect
  bfd multiplier 2
  fast-reroute per-prefix
  mpls traffic-eng
  interface Bundle-Ether100.1
    bfd fast-detect
    fast-reroute per-prefix
!
  interface Bundle-Ether100.2
    bfd fast-detect
    fast-reroute per-prefix
!
interface Loopback0
!
```

In the above configuration example, **bfd minimum-interval 3** and **bfd multiplier 2** is configured; this means, that when a core-facing interface of a remote peer is down, the router detects this disconnect event in as short a time as 6 milliseconds.

Configuring MPLS LFA FRR

Before you begin

In MPLS configuration, configure session protection to support LFA FRR feature. The detailed configuration steps and an example follows.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **router ospf 0**

Example:

```
RP/0/RSP0/CPU0:router(config)#mpls ldp
```

Enters the LDP configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# nsr
```

Configures non-stop routing.

Step 4 **graceful-restart**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# graceful-restart
```

Restarts the interface.

Step 5 **router-id 20.20.20.20**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# router-id 20.20.20.20
```

Configures a router-id for the LDP process.

Step 6 **session protection**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# session protection
```

Enables LFA FRR in the per-prefix mode.

Step 7 **address-family ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# address-family ipv4
```

Enters address family configuration mode.

Step 8 **commit**

Example

Example: Configuration to enable MLDP LFA FRR

```
mpls ldp
 nsr
 graceful-restart
 !
router-id 20.20.20.20
session protection
address-family ipv4
!
!
```

Make Before Break Configuration for LFA FRR

Make Before Break (MBB) is an inherent nature of MLDP. In MBB configuration, configure forwarding recursive to enable LFA FRR feature. If forwarding recursive is not configured, MLDP uses non-recursive method to select MLDP core facing interface towards next hop. The detailed configuration steps and an example follows.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp`	Enters the LDP configuration mode.
Step 3	**log** **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# log`	Enters the log sub mode under the LDP sub mode.
Step 4	**neighbor** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-log)# neighbor`	Configures the specified neighbor to the MLDP policy.
Step 5	**nsr** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-log)# nsr`	Configures non-stop routing.
Step 6	**graceful-restart** **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# graceful-restart`	Restarts the interface.
Step 7	**commit**	
Step 8	**mldp** **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# mldp`	Enters the MLDP sub mode under the LDP sub mode.
Step 9	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# address-family ipv4`	Enters the Address Family sub mode under the MLDP sub mode.

	Command or Action	Purpose
Step 10	**forwarding recursive** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# forwarding recursive`	Enables LFA FRR.
Step 11	**make-before-break delay {seconds}** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# make-before-break delay 60`	Sets the make-before-break delay to the specified number of seconds.
Step 12	**commit**	

Example

Example Configuration Example of MBB for LFA FRR

```
mpls ldp
log
neighbor
nsr
graceful-restart
!
mldp
address-family ipv4
forwarding recursive
make-before-break delay 60
!
!
```

Configuring Make Before Break Delay and Delete

By default, MBB is set to 10 seconds. You can configure different MBB timing to determine when the merge node starts to accept the new label.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp`	Enters the LDP configuration mode.
Step 3	**mldp** **Example:**	Enters the MLDP sub mode under the LDP sub mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ldp)# mldp`	
Step 4	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# address-family ipv4`	Enters the Address Family sub mode under the MLDP sub mode.
Step 5	**make-before-break delay {seconds}** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# make-before-break delay 90`	Sets the Make Before Break delay to 90 seconds.
Step 6	**make-before-break delay {seconds} delete {seconds}** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# make-before-break delay 90 delete 60`	Sets the Make Before Break delete delay to 60 seconds.
Step 7	**commit**	

Example

Example: Make Before Break Delay And Delete

```
mldp
address-family ipv4
make-before-break delay ?
<0-600> Forwarding delay in seconds
make-before-break delay 90 ?
<0-60> Delete delay in seconds
make-before-break delay 90 delete 60
!
!
```

In the above configuration example, the MBB (delay) period is set of 90 seconds. The merge node starts accepting new label 90 seconds after detecting the link disconnection towards the head node. The delete delay is set to 60 seconds; that is, when MBB expires, the time period after which the merge node sends old label delete request to head node is 60 seconds. The default value is zero. The range of delete delay is from 30 to 60, for scale LSPs.

Configuring FRR Time for Scalable Number of mLDP LSPs

In a scalable setup with more than 500 LSPs, when an FRR occurs, the unicast Internet Gateway Protocol (IGP) converges faster than multicast updates (LMRIB to FIB) for MLDP label updates. As a result, FIB can mark off FRR bit in 2 seconds after an FRR event, where MLDP label hardware programing is not complete in the egress LC hosting backup path.

The command **frr-holdtime** configures frr-holdtime to be proportional to the scale number of LSPs. The recommended frr-holdtime value is either the same, or lesser than the MBB delay timer. This ensures that the egress LC is in FRR state after the primary path down event.

When not configured, the default frr-holdtimer, in seconds, is set to 2.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**cef platform lsm frr-holdtime** <*seconds*> **Example:** `RP/0/RSP0/CPU0:router(config)# cef platform lsm frr-holdtime 30`	Configures frr-holdtime to be proportional to the scale number of LSPs. In this case, configures the frr-holdtime to 30 seconds.
Step 3	**commit**	

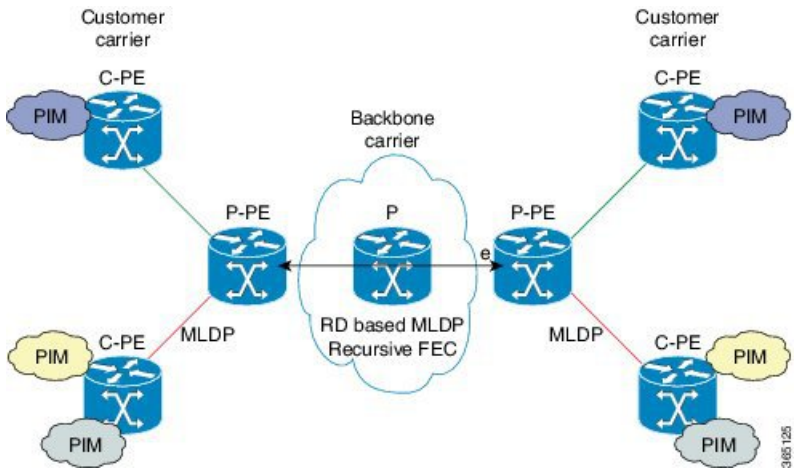
Example: Configure FRR Holdtime

```
cef platform lsm frr-holdtime ?
  <3-180>  Time in seconds
cef platform lsm frr-holdtime 45
commit
!
!
```

MLDP Carrier Supporting Carrier Based MVPN

The carrier-supporting-carrier (CSC) feature enables one MPLS VPN-based service provider to allow other service providers to use a segment of its backbone network. The service provider that provides the segment of backbone network to the other provider is called the backbone carrier, whereas the service provider that uses the backbone network is called the customer carrier. The customer carrier can be either an ISP itself or a BGP or MPLS VPN service provider, and can run an IP or MPLS in its network in former and later cases respectively. In either case, MPLS is run in backbone network and between the backbone and the customer carrier (on PE-CE link).

Figure 18: MLDP CSC based MVPN



In the above illustration, P-PE and P routers are a part of backbone carrier. Customer carrier PEs is labeled C-PE. The Link between P-PE and C-PE is on VRF on P-PE and global table on C-PE. LDP/MLDP sessions run in VRF context on P-PEs link towards C-PE. There is an iBGP sessions between P-PEs exchanging vpnv4 addresses.

MLDP CsC - Restrictions

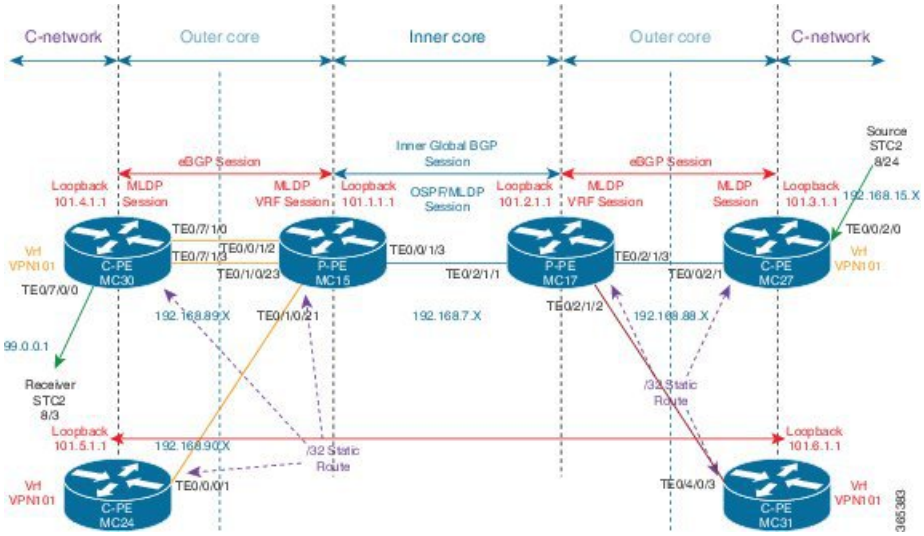
The following are the limitations of the MLDP CsC solution:

- P2MP LSPs are supported for CsC, however, no MP2MP support is provided.
- MBB cannot be enabled per VRF. It is either to be enabled for all VRFs or none can be enabled.
- MBB delay can be configured per VRF only.

CsC Configuration Example - Overview

The following figure describes an example configuration of the CsC feature:

Figure 19: CsC - Configuration Overview



The network is consists of:

- Two cores: Inner and outer cores.
  - Inner Core: includes P-PE and P routers.
  - Outer Core: includes P-PE routers which are connected directly to C-PE routers.
  - VRF-lite: more than one C-PE connected to the same P-PE.
- IGP: [OSPF] on inner core routers in the global table (not on VRFs)
- BGP:
  - BGP/iBGP between P-PE routers.
  - eBGP between C-PE and P-PE routers.

Note C-PE and P-PE are directly connected.

- Static Routing: between C-PE and P-PE to trigger a creation of a label
- MLDP/MPLS:
  - Two types of sessions: Global table of P-PE and P routers (of the inner core) and VRF of the P-PE routers and the global table of the C-PE routers.
  - Peer model: a P2MP tree is created in the inner core for each P2MP that exists in the outer core. When data MDT is selected, one LSP is created for each Mroute.
- PIM/Multicast: Not run either in the inner or outer cores. The inner core is transparent to PIM. Only profiles 12 and 14 are applicable.

MC-15: Basic VRF and Interface Configuration

```
vrf vpn101
 vpn id 1:1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
! Loopback interfaces
interface Loopback0
 ipv4 address 15.15.15.15 255.255.255.255 !
interface Loopback101
 vrf vpn101
 ipv4 address 101.1.1.1 255.255.255.255
!

! core interface
interface TenGigE0/0/1/3
 ipv4 address 192.168.7.1 255.255.255.0
!

! vrf interface
interface TenGigE0/1/0/23
 vrf vpn101
 ipv4 address 192.168.89.1 255.255.255.0  transceiver permit pid all !

route-policy p2mp
  set core-tree mldp-default
end-policy
!
route-policy pass-all
  pass
end-policy
!
router static
 address-family ipv4 unicast
  0.0.0.0/0 172.18.51.1
 !
 vrf vpn101
  address-family ipv4 unicast
   192.168.89.2/32 TenGigE0/1/0/23
   192.168.90.2/32 TenGigE0/1/0/21
  !
 !
!
router ospf 0
 nsr
 router-id 15.15.15.15
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface TenGigE0/0/1/3
  !
 !
 mpls traffic-eng router-id 15.15.15.15
!
router bgp 100
 nsr
 mvpn
```

```
 bgp router-id 15.15.15.15
 bgp graceful-restart
 address-family vpnv4 unicast
 !
 neighbor 17.17.17.17
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  !
 !
 vrf vpn101
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
   redistribute static
   allocate-label all
  !
  neighbor 192.168.89.2
   remote-as 101
   address-family ipv4 labeled-unicast
    route-policy pass-all in
    route-policy pass-all out
    as-override
   !
  !
  neighbor 192.168.90.2
   remote-as 101
   address-family ipv4 labeled-unicast
    route-policy pass-all in
    route-policy pass-all out
    as-override
   !
  !
 !
!
mpls traffic-eng
 interface TenGigE0/0/1/3
 !
!
mpls ldp
 log
  neighbor
 !
 nsr
 graceful-restart
 mldp
  address-family ipv4
   carrier-supporting-carrier
   make-before-break delay 100
   recursive-fec
  !
 !
 router-id 15.15.15.15
 session protection
 interface TenGigE0/0/1/3
 !
 vrf vpn101
  router-id 101.1.1.1
  address-family ipv4
  !
  interface TenGigE0/1/0/21
   address-family ipv4
   !
  !
```

```
  interface TenGigE0/1/0/23
   address-family ipv4
   !
  !
 !
!
```

MC-30: Basic VRF and Interface Configuration

```
vrf vpn101
 vpn id 10:1
 address-family ipv4 unicast
  import route-target
   10:1
  !
  export route-target
   10:1
  !
 !
 address-family ipv6 unicast
  import route-target
   10:1
  !
  export route-target
   10:1
  !
 !
!
interface Loopback0
 ipv4 address 30.30.30.30 255.255.255.255 !
interface Loopback101
 ipv4 address 101.4.1.1 255.255.255.255
!
interface Loopback1100
 vrf vpn101
 ipv4 address 101.4.100.1 255.255.255.255 !
interface Loopback1111
 ipv4 address 101.4.1.11 255.255.255.255 !

! Core interface
interface TenGigE0/7/1/3
 ipv4 address 192.168.89.2 255.255.255.0  transceiver permit pid all !
route-policy p2mp
  set core-tree mldp-default
end-policy
!
route-policy CSC-PEER
  if destination in (192.168.89.1/32) then
    pass
  endif
end-policy
!
route-policy pass-all
  pass
end-policy
!
route-policy rosen-mldp
  set core-tree mldp-default
end-policy
!
router static
 address-family ipv4 unicast
  0.0.0.0/0 172.18.51.1
  192.168.89.1/32 TenGigE0/7/1/3
```

```
 !
!
router bgp 101
 nsr
 bgp router-id 192.168.89.2
 bgp graceful-restart
 address-family ipv4 unicast
  redistribute connected
  redistribute static route-policy CSC-PEER
  allocate-label all
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
 neighbor 101.3.1.1
  remote-as 101
  update-source Loopback101
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
neighbor 101.5.1.1
  remote-as 101
  update-source Loopback101
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
 neighbor 192.168.89.1
  remote-as 100
  ebgp-multihop 55
  address-family ipv4 labeled-unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 vrf vpn101
  rd 10:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
```

```
!
mpls ldp
 log
  neighbor
 !
 nsr
 graceful-restart
 mldp
  address-family ipv4
  !
 !
 router-id 101.4.1.1
 session protection
 interface TenGigE0/7/1/3
 !
!
multicast-routing
 address-family ipv4
  mdt source Loopback101
  interface all enable
 !
 address-family ipv6
  mdt source Loopback101
  interface all enable
 !
 vrf vpn101
  address-family ipv4
   mdt source Loopback1111
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
   !
   mdt default mldp p2mp
  !
  address-family ipv6
   mdt source Loopback1112
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
    inter-as
   !
   mdt default mldp p2mp
   mdt data 5
  !
 !
!
router pim
 address-family ipv4
  hello-interval 600
  join-prune-interval 180
  nsf lifetime 180
 !
 vrf vpn101
  address-family ipv4
   rpf topology route-policy p2mp
   hello-interval 600
   join-prune-interval 180
   mdt c-multicast-routing bgp
   !
   rp-address 101.3.1.11
   log neighbor changes
   bsr candidate-bsr 101.4.100.1 hash-mask-len 32 priority 1
```

```
   bsr candidate-rp 101.4.100.1 priority 192 interval 60
  !
  address-family ipv6
   rpf topology route-policy p2mp
   hello-interval 600
  !
 !
!
```

Configuration Changes for CsC

The following are the configuration changes that are required for supporting CsC solution:

Configuration Changes for IGP for CsC

The following are the configuration changes that are required for IGP for supporting CsC solution:

- OSPF configuration: on the two inner core P-PE routers, MC15 and MC17.
- Static routes: required for CsC features, are configured on the eBGP links between P-PE and C-PE.

The IGP configurations will be similar to the following:

On MC15

```
router ospf 0
 nsr
 router-id 15.15.15.15
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface TenGigE0/0/1/3
  !
 !
 mpls traffic-eng router-id 15.15.15.15
!
```

On MC17

```
router ospf 0
 nsr
 router-id 17.17.17.17
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface TenGigE0/2/1/1
  !
 !
 mpls traffic-eng router-id 17.17.17.17
!
```

On MC30, eBGP link between P-PE

```
router static
 address-family ipv4 unicast
  0.0.0.0/0 172.18.51.1
  192.168.89.1/32 TenGigE0/7/1/3
 !
```

On MC15, eBGP link between C-PE

```
router static
 address-family ipv4 unicast
  0.0.0.0/0 172.18.51.1
 !
 vrf vpn101
  address-family ipv4 unicast
   192.168.89.2/32 TenGigE0/1/0/23
   192.168.90.2/32 TenGigE0/1/0/21
!
```

Configuration Changes for Supporting MLDP CsC

The following are the configuration changes that are required for supporting mLDP CsC solution:

- Enable MLDP globally.
- Enable LDP under VRF. This enables MLDP on same VRF.
- Enable the MLDP-specific VRF configurations (like the MBB, recursive forwarding, and so on) under the MLDP VRF submode.
- Configure the new Carrier supporting Carrier knob, added under global MLDP submode, on ingress P-PE routers.

The MLDP configuration will be similar to the following:

On MC30, C-PE router

```
mpls ldp
 log
  neighbor
 !
 nsr
 graceful-restart
 mldp
  address-family ipv4
  !
 !
 router-id 101.4.1.1
 session protection
 interface TenGigE0/7/1/3
 !
!
```

On MC15, P-PE router

```
mpls traffic-eng
 interface TenGigE0/0/1/3
 !
!
mpls ldp
 log
  neighbor
 !
 nsr
 graceful-restart
 mldp
  address-family ipv4
   carrier-supporting-carrier
```

```
   make-before-break delay 100
   recursive-fec
  !
 !
 router-id 15.15.15.15
 session protection
 interface TenGigE0/0/1/3
 !
 vrf vpn101
  router-id 101.1.1.1
  address-family ipv4
  !
  interface TenGigE0/1/0/21
   address-family ipv4
   !
  !
  interface TenGigE0/1/0/23
   address-family ipv4
```

Configuration Changes for iBGP and eBGP for CsC

The following are the configuration changes that are required for iBGP and eBGP for supporting CsC solution:

- Configure iBGP between P-PE routers.
- Configure eBGP between P-PE and C-PE.

The configurations will be similar to the following:

On MC15, iBGP between P-PE

```
router bgp 100
 nsr
 mvpn
 bgp router-id 15.15.15.15
 bgp graceful-restart
 address-family vpnv4 unicast
 !
 neighbor 17.17.17.17
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  !
 !
vrf vpn101
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
   redistribute static
   allocate-label all
  !
  neighbor 192.168.89.2
   remote-as 101
   address-family ipv4 labeled-unicast
    route-policy pass-all in
    route-policy pass-all out
    as-override
   !
  !
  neighbor 192.168.90.2
   remote-as 101
   address-family ipv4 labeled-unicast
    route-policy pass-all in
```

```
    route-policy pass-all out
    as-override
   !
```

On MC30, eBGP between P-PE and C-PE

```
router bgp 101
 nsr
 bgp router-id 192.168.89.2
 bgp graceful-restart
 address-family ipv4 unicast
  redistribute connected
  redistribute static route-policy CSC-PEER
  allocate-label all
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
 neighbor 101.3.1.1
  remote-as 101
  update-source Loopback101
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
  neighbor 101.5.1.1
  remote-as 101
  update-source Loopback101
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
 neighbor 192.168.89.1
  remote-as 100
  address-family ipv4 labeled-unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 vrf vpn101
  rd 10:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
```

```
  address-family ipv6 mvpn
```

Configuration Changes for Multicast for CsC

The following are the configuration changes that are required for Multicast for supporting CsC solution.

Note Multicast is active only in C-network on C-PE routers.

The configurations will be similar to the following:

On MC27

```
multicast-routing
 address-family ipv4
  mdt source Loopback101
  interface all enable
 !
 address-family ipv6
  mdt source Loopback101
  interface all enable
 !
vrf vpn101
  address-family ipv4
   mdt source Loopback1111
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
   !
   mdt default mldp p2mp
   mdt data 5
  !
  address-family ipv6
   mdt source Loopback1112
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
    inter-as
   !
   mdt default mldp p2mp
   mdt data 5
  !
 !
!


router pim
 address-family ipv4
  auto-rp mapping-agent TenGigE0/0/2/0 scope 11 interval 60
  auto-rp candidate-rp TenGigE0/0/2/0 scope 10 group-list 224-4 interval 60
  nsf lifetime 180
 !
vrf vpn101
  address-family ipv4
   rpf topology route-policy p2mp
   hello-interval 1
   mdt c-multicast-routing bgp
```

```
   !
   bsr candidate-bsr 101.3.100.1 hash-mask-len 32 priority 1
   bsr candidate-rp 101.3.100.1 priority 192 interval 60
  !
  address-family ipv6
   rpf topology route-policy p2mp
   hello-interval 3600
   mdt c-multicast-routing pim
   !
  !
 !
!
```

Multipoint Label Distribution Protocol Route Policy Map

Multicast supports Multipoint Label Distribution Protocol Route Policy Map, wherein Multipoint Label Distribution Protocol uses the route policy maps to filter Label Mappings and selectively apply the configuration features on Cisco IOS-XR operating system.

Route policy map for configuration commands:

The route policy map for the configuration commands provide you the flexibility to selectively enable some of the mLDP features such as Make Before Break (MBB), Multicast only FRR (MoFRR) features, and so on, on the applicable LSPs. Features like Make Before Break (MBB), Multicast only FRR (MoFRR), etc. can be enabled on mLDP on IOS-XR operating system. When each of these features are enabled, they are enabled for all of the mLDP Labeled-Switched Paths (LSPs) irrespective of whether they are applicable for the particular LSP or not. For example, MoFRR is used for IPTV over mLDP in-band signaled P2MP LSPs, but not for the generic MVPN using a MP2MP LSPs. Using the route policy map, you can configure mLDP to to selectively enable some of the features.

Route policy for label mapping filtering:

The route policy map for the Label Mapping filtering provides a way to prevent the mLDP from crossing over from one plane to another in the event of a failure.

Generally, the LSPs based on mLDP are built on unicast routing principle, and the LSPs follow unicast routing as well. However, some networks are built on the concept of dual-plane design, where an mLDP LSP is created in each of the planes to provide redundancy. In the event of a failure, mLDP crosses over to another plane. To prevent mLDP from crossing over, mLDP Label Mappings are filtered either in an inbound or outbound direction.

mLDP uses the existing RPL policy infrastructure in IOS-XR. With the existing RPL policy, mLDP FECs are created and compared with the real mLDP FEC for filtering and configuration commands. (To create mLDP FECs for filtering, create a new RPL policy (specific for mLDP FECs) with the necessary show and configuration commands.) An mLDP FEC consists of 3 tuples: a tree type, root node address, and the opaque encoding, which uniquely identifies the mLDP LSP. An opaque encoding has a different TLV encoding associated with it. For each of the different opaque TLV, a unique RPL policy is to be created since the information in the mLDP opaque encoding is different.

The implementation of mLDP FEC based RPL filter is done in both RPL and LDP components.

- mLDP FEC

  The mLDP FEC Route Policy Filtering is a combination of a root node and opaque types.

  - Root Node:

    Filtering is allowed only at the root node in combination with opaque types.

- Opaque Types:

  The following are the opaque types allowed to create the Route Policies.

  - IPV4 In-band type
  - IPV6 In-band type
  - VPNv4 In-band type
  - VPNv6 In-band type
  - MDT Rosen model (VPN-ID) type
  - Global ID type
  - Static ID type
  - Recursive FEC type
  - VPN Recursive FEC type

- mLDP Label Mapping Filtering:

  Label mapping filtering is supported either in inbound or outbound directions, based on the user preference. All default policies applicable in the neighborhood are supported by Label Mapping Filtering.

- mLDP Feature Filtering:

  The RPL policy allows selective features to be enabled, applies to the following feature configuration commands:

  - MoFRR
  - Make Before Break
  - Recursive FEC

Configuring mLDP User Interface (Opaque Types) Using the Routing Policy

Perform this task to configure the LDP user interface using the route policy to filter Label Mappings and selectively apply the configuration features. LDP interface can be configured using the various available mLDP opaque parameters like the Global ID, IPv4, IPv6, MDT, Recursive, Recursive RD, Static ID, VPNv4, and VPNv6.

See the *Implementing Routing Policy* on Cisco ASR 9000 Series Router module of *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* for a list of the supported attributes and operations that are valid for policy filtering.

Configuring the mLDP User Interface for LDP Opaque Global ID Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque global-id** *32-bit decimal number* **then pass endif**
4. **end-policy**

5. **commit**
6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque global-id** *32-bit decimal number* **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque global-id then pass endif`	Configures the mLDP global id to the specific global-id.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Wed Jun 18 11:41:09.333 IST`
		`route-policy mldp_policy`
		`  if mldp opaque global-id 10 then`
		`    pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque IPv4 Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque ipv4** [*ipv4 address*| *ipv4 address range*] **then pass endif**
4. **end-policy**
5. **commit**

6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque ipv4** [*ipv4 address*| *ipv4 address range*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque ipv4 then pass endif`	Configures the mLDP ipv4 address variable to the specified range of IPv4 IP address.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Sun Jun 22 20:03:34.308 IST`
		`route-policy mldp_policy`
		`  if mldp opaque ipv4 10.0.0.1 224.1.1.1 then`
		`  pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque IPv6 Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque ipv6** [*ipv6 address*| *ipv6 address range*] **then pass endif**
4. **end-policy**
5. **commit**

6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque ipv6** [*ipv6 address*| *ipv6 address range*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque ipv6 then pass endif`	Configures the mLDP ipv6 variable to the specified range of IPv6 IP address.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Sun Jun 22 20:03:34.308 IST`
		`route-policy mldp_policy`
		`  if mldp opaque ipv6 10::1 ff05::1 then`
		`  pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque MDT Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque mdt** [*1:1*] **then pass endif**
4. **end-policy**
5. **commit**

6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque mdt** [*1:1*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque mdt then pass endif`	Configures the mLDP VPNID to the specific MDT number.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	Example outputs are as shown:
		`Sun Jun 22 20:03:34.308 IST`
		`route-policy mldp_policy`
		`  if mldp opaque mdt 1:1 0 then`
		`  pass`
		`  endif`
		`end-policy`
		
		`route-policy mldp_policy`
		`  if mldp opaque mdt any 10 then`
		`  pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque Static ID Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque static-id** *32-bit decimal number* **then pass endif**

4. **end-policy**
5. **commit**
6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque static-id** *32-bit decimal number* **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque static-id then pass endif`	Configures the mLDP static id to the specific static id.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown below:
		`Wed Jun 18 11:41:09.333 IST`
		`route-policy mldp_policy`
		`  if mldp opaque static-id 10 then`
		`    pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque Recursive Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque recursive then pass endif**
4. **end-policy**

5. **commit**
6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque recursive then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque recursive then pass endif`	Configures the mLDP recursive variable.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Mon Jun 23 11:46:15.559 IST`
		`route-policy mldp_policy`
		`  if mldp opaque recursive then`
		`    pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque Recursive-RD Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque recursive -rd** [*2:2*] **then pass endif**
4. **end-policy**
5. **commit**

6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque recursive -rd** [*2:2*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque recursive-rd then pass endif`	Configures the mLDP recursive to the specified variable.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Mon Jun 23 12:15:37.512 IST`
		`route-policy mldp_policy`
		`  if mldp opaque recursive-rd 2:2 then`
		`  pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque VPNv4 Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque vpnv4** [*2:2*] **then pass endif**
4. **if mldp opaque vpnv4** [*2:2 10.1.1.1 232.1.1.1*] **then pass endif**
5. **end-policy**
6. **commit**

7. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque vpnv4** [*2:2*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque vpnv4 then pass endif`	Configures the mLDP vpnv4 variable to the specified variable.
Step 4	**if mldp opaque vpnv4** [*2:2 10.1.1.1 232.1.1.1*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque vpnv4 then pass endif`	Configures the mLDP vpnv4 variable to the specified range of variable addresses.
Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 6	**commit**	

Step 7	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	Example outputs are as shown:
		`Sun Jun 22 20:03:34.308 IST`
		`route-policy mldp_policy`
		`  if mldp opaque vpnv4 2:2 10.1.1.1 232.1.1.1 then`
		`    pass`
		`  endif`
		`end-policy`
		`route-policy mldp_policy`
		`  if mldp opaque vpnv4 any 0.0.0.0 224.1.1.1 then`
		`    pass`
		`  endif`
		`end-policy`
		`!`

Configuring the mLDP User Interface for LDP Opaque VPNv6 Using the Routing Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp opaque vpnv6** [*2:2*] **then pass endif**
4. **if mldp opaque vpnv6** [*2:2 10::1 FF05::1*] **then pass endif**
5. **end-policy**
6. **commit**
7. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp opaque vpnv6** [*2:2*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque vpnv4 then pass endif`	Configures the mLDP vpnv6 variable to the specified variable.
Step 4	**if mldp opaque vpnv6** [*2:2 10::1 FF05::1*] **then pass endif** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp opaque vpnv6 then pass endif`	Configures the mLDP vpnv6 variable to the specified variable range of addresses.
Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 6	**commit**	

Step 7	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	An example output is as shown:
		`Sun Jun 22 20:03:34.308 IST`
		`route-policy mldp_policy`
		`  if mldp opaque vpnv6 2:2 10::1 ff05::1 then`

	Command or Action	Purpose
		pass endif end-policy !

Configuring mLDP FEC at the Root Node

Perform this task to configure mLDP FEC at the root node using the route policy to filter Label Mappings and selectively apply the configuration features. Currently, mLDP FEC is configured to filter at the IPV4 root node address along with the mLDP opaque types.

Configuring the mLDP FEC at the Root Node Using the Route Policy

SUMMARY STEPS

1. **configure**
2. **route-policy mldp_policy**
3. **if mldp root**
4. **end-policy**
5. **commit**
6. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config)#route-policy mldp_policy`	Enters the Route-policy configuration mode, where you can define the route policy.
Step 3	**if mldp root** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# if mldp root[ipv4 address]then pass endif`	Configures the mLDP root address to the specified IPv4 IP address.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	
Step 5	**commit**	

	Command or Action	Purpose

Step 6	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	The current configuration output is as shown:
		`route-policy mldp_policy`
		`  if mldp root 10.0.0.1 then`
		`  pass`
		`  endif`
		`end-policy`
		`!`

Example of an MLDP Route Policy which shows the filtering option of a Root Node IPv4 address and mLDP Opaque IPv4 address

Show configuration output for the mLDP root IPv4 address and mLDP opaque IPv4 address range

```
route-policy mldp_policy
  if mldp root 10.0.0.1 and mldp opaque ipv4 192.168.3.1 232.2.2.2 then
    pass
  endif
end-policy
!
```

Configuring the mLDP User Interface to Filter Label Mappings

Label mapping filtering is supported either in inbound or outbound directions, based on the user preference. All default policies applicable in the neighborhood are supported by Label Mapping Filtering.

Configuring the mLDP User Interface to Filter Label Mappings

SUMMARY STEPS

1. **configure**
2. **mpls ldp mldp**
3. **address-family ipv4**
4. **neighbor**[*ipv4 ip address*]**route-policy mldp_policy in | out**
5. **end-policy**
6. **commit**
7. Use the show command to verify the configuration: **show running-config route-policy mldp_policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**mpls ldp mldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp mldp`	Enters the LDP configuration mode.
Step 3	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# address-family ipv4`	Enters the MLDP address family configuration mode.
Step 4	**neighbor**[*ipv4 ip address*]**route-policy mldp_policy in | out** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# neighbor ipv4 ip address route-policy mldp_policy in | out`	Configures the specified neighborhood IPv4 IP address to the MLDP policy as either inbound or outbound route policy.
Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# end-policy`	
Step 6	**commit**	

Step 7	Use the show command to verify the configuration: **show running-config route-policy mldp_policy**	Example outputs are as shown:
		`Wed Jun 18 11:41:09.333 IST`
		`mpls ldp`
		`mldp`
		`neighbor route-policy mldp_policy out`
		`!`
		`mpls ldp`
		`mldp`
		`neighbor 172.16.0.3 route-policy mldp_policy in`
		`!`

Configuring the mLDP User Interface for Feature Filtering

RPL policy allows the user to selectively enable features for filtering.

Configuring the mLDP User Interface for Feature Filtering - MoFRR

SUMMARY STEPS

1. **configure**
2. **mpls ldp mldp**
3. **mofrr route-policy mldp_policy**
4. **end**
5. **commit**
6. Use the show command to verify the configuration: **show running-config mpls ldp mldp**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp mldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp mldp`	Enters the LDP configuration mode.
Step 3	**mofrr route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# mofrr route-policy mldp_policy`	Configures the specified feature to the MLDP policy to be allowed to be selected for filtering.
Step 4	**end** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# end`	
Step 5	**commit**	

Step 6	Use the show command to verify the configuration: **show running-config mpls ldp mldp**	An example output is as shown:
		`Wed Jun 25 12:46:52.177 IST`
		`mpls ldp`
		`mldp`
		`address-family ipv4`
		`make-before-break delay 0`
		`mofrr route-policy mldp_policy`
		`neighbor route-policy mldp_policy out`
		`neighbor 172.16.0.2 route-policy mldp_policy in`
		`neighbor 172.16.0.2 route-policy mldp_policy out`
		`neighbor 2.2.2.2 route-policy mldp_policy out`
		`recursive-fec route-policy mldp_policy`
		`!`
		`!`
		`!`

An example output showing the mLDP MoFRR output

```
RP/0/1/CPU0:GSR3#sh mpls mldp database opaquetype ipv4
mLDP database
LSM-ID: 0x00019   Type: P2MP   Uptime: 03:25:15
  FEC Root           : 10.0.0.1
  Opaque decoded     : [ipv4 0.0.0.0 224.1.1.1]
  Features           : MoFRR
```

```
Upstream neighbor(s) :
  10.0.0.1:0    [Active]       Uptime: 03:25:15
    Next Hop        : 10.0.3.3
    Interface       : GigabitEthernet0/2/1/1
    Local Label (D) : 16028
Downstream  client(s):
  LDP 10.0.0.2:0     Uptime: 03:25:15
    Next Hop         : 10.0.4.2
    Interface        : GigabitEthernet0/2/1/2
    Remote label (D) : 16029
```

Configuring the mLDP User Interface for Feature Filtering - Make-before-break

SUMMARY STEPS

1. **configure**
2. **mpls ldp mldp**
3. **address-family ipv4**
4. **make-before-break route-policy mldp_policy**
5. **end**
6. **commit**
7. Use the show command to verify the configuration: **show running-config mpls ldp mldp**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp mldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp mldp`	Enters the LDP configuration mode.
Step 3	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)#address-family ipv4`	Enters the LDP address family configuration mode.
Step 4	**make-before-break route-policy mldp_policy** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# make-before-break route-policy mldp_policy`	Configures the specified feature to the MLDP policy to be allowed to be selected for filtering.
Step 5	**end** **Example:**	

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# end`	
Step 6	**commit**	

Step 7	Use the show command to verify the configuration: **show running-config mpls ldp mldp**	An example output is as shown:
		
		`Wed Jun 25 13:05:31.303 IST`
		`mpls ldp`
		`mldp`
		`address-family ipv4`
		`make-before-break delay 0`
		`make-before-break route-policy mldp_policy`
		`mofrr route-policy mldp_policy`
		`neighbor route-policy mldp_policy out`
		`neighbor 172.16.0.2 route-policy mldp_policy in`
		
		`neighbor 172.16.0.2 route-policy mldp_policy out`
		`neighbor 2.2.2.2 route-policy mldp_policy out`
		`!`
		`!`
		`!`

An example output showing the mLDP make-before-break output

```
RP/0/1/CPU0:GSR3#sh mpls mldp database opaquetype ipv4
mLDP database
LSM-ID: 0x00019   Type: P2MP   Uptime: 03:25:15
  FEC Root           : 10.0.0.1
  Opaque decoded     : [ipv4 0.0.0.0 224.1.1.1]
  Features           : MoFRR MBB
  Upstream neighbor(s) :
    10.0.0.1:0    [Active]       Uptime: 03:25:15
      Next Hop        : 10.0.3.3
      Interface       : GigabitEthernet0/2/1/1
      Local Label (D) : 16028
  Downstream  client(s):
    LDP 10.0.0.2:0     Uptime: 03:25:15
      Next Hop         : 10.0.4.2
      Interface        : GigabitEthernet0/2/1/2
      Remote label (D) : 16029
```

Configuring the mLDP User Interface for Feature Filtering - Recursive FEC

SUMMARY STEPS

1. **configure**
2. **mpls ldp mldp**
3. **address-family ipv4**
4. **recursive-fec route-policy rfec**
5. **end**
6. **commit**
7. Use the show command to verify the configuration: **show running-config mpls ldp mldp**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**mpls ldp mldp** **Example:** `RP/0/RSP0/CPU0:router(config)#mpls ldp mldp`	Enters the LDP configuration mode.
Step 3	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)#address-family ipv4`	Enters the LDP address family configuration mode.
Step 4	**recursive-fec route-policy rfec** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# recursive-fec route-policy rfec`	Configures the specified feature to the MLDP policy to be allowed to be selected for filtering.
Step 5	**end** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp-af)# end`	
Step 6	**commit**	

Step 7	Use the show command to verify the configuration: **show running-config mpls ldp mldp**	An example output is as shown:
		`Wed Jun 25 13:05:31.303 IST`
		`mpls ldp`
		`mldp`
		`address-family ipv4`
		`make-before-break delay 0`
		`make-before-break route-policy mldp_policy`
		`mofrr route-policy mldp_policy`
		`neighbor route-policy mldp_policy out`
		`neighbor 172.16.0.2 route-policy mldp_policy in`
		`neighbor 172.16.0.2 route-policy mldp_policy out`
		`neighbor 2.2.2.2 route-policy mldp_policy out`
		`recursive-fec route-policy rfec`
		`!`
		`!`
		`!`

An example output showing the mLDP make-before-break output

```
RP/0/1/CPU0:GSR3#sh mpls mldp database opaquetype ipv4
mLDP database
LSM-ID: 0x00019   Type: P2MP   Uptime: 03:25:15
  FEC Root           : 10.0.0.1
  Opaque decoded     : [ipv4 0.0.0.0 224.1.1.1]
  Features           : MoFRR MBB RFEC
  Upstream neighbor(s) :
    10.0.0.1:0    [Active]       Uptime: 03:25:15
      Next Hop        : 10.0.3.3
      Interface       : GigabitEthernet0/2/1/1
      Local Label (D) : 16028
  Downstream  client(s):
    LDP 10.0.0.2:0     Uptime: 03:25:15
      Next Hop         : 10.0.4.2
      Interface        : GigabitEthernet0/2/1/2
      Remote label (D) : 16029
```

Limitations of Route Policy Map

Limitations:

The following are the limitations of the route policy map:

- After changing the Route Policy filter to be more restrictive, the mLDP label bindings that were earlier allowed are not removed. You have to run the **clear mpls ldp neighbor** command to clear the mLDP database.
- If you select a less restrictive filter, mLDP initiates a wildcard label request in order to install the mLDP label bindings that were denied earlier.
- Creating an RPL policy that allows filtering based on the recursive FEC content is not supported.
- Applying an RPL policy to configuration commands impacts the performance to a limited extent.

Next-Generation Multicast VPN

Next-Generation Multicast VPN (NG-MVPN) offers more scalability for Layer 3 VPN multicast traffic. It allows point-to-multipoint Label Switched Paths (LSP) to be used to transport the multicast traffic between PEs, thus allowing the multicast traffic and the unicast traffic to benefit from the advantages of MPLS transport, such as traffic engineering and fast re-route. This technology is ideal for video transport as well as offering multicast service to customers of the layer 3 VPN service.

NG-MVPN supports:

- VRF Route-Import and Source-AS Extended Communities
- Upstream Multicast Hop (UMH) and Duplicate Avoidance
- Leaf AD (Type-4) and Source-Active (Type-5) BGP AD messages
- Default-MDT with mLDP P2MP trees and with Static P2MP-TE tunnels

- BGP C-multicast Routing
- RIB-based Extranet with BGP AD
- Accepting (*,G) S-PMSI announcements
- Egress-PE functionality for Ingress Replication (IR) core-trees
- Enhancements for PIM C-multicast Routing
- Migration of C-multicast Routing protocol
- PE-PE ingress replication
- Dynamic P2MP-TE tunnels
- Flexible allocation of P2MP-TE attribute-sets
- Data and partitioned MDT knobs
- Multi-instance BGP support
- SAFI-129 and VRF SAFI-2 support
- Anycast-RP using MVPN SAFI

Supported Features

The following are the supported features on next generation Multicast MVPN on IOS-XR:

- GTM using MVPN SAFI
- MVPN enhancements

GTM using MVPN SAFI

In a GTM procedure, special RD values are used that are created in BGP. The values used are all 0's RD. A new knob, **global-table-multicast** is introduced under BGP to create the contexts for these RDs.

MVPN procedures require addition of VRF Route-Import EC, Source-AS EC, and so on to the VPNv4 routes originated by PEs. With GTM, there are no VRFs and no VPNv4 routes. The multicast specific attributes have to be added to Global table iBGP routes (either SAFI-1 or SAFI-2). These routes are learnt through eBGP (from a CE) or from a different Unicast routing protocol.

- The single forwarder selection is not supported for GTM.
- **Route Targets:** With GTM, there are no VRFs, hence the export and import RTs configured under VRFs are not reliable. For MVPN SAFI routes, RT(s) must be attached. Export and import Route Targets configuration under multicast routing is supported. These are the RTs used for Type 1, 3, and 5 routes. MVPN SAFI routes received without any RTs will not be accepted by an XR PE.
- **Core-Tree Protocols:** mLDP, P2MP-TE (static and dynamic), and IR core-trees are supported.
- **C-multicast Routing:** PIM and BGP C-multicast routing are supported.
- **MDT Models:** Default-MDT and Partitioned-MDT models are supported. Data-MDT is supported, with its various options (threshold zero, immediate-switch, starg s-pmsi, and so on.)

The configuration is as shown below for Ingress or Egress PEs:

```
multicast-routing
 address-family [ipv4| ipv6]
  mdt source Loopback0
  mdt default <MLDP | P2MP-TE | ingress-replication>
  mdt partitioned <MLDP | P2MP-TE | ingress-replication>
  bgp auto-discovery [mldp | p2mp-te | ingress-replication]
   export-rt <value>
   import-rt <value>
  !
 !
!
```

Note

- The mdt default, mdt partitioned, and the bgp auto-discovery configurations, are present under VRFs, however, with GTM Using MVPN SAFI, the configurations are reflected in global table as well.
- Do not reuse the same RT values for GTM and regular VRF-based MVPN. For VRF-based MVPN, use the VRF route-target configuration unless multicast-specific RT scoping is required.

```
router bgp 100
 address-family [ipv4| ipv6] mvpn
  global-table-multicast
 !
!
```

The global-table-multicast configuration enables processing of All-0's RD.

MVPN enhancements

- **Anycast RP using MVPN SAFI** This procedure uses Type-5 MVPN SAFI routes to convey source information between RPs. Use this method to support Anycast-RP, instead of using MSDP. This supports Anycast-RP for both IPv4 and IPv6. Currently, Anycast-RP is supported for IPv4 (using MSDP). BGP method is supported for GTM using MVPN SAFI and MVPNs.

  The configuration is as shown below for Ingress or Egress PEs:

  ```
  multicast-routing
   address-family [ipv4| ipv6]
    bgp auto-discovery [mldp | p2mp-te | ingress-replication]
     anycast-rp route-policy <anycast-policy>
    !
   !
   vrf <name>
    address-family [ipv4| ipv6]
     bgp auto-discovery [mldp | p2mp-te | ingress-replication]
      anycast-rp route-policy <anycast-policy>
    !
   !
  !
  ```

  The route-policy for anycast RP is as defined below.

  ```
  route-policy anycast-policy
  ```

```
  if destination in group-set then
    pass
  endif
end-policy
!
```

The **group-set** command is a XR prefix-set configuration, an example is as shown below:

```
prefix-set group-set
227.1.1.1/32
end-set
```

Note When configuring the **bgp auto-discovery anycast-rp route-policy**, you must explicitly permit the multicast groups used for sparse-mode. A default "pass" policy is insufficient for this purpose. The prefix configured in the route-policy (RPL) becomes a Route-Target in the Type-5 "source active" route originated by the source PE/RP, which must be matched on the receiver PE/RP.

Example of an incorrect configuration:

```
route-policy anycast-policy
  pass
end-policy

multicast-routing
  vrf one
  address-family ipv4
  bgp auto-discovery mldp
  anycast-rp route-policy anycast-policy
```

To configure a default "allow all groups" policy, omit the route-policy in the **bgp auto-discovery anycast-rp** configuration, as shown below:

```
multicast-routing
  vrf one
  address-family ipv4
  bgp auto-discovery mldp
  anycast-rp
```

An alternate way of performing this procedure is using export-rt and import-rt configuration commands. Here, the router announcing the Type-5 route must have the export-rt configured, and the router learning the source must have the import-rt configured.

```
multicast-routing
 vrf one
  address-family ipv4
   export-rt 51.52.53.54:0 <<<<<<<
   interface all enable
   bgp auto-discovery ingress-replication
    inter-as
    anycast-rp <<<<<<<<<
   !
   mdt partitioned ingress-replication
  !
 !
!
```

```
multicast-routing
 vrf one
  address-family ipv4
   import-rt 51.52.53.54:0 <<<<<<<<<<<
   interface all enable
   bgp auto-discovery ingress-replication
    inter-as
    anycast-rp <<<<<<<<<<<<<<<
   !
   mdt partitioned ingress-replication
  !
 !
!
```

- **Receiver-only VRFs** Supports receiver-only VRFs. In receiver-only VRFs, the I-PMSI or the MS-PMSI routes do not carry any tunnel information. This reduces the state on the P routers.

  The configuration is as shown below:

```
multicast-routing
 address-family [ipv4| ipv6]
  bgp auto-discovery [mldp | p2mp-te | ingress-replication]
   receiver-site
  !
 !
 vrf <name>
  address-family [ipv4| ipv6]
   bgp auto-discovery [mldp | p2mp-te | ingress-replication]
    receiver-site
  !
 !
!
```

- **RPF vector insertion in Global Table** Unified MPLS deployments, for example, UMMT or EPN model face issues, where some of the PEs do not support the enhancement procedures. In this case, to retain BGP-free core in the Ingress and Egress segments, the PEs send PIM Joins with RPF-proxy vector. To interoperate in such scenarios, the XR border acts as a transit node for RPF vector. This can be used in other cases of BGP-free core as well. The RPF-vector support is only for GTM and not for MVPNs (Inter-AS Option B). Support is enabled for the RPF-vector address-family being same as the Multicast Join address-family.

  The configuration is as shown below:

```
router pim
 address-family [ipv4|ipv6]
  rpf-vector
 !
!
```

Note IOS-XR supports termination of RPF vectors as well as acts as a transit router for RPF vector. The termination of RPF vectors was introduced from release 4.3.1, however, the support for acting as a transit router existed in earlier releases as well.

To configure IOS-XR to support termination of RPF vector, PIM and multicast-routing must be enabled on the loopback configured with the RPF-proxy address.

Example:

```
interface Loopback0
ipv4 address 192.0.2.1 255.255.255.0        (sample of a RPF-proxy address)
!
router pim
address-family ipv4
rpf-vector
  interface Loopback0
   enable
  !
multicast-routing
address-family ipv4
  interface Loopback0
   enable
```

RPF Vector Encoding Using IETF Standard

RPF vector is a PIM proxy that lets core routers without RPF information forward join and prune messages for external sources (for example, a MPLS-based BGP-free core, where the MPLS core router is without external routes learned from BGP). The RPF vector encoding is now compatible with the new IETF encoding. The new IETF standard encodes PIM messages using PIM Hello option 26.

Configuring RPF Vector (IETF Standard Encoding)

This example shows how to enable RPF encoding using IETF standard:

```
(config)# router pim
(config-pim-default-ipv4)# address-family ipv4
(config-pim-default-ipv4)# rpf-vector use-standard-encoding
!
(config)# multicast-routing
(config-mcast)# interface TenGigE
(config-mcast)# interface TenGigE
```

Verification

```
Router#show pim neighbor
Tue Apr 17 10:15:40.961 PDT

PIM neighbors in VRF default
Flag: B - Bidir capable, P - Proxy capable, DR - Designated Router,
      E - ECMP Redirect capable
      * indicates the neighbor created for this router

Neighbor Address             Interface              Uptime    Expires  DR pri   Flags

25.25.25.1                   TenGigE       1w3d      00:01:36 1      B P
25.25.25.2*                  TenGigE       1w3d      00:01:41 1 (DR) B P E
32.32.32.2*                  TenGigE
       1w4d      00:01:40 1      B P E
32.32.32.3                   TenGigE
       1w4d      00:01:42 1 (DR) B P
```

In the above output, you can see "P" tag on the multicast enabled interfaces.

PE-PE Ingress Replication

The ingress PE replicates a C-multicast data packet belonging to a particular MVPN and sends a copy to all or a subset of the PEs that belong to the MVPN. A copy of the packet is tunneled to a remote PE over a Unicast Tunnel to the remote PE.

IR-MDT represents a tunnel that uses IR as the forwarding method. It is usually, one IR-MDT per VRF, with multiple labeled switch paths (LSP) under the tunnel.

When PIM learns of Joins over the MDT (using either PIM or BGP C-multicast Routing), it downloads IP S,G routes to the VRF table in MRIB, with IR-MDT forwarding interfaces. Each IR-MDT forwarding interface has a LSM-ID allocated by PIM. Currently, LSM-ID is managed by mLDP and can range from 0 to 0xFFFFF (20-bits). For IR, the LSM-ID space is partitioned between mLDP and IR. For IR tunnels, the top (20th) bit is always be set, leading to a range of 0x80000 to 0xFFFFF. mLDP's limit is 0 to 0x7FFFF.

MVPN over GRE

A unicast GRE tunnel could be the accepting or forwarding interface for either a mVPN-GRE VRF route or a core route. When multicast packets arrive on the VRF interface with the intent of crossing the core, they are first encapsulated with a multicast GRE header (S,G) which are applicable to the VRF's MDT. Then, before the packets are actually forwarded , they are encapsulated in a unicast GRE header. The (S,D) in this packet are the origination and termination addresses for the unicast GRE tunnel.

GRE tunnel stitching is when both the accepting and forwarding interfaces are unicast GRE tunnels. Here, the packet has two GRE encaps. The outer encap is the unicast header for the GRE tunnel. The inner encap is the multicast GRE header for the MDT. This is called as double encap. There is a loss in terms of both bandwidth and throughput efficiency. The bandwidth efficiency loss is because 48 bytes of encap headers are being added to the original (VRF) packet. The throughput efficiency loss is the result of the processing time required to apply two encaps.

For the mVPN-GRE, if the VRF interface is a GRE tunnel, the protocol packets received from LPTS will be accompanied with the receiving unicast GRE tunnel interface and the VRF id of the VRF in which the GRE tunnel is configured. Thus VRF specific processing can be done on the packet.

Restrictions

- MVPN over GRE is supported only on ASR 9000 Enhanced Ethernet LCs.

Native Multicast

GRE tunneling provides a method to transport native multicast traffic across a non-Multicast enabled IP network. Once the multicast traffic in encapsulated with GRE, it appears as an IP packet to the core transport network.

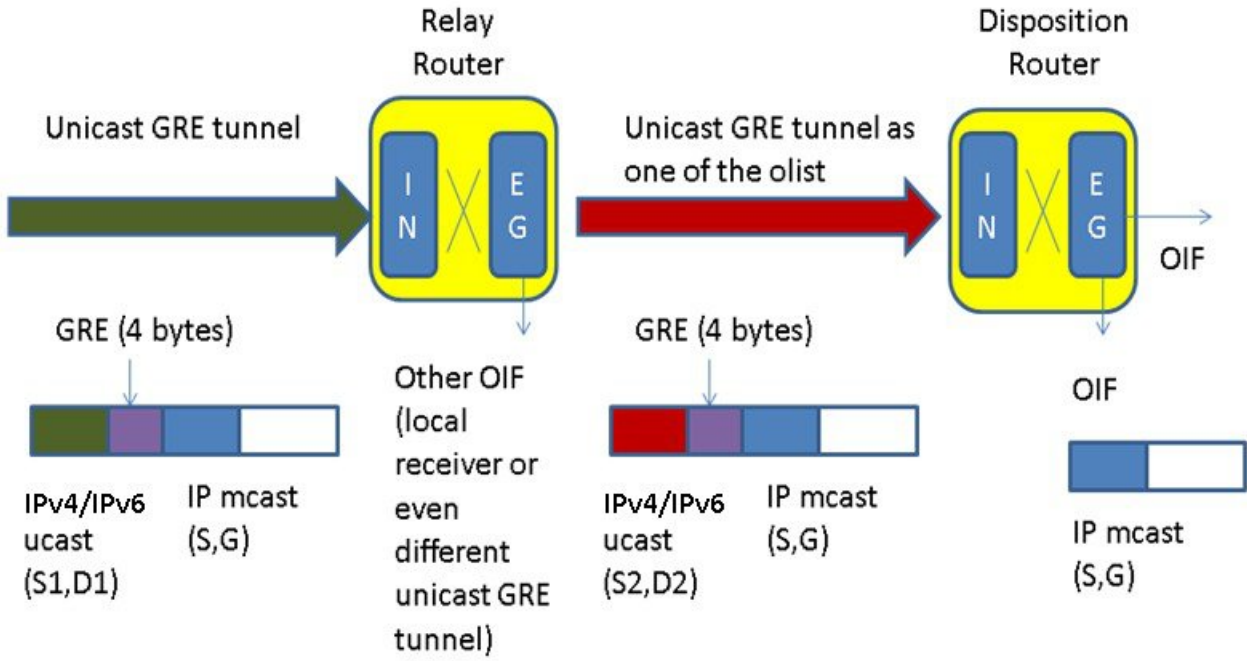
A GRE tunnel can be a forwarding interface when the router is the imposition (or encap) router for that GRE tunnel. The imposition router must prepend a unicast IPv4 header and GRE header to the multicast packet. The source and destination IPv4 addresses for the added header are determined by the user configuration of the tunnel. The newly encapsulated packet is then forwarded as a unicast packet.

When a GRE tunnel is an accepting interface for a multicast route, the router is the disposition (or decap) router for the tunnel. The outer IPv4 header and GRE header must be removed to expose the inner multicast packet. The multicast packet will then be forwarded just as any other multicast packet that arrives on a non-tunnel interface.

Forwarding behavior

Figure depicts a Unicast GRE tunnel between two routers. The imposition router has a multicast (S,G) route which has the GRE tunnel as a forwarding interface. At the disposition router, the GRE tunnel is an accepting interface for the multicast (S,G). As seen, the packet is unicast GRE encapsulated when it traverses the tunnel.

Figure 20: Unicast GRE tunnel between two routers



Note Starting with IOS XR 5.3.2 release, IPv6 traffic is supported.

GRE Limitations

Listed below are the limitations for unicast GRE tunnels:

- GRE unicast tunnel supports IPv4 encapsulation only.

Note Starting from the IOS XR 5.3.2 release, GRE unicast tunnels support IPv6 encapsulation.

- Native and mVPN traffic over underlying ECMP links are not supported.

Note Starting with IOS XR 5.3.2 release, native and mVPN traffic over underlying ECMP links, including bundles is supported.

- IPv6 multicast for GRE unicast tunnels is not supported, in releases prior to IOS XR 5.3.2.
- Transport header support is limited to IPv4.

- Path MTU discovery will not be supported over GRE tunnel interfaces. When size of the packet going over GRE tunnel interface exceeds the tunnel MTU, the microcode will punt the packet to the slow path for best effort fragmentation. Since punted packets are policed, this doesn't provide real fragmentation support. This combined with no support for path MTU discovery means that user is responsible for making sure the MTUs configured along the tunnel path are large enough to guarantee the GRE packet will not be fragmented between tunnel source and destination routers.
- No support for optional checksum, key, and sequence number fields.
- No support for nested and concatenated GRE tunnels. If packets with nested GRE header are received they will be dropped.
- No L3 features (like QoS, ACL and netflow) support for GRE tunnel interfaces. Features configured on the underlying physical interface will be applied.
- ASR9000 SIP-700 linecard unicast GRE is NOT supported on VRFs.
- Support for up to 500 GRE tunnels per system for multicast.
- Multicast forwarding over a GRE tunnel on a NV Satellite access interface is not supported.

Signaling and RPF on GRE Tunnels

Signaling will use the same mechanism when a unicast GRE tunnel terminated at an ingress linecard regardless of whether the GRE tunnel interface belongs to a VRF or not. In the case of mVPN-GRE the Primary Linecard / Primary NP mechanism must still be used for egress punts of decapsulated VRF packets.

RPF selection can be static configured via a route policy configuration. Static RPF is more preferred and expected if the RPF should be the GRE tunnel. RPF may be selected dynamically via RIB updates for the upstream router’s unicast reach-ability, although this is not preferred.

PIM Registration

PIM registration packets can be forwarded on a unicast GRE tunnel as long as the IPv4 unicast GRE interface is selected by FIB for unicast forwarding of the encapsulated PIM registration packets toward the PIM RP. In this case, the packet is essentially double encapsulated with unicast, ie, the original multicast packet is encapsulated by PIM in a unicast PIM register packet. This is then encapsulated with the unicast GRE tunnel header.

At the PIM RP, outermost unicast header will be removed and the PIM registration packets will be delivered to PIM via LPTS as in the current PIM registration packet processing. It’s advisable to avoid any MTU/TTL or ACL/QoS configuration issues that result in the registration packets getting dropped.

Configuration Example

PIM registration fails if the GRE interface MTU is greater than the physical interface. In such cases, ensure to adjust the PIM MTU, as shown in this example:

```
router pim
vrf VRF6
  address-family ipv4
   rpf topology route-policy prof3-pim-default
   hello-interval 1
   join-prune-mtu 576
   rp-address 130.0.6.1
  !
  address-family ipv6
```

```
   rpf topology route-policy prof3-pim-default
   join-prune-mtu 576
   rp-address 130::6:1
  !
!
```

Auto-RP

Auto-RP lite on PEs, Auto-RP/BSR/static-RP/ Anycast-RP with MSDP peering etc can be supported over GRE tunnels with MFIB netio chain support. It is advisable to avoid any MTU/TTL or ACL/QoS configuration issues that result in the registration packets getting dropped. Auto-RP routes will flood autp-rp packets to every multicast egress interface including IPv4 unicast GRE tunnels.

Multicast IRB

Multicast IRB provides the ability to route multicast packets between a bridge group and a routed interface using a bridge-group virtual interface (BVI). It can be enabled with multicast-routing. THE BVI is a virtual interface within the router that acts like a normal routed interface. For details about BVI, refer *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*

BV interfaces are added to the existing VRF routes and integrated with the replication slot mask. After this integration, the traffic coming from a VRF BVI is forwarded to the VPN.

Supported Bridge Port Types

- Bundles
- Satellites
- EFPs (physical, vlans, etc)

Restrictions

- Supported only on Ethernet line cards and enhanced ethernet line cards.
- Support only for IPv4

Example

The CE-PE is collapsed into 1 router (IRB) and IGMP snooping is enabled on the BVIs.

BVI type is included in a multicast VRF. After the BVI slot mask is included in the VRF route slot mask, the traffic from the VRF BVI is forwarded to the VPN/ core.

Multicast support for PW-HE interfaces

Multicast support for Pseudowire Head-end (PW-HE) interfaces is available only on the enhanced ethernet cards.

Multicast support is available under these circumstances:

- IPv4 and IPv6 multicast traffic forwarding over the L3 PW-HE interface/sub-interface. PW-HE interface type can be PW-ether (VC4 or VC5) or PW-iw (VC11). IPv6 multicast is not available on VC11.

- L3 PW-HE interfaces/sub-interfaces in global , MVPNv4 and MVPNv6 VRFs.
- L3 PW-HE interface/sub-interfaces in MVPNv4 and MVPNv6 where the core can be GRE or MLDP.
- PIM-SM, PIM-SSM (PE-CE) , MSDP and PIM Auto-RP over the PW-HE interface.
- IGMP/ MLD snooping on L2 PW-HE VC5 sub-interface.
- VC label-based load balancing.

Multicast Source Discovery Protocol

Multicast Source Discovery Protocol (MSDP) is a mechanism to connect multiple PIM sparse-mode domains. MSDP allows multicast sources for a group to be known to all rendezvous points (RPs) in different domains. Each PIM-SM domain uses its own RPs and need not depend on RPs in other domains.

An RP in a PIM-SM domain has MSDP peering relationships with MSDP-enabled routers in other domains. Each peering relationship occurs over a TCP connection, which is maintained by the underlying routing system.

MSDP speakers exchange messages called Source Active (SA) messages. When an RP learns about a local active source, typically through a PIM register message, the MSDP process encapsulates the register in an SA message and forwards the information to its peers. The message contains the source and group information for the multicast flow, as well as any encapsulated data. If a neighboring RP has local joiners for the multicast group, the RP installs the S, G route, forwards the encapsulated data contained in the SA message, and sends PIM joins back towards the source. This process describes how a multicast path can be built between domains.

Note Although you should configure BGP or Multiprotocol BGP for optimal MSDP interdomain operation, this is not considered necessary in the Cisco IOS XR Software implementation. For information about how BGP or Multiprotocol BGP may be used with MSDP, see the MSDP RPF rules listed in the Multicast Source Discovery Protocol (MSDP), Internet Engineering Task Force (IETF) Internet draft.

VRF-aware MSDP

VRF (VPN Routing and Forwarding) -aware MSDP enables MSDP to function in the VRF context. This in turn, helps the user to locate the PIM (protocol Independent Multicast) RP on the Provider Edge and use MSDP for anycast-RP.

MSDP needs to be VRF-aware when:

- Anycast-RP is deployed in an MVPN (Multicast MVPN) in such a manner that one or more PIM RPs in the anycast-RP set are located on a PE. In such a deployment, MSDP needs to operate in the VRF context on the PE.
- The PIM RP is deployed in an MVPN in such a manner that it is not on a PE and when the customer multicast routing type for the MVPN is BGP and the PEs have suppress-shared-tree-join option configured. In this scenario, there is no PE-shared tree link, so traffic may stop at the RP and it does not flow to other MVPN sites. An MSDP peering between the PIM RP and one or more PEs resolves the issue.

Multicast Nonstop Forwarding

The Cisco IOS XR Software nonstop forwarding (NSF) feature for multicast enhances high availability (HA) of multicast packet forwarding. NSF prevents hardware or software failures on the control plane from disrupting the forwarding of existing packet flows through the router.

The contents of the Multicast Forwarding Information Base (MFIB) are frozen during a control plane failure. Subsequently, PIM attempts to recover normal protocol processing and state before the neighboring routers time out the PIM hello neighbor adjacency for the problematic router. This behavior prevents the NSF-capable router from being transferred to neighbors that will otherwise detect the failure through the timed-out adjacency. Routes in MFIB are marked as stale after entering NSF, and traffic continues to be forwarded (based on those routes) until NSF completion. On completion, MRIB notifies MFIB and MFIB performs a mark-and-sweep to synchronize MFIB with the current MRIB route information.

Multicast Configuration Submodes

Cisco IOS XR Software moves control plane CLI configurations to protocol-specific submodes to provide mechanisms for enabling, disabling, and configuring multicast features on a large number of interfaces.

Cisco IOS XR Software allows you to issue most commands available under submodes as one single command string from the global or XR config mode.

For example, the **ssm** command could be executed from the PIM configuration submode like this:

```
RP/0/RSP0/CPU0:router(config)# router pim
RP/0/RSP0/CPU0:router(config-pim)# address-family ipv4
RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# ssm range
```

Alternatively, you could issue the same command from the global or XR config mode like this:

```
RP/0/RSP0/CPU0:router(config)# router pim ssm range
```

The following multicast protocol-specific submodes are available through these configuration submodes:

Multicast-Routing Configuration Submode

In Cisco IOS XR software release 3.7.2 and later, basic multicast services start automatically when the multicast PIE (asr9k-mcast-p.pie) is installed, without any explicit configuration required. The following multicast services are started automatically:

- MFWD
- MRIB
- PIM
- IGMP

Other multicast services require explicit configuration before they start. For example, to start the MSDP process, you must enter the **router msdp** command and explicitly configure it.

When you issue the **multicast-routing ipv4 or multicast-routing ipv6** command, all default multicast components (PIM, IGMP, MLD, MFWD, and MRIB) are automatically started, and the CLI prompt changes to “config-mcast-ipv4” or “config-mcast-ipv6”, indicating that you have entered multicast-routing configuration submode.

PIM Configuration Submode

When you issue the **router pim** command, the CLI prompt changes to "config-pim-ipv4," indicating that you have entered the default pim address-family configuration submode.

To enter pim address-family configuration submode for IPv6, type the **address-family ipv6** keyword together with the **router pim** command before pressing Enter.

IGMP Configuration Submode

When you issue the **router igmp** command, the CLI prompt changes to "config-igmp," indicating that you have entered IGMP configuration submode.

MLD Configuration Submode

When you issue the **router mld** command, the CLI prompt changes to "config-mld," indicating that you have entered MLD configuration submode.

MSDP Configuration Submode

When you issue the **router msdp** command, the CLI prompt changes to "config-msdp," indicating that you have entered router MSDP configuration submode.

Understanding Interface Configuration Inheritance

Cisco IOS XR Software allows you to configure commands for a large number of interfaces by applying command configuration within a multicast routing submode that could be inherited by all interfaces. To override the inheritance mechanism, you can enter interface configuration submode and explicitly enter a different command parameter.

For example, in the following configuration you could quickly specify (under router PIM configuration mode) that all existing and new PIM interfaces on your router will use the hello interval parameter of 420 seconds. However, Packet-over-SONET/SDH (POS) interface 0/1/0/1 overrides the global interface configuration and uses the hello interval time of 210 seconds.

```
RP/0/RSP0/CPU0:router(config)# router pim
RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# hello-interval 420
RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# interface pos 0/1/0/1
RP/0/RSP0/CPU0:router(config-pim-ipv4-if)# hello-interval 210
```

The following is a listing of commands (specified under the appropriate router submode) that use the inheritance mechanism:

```
router pim
  dr-priority
  hello-interval
  join-prune-interval

multicast-routing
  version
  query-interval
  query-max-response-time
  explicit-tracking
router mld
```

```
  interface all disable
  version
  query-interval
  query-max-response-time
  explicit-tracking

router msdp
  connect-source
  sa-filter
  filter-sa-request list
  remote-as
  ttl-threshold
```

Understanding Interface Configuration Inheritance Disablement

As stated elsewhere, Cisco IOS XR Software allows you to configure multiple interfaces by applying configurations within a multicast routing submode that can be inherited by all interfaces.

To override the inheritance feature on specific interfaces or on all interfaces, you can enter the address-family IPv4 or IPv6 submode of multicast routing configuration mode, and enter the **interface-inheritance disable** command together with the **interface** *type interface-path-id* or **interface all** command. This causes PIM or IGMP protocols to disallow multicast routing and to allow only multicast forwarding on those interfaces specified. However, routing can still be explicitly enabled on specified individual interfaces.

The following configuration disables multicast routing interface inheritance under PIM and IGMP generally, although forwarding enablement continues. The example shows interface enablement under IGMP of GigabitEthernet 0/6/0/3:

```
RP/0/RSP0/CPU0:router# multicast-routing address-family ipv4
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface-inheritance disable

!

!
RP/0/RSP0/CPU0:router(config)# router igmp
RP/0/RSP0/CPU0:router(config-igmp)# vrf default
RP/0/RSP0/CPU0:router(config-igmp)# interface GigabitEthernet0/6/0/0
RP/0/RSP0/CPU0:router(config-igmp-name-if)# router enable
```

For related information, see Understanding Enabling and Disabling Interfaces, on page 119

Understanding Enabling and Disabling Interfaces

When the Cisco IOS XR Software multicast routing feature is configured on your router, by default, no interfaces are enabled.

To enable multicast routing and protocols on a single interface or multiple interfaces, you must explicitly enable interfaces using the **interface** command in multicast routing configuration mode.

To set up multicast routing on all interfaces, enter the **interface all** command in multicast routing configuration mode. For any interface to be fully enabled for multicast routing, it must be enabled specifically (or be default) in multicast routing configuration mode, and it must not be disabled in the PIM and IGMP/MLD configuration modes.

For example, in the following configuration, all interfaces are explicitly configured from multicast routing configuration submode:

```
RP/0/RSP0/CPU0:router(config)# multicast-routing
RP/0/RSP0/CPU0:router(config-mcast)# interface all enable
```

To disable an interface that was globally configured from the multicast routing configuration submode, enter interface configuration submode, as illustrated in the following example:

```
RP/0/RSP0/CPU0:router(config-mcast)# interface GigabitEthernet0pos 0/1/0/0
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4-if)# disable
```

Multicast Routing Information Base

The Multicast Routing Information Base (MRIB) is a protocol-independent multicast routing table that describes a logical network in which one or more multicast routing protocols are running. The tables contain generic multicast routes installed by individual multicast routing protocols. There is an MRIB for every logical network (VPN) in which the router is configured. MRIBs do not redistribute routes among multicast routing protocols; they select the preferred multicast route from comparable ones, and they notify their clients of changes in selected attributes of any multicast route.

Multicast Forwarding Information Base

Multicast Forwarding Information Base (MFIB) is a protocol-independent multicast forwarding system that contains unique multicast forwarding entries for each source or group pair known in a given network. There is a separate MFIB for every logical network (VPN) in which the router is configured. Each MFIB entry resolves a given source or group pair to an incoming interface (IIF) for reverse forwarding (RPF) checking and an outgoing interface list (olist) for multicast forwarding.

MSDP MD5 Password Authentication

MSDP MD5 password authentication is an enhancement to support Message Digest 5 (MD5) signature protection on a TCP connection between two Multicast Source Discovery Protocol (MSDP) peers. This feature provides added security by protecting MSDP against the threat of spoofed TCP segments being introduced into the TCP connection stream.

MSDP MD5 password authentication verifies each segment sent on the TCP connection between MSDP peers. The **password clear** command is used to enable MD5 authentication for TCP connections between two MSDP peers. When MD5 authentication is enabled between two MSDP peers, each segment sent on the TCP connection between the peers is verified.

Note MSDP MD5 authentication must be configured with the same password on both MSDP peers to enable the connection between them. The 'password encrypted' command is used only for applying the stored running configuration. Once you configure the MSDP MD5 authentication, you can restore the configuration using this command.

MSDP MD5 password authentication uses an industry-standard MD5 algorithm for improved reliability and security.

Overriding VRFs in IGMP Interfaces

All unicast traffic on the user-to-network interfaces of next-generation aggregation or core networks must be mapped to a specific VRF. They must then be mapped to an MPLS VPN on the network-to-network side. This requires the configuration of a physical interface in this specific VRF.

This feature allows mapping of IGMP packets entering through a user-to-user interface to the multicast routes in the global multicast routing table. This ensures that the interface in a specific VRF can be part of the outgoing list of interfaces in the table for a multicast route.

IGMP packets entering through a non-default VRF interface in the default (global) VRF are processed, with IGMP later distributing the interface-related multicast state (route/interface) to MRIB. This occurs through the default VRF rather than through the VRF to which the interface belongs. MRIB, PIM, MSDP, and MFIB then process the multicast state for this interface through the default VRF.

When an IGMP join for a specific (S, G) is received on the configured interface, IGMP stores this information in its VRF-specific databases. But, when sending an update to MRIB, IGMP sends this route through the default VRF. MRIB then programs this (S, G) along with this interface as an OLIST member in the default multicast routing table.

Similarly, when PIM requests information about IGMP routes from MRIB, MRIB sends this update to PIM in the context of the default VRF.

This feature specifically supports:

- Mapping of IGMP requests on an interface in a non-default VRF to the default VRF multicast routing table.
- Enabling and disabling of VRF override functionality at run time.
- Routing policy configuration at the global (default) VRF level, because routing policy configuration cannot be done at the granularity of an individual interface.
- Enablement and disablement of an IGMP VRF override on all Layer- 3 and Layer- 2 interface types, including physical Ethernet, VLAN sub-interface, bundles and VLANs over bundles.
- The same scale of multicast routes and OLIST interfaces currently supported by the platform even when VRF override functionality is operational.

Restriction

IGMP VRF Override is not supported with BVI interfaces.

VRF support for MLD

MLD receives MLD joins, membership queries and membership reports under VRF. The MLD process will have LPTS entries per VRF and traffic is redirected based on the matching VRF entry to the correct interface configured under the given VRF. Support for Source-Specific Multicast(SSM) is also provided under VRF .

Layer 3 Multicast Bundle Subinterface Load Balancing

The Layer 3 (L3) Multicast Bundle Subinterface Load Balancing feature allows you statically configure hash values to gain more control for bandwidth allocation (that is to ensure there is no oversubscription) and QoS (Quality of Service).

L3 native multicast and MVPN Rosen GRE traffic, for which the L3 bundle subinterface is an OIF (Outgoing Interface), honor the **bundle load-balancing hash** configuration, and traffic egresses out of the physical member associated with the bundle hash specified in the configuration.

Benefit of Bundle Subinterface Load Balancing

Bundle subinterface load balancing associates all traffic with a bundle subinterface to a given underlying bundle member to get more control of bandwidth allocation, while still having redundancy.

Enable/Disable Bundle Subinterface Load Balancing

By default, the bundle subinterface load balancing feature is not enabled. The feature is enabled by using the **bundle load-balancing hash** command in the subinterface configuration mode. The feature is disabled if you remove or change the configuration.

Note The configuration is applicable to only a bundle subinterface.

Bundle Load Balance Auto Option

The configuration option for bundle VLANs is **bundle load-balancing hash auto**. When this option is specified, a static hash value is used instead of the normal (S,G) hash. This ensures that all the traffic that egresses out of this bundle VLAN interface, selects the same bundle member.

Bundle Load Balance Value

The bundle load balance value configuration is similar to bundle load balance auto configuration. However, instead of a generated value, you specify a hash value for use. The supplied hash value is used instead of the normal (S,G) hash. As in bundle load balance auto, this configuration too ensures that all the traffic that egresses out of this bundle VLAN interface selects the same bundle member.

Restrictions of Bundle Subinterface Load Balancing

Bundle subinterface load balancing has these restrictions:

- Applicable for only bundle subinterfaces.
- Only Cisco ASR 9000 Enhanced Ethernet Line Cards and Cisco ASR 9000 High Density 100GE Ethernet Line Cards are supported.
- Only L3 IP multicast and Rosen GRE MVPN are supported. P2MP and MLDP are not supported.

- PWHE bundles and BNG interfaces with bundles are not supported.
- Satellite and bundle over bundle for Satellite are not supported.

To Configure Bundle Subinterface Load Balance

Perform this task to statically configure the hash values for a bundle subinterface.

Note This configuration is applicable to only a bundle subinterface.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *interface-number*
3. **bundle load-balancing hash** { *hash-value* | **auto** }
4. **encapsulation dot1q** *vlan-id*
5. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface Bundle-Ether** *interface-number* **Example:** `RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 200.200`	Enters subinterface configuration mode.

Step 3	**bundle load-balancing hash** { *hash-value* | **auto** }
	Example:
	`RP/0/RSP0/CPU0:router(config-subif)# bundle load-balancing hash auto (OR)`
	`RP/0/RSP0/CPU0:router(config-subif)# bundle load-balancing hash 2`
		Specifies the hash function used for traffic forwarded over the bundle.
		When a hash value is specified, the specified hash value is used instead of the normal (S,G) hash. The range is from 1 to 64.
		When the **auto** option is specified, a static hash value is used instead of the normal (S,G) hash.

	Command or Action	Purpose
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 200`	Defines the encapsulation format as IEEE 802.1Q (dot1q), and specifies the VLAN ID (identifier). VLAN ID range is from 1 to 4094.
Step 5	**commit**	

Verify Bundle Subinterface Load Balance: Examples

The following examples show how to verify bundle subinterface load balance:

For MRIB:

```
RP/0/RSP0/CPU0:RTP-VIKING-MCAST-33# show mrib platform interface bundle-ether 2

Mon Oct  5 14:37:14.853 EDT
  ------------------------------------------------------------
  Bundle-Ether2.1 (0x2000220)
  ------------------------------------------------------------
   Bundle Interface:   Bundle-Ether2 (0x20001a0)
   Root Interface:     Bundle-Ether2 (0x20001a0)
   LAG Hash:           0x20
   RT OLE Refcount:    1
  ------------------------------------------------------------
  Bundle-Ether2.1 (0x2000220) RT OLE List
 ------------------------------------------------------------
  Route OLE on Bundle-Ether2.1 (0x2000220)
   Route:         0xe0000000:(20.1.1.2, 232.0.0.1)/32
   UL Interface:  HundredGigE0/0/0/6 (0x100)
   Bundle Member: HundredGigE0/0/0/6 (0x100)
   Raw hash:      0x53ad810d
   Intrf Handle:  0x10077d28
   Entry Handle:  0x100473e4
 ------------------------------------------------------------
```

For MFIB:

```
RP/0/RSP0/CPU0:RTP-VIKING-MCAST-33# show mfib hardware route olist location 0/0/CPU0

Source: 20.1.1.2        Group: 232.0.0.1       Mask: 64  RPF Int: BE1.1

  Route Information
  ------------------------------------------------------------------------------
  NP B  S  DC PL PR PF DR BD RI         FS     G      M      T  OC   Base
  ------------------------------------------------------------------------------
  0  F  F  F  F  F  F  F  F  0x20001e0  0x1    16902  6      0  0    0xc68c6b
  1  F  F  F  F  F  F  F  F  0x20001e0  0x1    16902  6      0  0    0xc68ce3
  2  F  F  F  F  F  F  F  F  0x20001e0  0x1    16902  6      0  0    0xc68c6b
  3  F  F  F  F  F  F  F  F  0x20001e0  0x1    16902  6      0  1    0xc68ce3
  ------------------------------------------------------------------------------

  Interface Information
  ------------------------------------------------------------------------------------------------
  NP Intf            OT  U      T  IC B  EU   IB EH   OIDX PT VRF  Base     RM(P,B)  L
  ------------------------------------------------------------------------------
```

```
3  BE2.1           REG 14     0  F  F  14   T  T/T  0    T  0    0xc68ceb 0x0,0x0 0x20
------------------------------------------------------------------------------------------

Software OLIST Information
--------------------------------
NP    SW OC    HW OC    T    SD
--------------------------------
3     1        1        0    T
--------------------------------

Virtual Interface Local Presence
-----------------------------------------------------
NP Intf            UL Intf         Bundle Parent
-----------------------------------------------------
3  Bundle-Ether2.1 Hu0/0/0/6       Bundle-Ether2
-----------------------------------------------------
```

How to Implement Multicast Routing

This section contains instructions for both building a basic multicast configuration, as well as optional tasks to help you to optimize, debug, and discover the routers in your multicast network.

Configuring PIM-SM and PIM-SSM

SUMMARY STEPS

1. **configure**
2. **multicast-routing** [**address-family** {**ipv4** | **ipv6**}]
3. **interface all enable**
4. exit
5. Use **router igmp** for IPv4 hosts or use **router mld** for IPv6
6. **version** {**1** | **2** | **3**} for IPv4 (IGMP) hosts or **version** {**1** | **2**} for IPv6 (MLD) hosts.
7. **commit**
8. **show pim** [**ipv4** | **ipv6**] **group-map** [*ip-address-name*] [**info-source**]
9. **show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **topology** [*source-ip-address* [*group-ip-address*] | **entry-flag** *flag* | **interface-flag** | **summary**] [**route-count**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode. • The following multicast processes are started: MRIB, MFWD, PIM, and IGMP. • For IPv4, IGMP version 3 is enabled by default.

	Command or Action	Purpose
Step 3	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces.
Step 4	exit **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# exit`	Exits multicast routing configuration mode, and returns the router to the source configuration mode.

Step 5	Use **router igmp** for IPv4 hosts or use **router mld** for IPv6
	Example:
	`RP/0/RSP0/CPU0:router(config)# router igmp`
	`RP/0/RSP0/CPU0:router(config)# router mld`
		(Optional) Enters router IGMP configuration mode (for IPv4 hosts), or enters router MLD configuration mode (for IPv6 hosts).

Step 6	**version** {**1** | **2** | **3**} for IPv4 (IGMP) hosts or **version** {**1** | **2**} for IPv6 (MLD) hosts.
	Example:
	`RP/0/RSP0/CPU0:router(config-igmp)# version 3`
	`RP/0/RSP0/CPU0:router(config-mld)# version 2`
		(Optional) Selects the IGMP or MLD version that the router interface uses.
		• The version range for IGMP is 1-3; the range for MLD is 1-2.
		• The default for IGMP is version 3; the default for MLD is version 1.
		• Host receivers must support IGMPv3 for PIM-SSM operation.
		• If this command is configured in router IGMP or router MLD configuration mode, parameters are inherited by all new and existing interfaces. You can override these parameters on individual interfaces from interface configuration mode.

Step 7	**commit**	
Step 8	**show pim** [**ipv4** | **ipv6**] **group-map** [*ip-address-name*] [**info-source**] **Example:** `RP/0/RSP0/CPU0:router# show pim ipv4 group-map`	(Optional) Displays group-to-PIM mode mapping.
Step 9	**show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **topology** [*source-ip-address* [*group-ip-address*] | **entry-flag** *flag* | **interface-flag** | **summary**] [**route-count**] **Example:**	(Optional) Displays PIM topology table information for a specific group or all groups.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show pim topology`	

Configuring PIM-SSM for Use in a Legacy Multicast Deployment

Deploying PIM-SSM in legacy multicast-enabled networks can be problematic, because it requires changes to the multicast group management protocols used on the various devices attached to the network. Host, routers, and switches must all be upgraded in such cases.

To support legacy hosts and switches in a PIM-SSM deployment,Cisco ASR 9000 Series Routers offer a configurable mapping feature. Legacy group membership reports for groups in the SSM group range are mapped to a set of sources providing service for that set of (S,G) channels.

This configuration consists of two tasks:

Restrictions for PIM-SSM Mapping

PIM-SSM mapping does not modify the SSM group range. Instead, the legacy devices must report group membership for desired groups in the SSM group range .

Configuring a Set of Access Control Lists for Static SSM Mapping

This tasks configures a set of access control lists (ACLs) where each ACL describes a set of SSM groups to be mapped to one or more sources.

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *acl-name*
3. [*sequence-number*] **permit** *source* [*source-wildcard*]
4. Repeat Step 3, on page 128 to add more entries to the ACL.
5. Repeat Step 2, on page 127 through Step 4, on page 128 until you have entered all the ACLs you want to be part of the set.
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**ipv4 access-list** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list mc3`	Enters IPv4 ACL configuration submode and creates a name for an IPv4 access list.

	Command or Action	Purpose
Step 3	[*sequence-number*] **permit** *source* [*source-wildcard*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit 1 host 232.1.1.2 any`	Sets conditions for the access list to recognize the source as part of the specified access list set, in which each ACL describes a set of SSM groups to be mapped.
Step 4	Repeat Step 3, on page 128 to add more entries to the ACL.	—
Step 5	Repeat Step 2, on page 127 through Step 4, on page 128 until you have entered all the ACLs you want to be part of the set.	—
Step 6	**commit**	

Configuring a Set of Sources for SSM Mapping

This task consists of configuring a set of sources mapped by SSM groups, as described by access lists (ACLs).

SUMMARY STEPS

1. **configure**
2. **router igmp** [**vrf** *vrf-name*]
3. **ssm map static** *source-address access-list*
4. Repeat Step 3, on page 128 as many times as you have source addresses to include in the set for SSM mapping.
5. **commit**
6. **show igmp** [**vrf** *vrf-name*] **ssm map** [*group-address*][*detail*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router igmp** [**vrf** *vrf-name*] **Example:** `RP/0/RSP0/CPU0:router(config)# router igmp vrf vrf20`	Enters router IGMP configuration mode.
Step 3	**ssm map static** *source-address access-list* **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# ssm map static 232.1.1.1 mc2`	Configures a source as part of a set of sources that map SSM groups described by the specified access list.

	Command or Action	Purpose
Step 4	Repeat Step 3, on page 128 as many times as you have source addresses to include in the set for SSM mapping.	—
Step 5	**commit**	

Step 6	**show igmp** [**vrf** *vrf-name*] **ssm map** [*group-address*][*detail*]
	Example:
	`RP/0/RSP0/CPU0:router# show igmp vrf vrf20 ssm map 232.1.1.1`
	`232.1.1.1 is static with 1 source`
	or
	`RP/0/RSP0/CPU0:router# show igmp vrf vrf20 ssm map`
	`232.1.1.0 is static with 3 sources`
	`232.1.1.1 is static with 1 source`
		(Optional) Queries the mapping state.
		• When you provide one address for mapping, you receive the state for that address alone.
		• When you provide no address for mapping, you receive the state for all sources.

Configuring the DNS-based SSM Mapping

Perform this task to configure the last hop router to perform DNS look-ups to learn the IP addresses of sources sending to a group.

Before you begin

- Enable IP multicast routing, enable PIM sparse mode, and configure SSM before performing this task. For more information, see Configuring PIM-SM and PIM-SSM, on page 125 and Configuring PIM-SSM for Use in a Legacy Multicast Deployment, on page 127.
- Before you can configure and use SSM mapping with DNS lookups, you need to be able to add records to a running DNS server. If you do not already have a DNS server running, you need to install one. The Cisco IOS XR software does not provide for DNS server functionality.

SUMMARY STEPS

1. **configure**
2. *(Optional)* **domain multicast** *domain-prefix*
3. **domain name-server** *server-address*
4. **router igmp**
5. **ssm map query dns**
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	*(Optional)* **domain multicast** *domain-prefix* **Example:** `RP/0/RSP0/CPU0:router(config)# domain multicast cisco.com`	Specifies the domain prefix used for DNS-based SSM mapping.
Step 3	**domain name-server** *server-address* **Example:** `RP/0/RSP0/CPU0:router(config)# domain name-server 10.10.10.1`	Specifies the IPv4 or IPv6 address of the domain name server to use for name and address resolution. Repeat this step to specify additional domain name servers.
Step 4	**router igmp** **Example:** `RP/0/RSP0/CPU0:router(config)# router igmp`	Enters router IGMP configuration mode.
Step 5	**ssm map query dns** **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# ssm map query dns`	Enables DNS-based ssm mapping.
Step 6	**commit**	

Configuring a Static RP and Allowing Backward Compatibility

When PIM is configured in sparse mode, you must choose one or more routers to operate as a rendezvous point (RP) for a multicast group. An RP is a single common root placed at a chosen point of a shared distribution tree. An RP can either be configured statically in each router, or learned through Auto-RP or BSR.

This task configures a static RP. For more information about RPs, see the Rendezvous Points, on page 20. For configuration information for Auto-RP, see the Configuring Auto-RP to Automate Group-to-RP Mappings, on page 132.

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family** {**ipv4** | **ipv6**}]
3. **rp-address** *ip-address* [*group-access-list*] [**bidir**] [**override**]
4. **old-register-checksum**
5. **exit**
6. {**ipv4** | **ipv6**} **access-list** *name*
7. [*sequence-number*] **permit** *source* [*source-wildcard*]
8. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters PIM configuration mode, or PIM address-family configuration submode.
Step 3	**rp-address** *ip-address* [*group-access-list*] [**bidir**] [**override**] **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# rp-address 172.16.6.22 rp-access`	Assigns an RP to multicast groups. • If you specify a **group-access-list-number** value, you must configure that access list using the **ipv4 access-list** command.
Step 4	**old-register-checksum** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4)# old-register-checksum`	(Optional) Allows backward compatibility on the RP that uses old register checksum methodology.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4)# exit`	Exits PIM configuration mode, and returns the router to the source configuration mode.
Step 6	{**ipv4** | **ipv6**} **access-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list rp-access`	(Optional) Enters access list configuration mode and configures the RP access list. • The access list called “rp-access” permits multicast group 239.1.1.0 0.0.255.255.
Step 7	[*sequence-number*] **permit** *source* [*source-wildcard*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit 239.1.1.0 0.0.255.255`	(Optional) Permits multicast group 239.1.1.0 0.0.255.255 for the “rp-access” list. **Tip** The commands in Step 6, on page 131 and Step 7, on page 131 can be combined in one command string like this: ipv4 access-list rp-access permit 239.1.1.0 0.0.255.255.
Step 8	**commit**	

Configuring Auto-RP to Automate Group-to-RP Mappings

This task configures the Auto-RP mechanism to automate the distribution of group-to-RP mappings in your network. In a network running Auto-RP, at least one router must operate as an RP candidate and another router must operate as an RP mapping agent. The VRF interface on Cisco ASR 9000 Series Routers cannot be an auto-rp candidate- rp.

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family ipv4**]
3. **auto-rp candidate-rp** *type instance* **scope** *ttl-value* [**group-list** *access-list-name*] [**interval** *seconds*] **bidir**
4. **auto-rp mapping-agent** *type number* **scope** *ttl-value* [**interval** *seconds*]
5. **exit**
6. **ipv4 access-list** *name*
7. [*sequence-number*] **permit** *source* [*source-wildcard*]
8. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family ipv4**] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters PIM configuration mode, or PIM address-family configuration submode.
Step 3	**auto-rp candidate-rp** *type instance* **scope** *ttl-value* [**group-list** *access-list-name*] [**interval** *seconds*] **bidir** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4)# auto-rp candidate-rp GigabitEthernet0/1/0/1 scope 31 group-list 2 bidir`	Configures an RP candidate that sends messages to the CISCO-RP-ANNOUNCE multicast group (224.0.1.39). • This example sends RP announcements out all PIM-enabled interfaces for a maximum of 31 hops. The IP address by which the router wants to be identified as an RP is the IP address associated with GigabitEthernet interface 0/1/0/1. • Access list 2 designates the groups this router serves as RP. • If you specify **group-list**, you must configure the optional **access-list** command.
Step 4	**auto-rp mapping-agent** *type number* **scope** *ttl-value* [**interval** *seconds*] **Example:**	Configures the router to be an RP mapping agent on a specified interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pim-ipv4)# auto-rp mapping-agent GigabitEthernet0/1/0/1 scope 20`	• After the router is configured as an RP mapping agent and determines the RP-to-group mappings through the CISCO-RP-ANNOUNCE (224.0.1.39) group, the router sends the mappings in an Auto-RP discovery message to the well-known group CISCO-RP-DISCOVERY (224.0.1.40). • A PIM DR listens to this well-known group to determine which RP to use. • This example limits Auto-RP discovery messages to 20 hops.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4)# exit`	Exits PIM configuration mode and returns the router to the source configuration mode.
Step 6	**ipv4 access-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list 2`	(Optional) Defines the RP access list.
Step 7	[*sequence-number*] **permit** *source* [*source-wildcard*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit 239.1.1.1 0.0.0.0`	(Optional) Permits multicast group 239.1.1.1 for the RP access list. **Tip** The commands in Step 6, on page 133 and Step 7, on page 133 can be combined in one command string and entered from the global or XR config mode like this: ipv4 access-list rp-access permit 239.1.1.1 0.0.0.0
Step 8	**commit**	

Configuring the Bootstrap Router

This task configures one or more candidate bootstrap routers (BSRs) and a BSR mapping agent. This task also connects and locates the candidate BSRs in the backbone portion of the network.

For more information about BSR, see the PIM Bootstrap Router, on page 22.

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family** {**ipv4** | **ipv6**}]
3. **bsr candidate-bsr** *ip-address* [**hash-mask-len** *length*] [**priority** *value*]
4. **bsr candidate-rp** *ip-address* [**group-list** *access-list* **interval** *seconds*] [**priority** *value*] **bidir**
5. **interface** *type interface-path-id*

6. **bsr-border**
7. **exit**
8. **exit**
9. {**ipv4** | **ipv6**} **access-list** *name*
10. Do one of the following:
    - [*sequence-number*] **permit** *source* [*source-wildcard*]
    - [*sequence-number*] **permit** *source-prefix dest-prefix*
11. **commit**
12. **clear pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr**
13. **show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr candidate-rp**
14. **show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr election**
15. **show pim** [**vrf** *vrf-name*][**ipv4** | **ipv6**] **bsr rp-cache**
16. **show pim** [**vrf** *vrf-name*][**ipv4** | **ipv6**] **group-map** [*ip-address-name*] [**info-source**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters PIM configuration mode, or address-family configuration submode.
Step 3	**bsr candidate-bsr** *ip-address* [**hash-mask-len** *length*] [**priority** *value*] **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# bsr candidate-bsr 10.0.0.1 hash-mask-len 30`	Configures the router to announce its candidacy as a BSR.
Step 4	**bsr candidate-rp** *ip-address* [**group-list** *access-list* **interval** *seconds*] [**priority** *value*] **bidir** **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# bsr candidate-rp 172.16.0.0 group-list 4 bidir`	Configures the router to advertise itself as a PIM Version 2 candidate RP to the BSR. • See Step 9, on page 135 for group list 4 configuration.
Step 5	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)#`	(Optional) Enters interface configuration mode for the PIM protocol.

	Command or Action	Purpose
	`interface GigE 0/1/0/0`	
Step 6	**bsr-border** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-if)# bsr-border`	(Optional) Stops the forwarding of bootstrap router (BSR) messages on a Protocol Independent Multicast (PIM) router interface.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-if)# exit`	(Optional) Exits PIM interface configuration mode, and returns the router to PIM configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# exit`	Exits PIM configuration mode.
Step 9	{**ipv4** | **ipv6**} **access-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list 4`	(Optional) Defines the candidate group list to the BSR. • Access list number 4 specifies the group prefix associated with the candidate RP address 172.16.0.0. (See Step 4, on page 134). • This RP is responsible for the groups with the prefix 239.
Step 10	Do one of the following: • [*sequence-number*] **permit** *source* [*source-wildcard*] • [*sequence-number*] **permit** *source-prefix dest-prefix* **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit 239.1.1.1 0.255.255.255`	(Optional) Permits multicast group 239.1.1.1 for the candidate group list. **Tip** The commands in Step 6, on page 135 and Step 7, on page 135 can be combined in one command string and entered from global configuration mode like this: ipv4 access-list rp-access permit 239.1.1.1 0.255.255.255
Step 11	**commit**	
Step 12	**clear pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr** **Example:** `RP/0/RSP0/CPU0:router# clear pim bsr`	(Optional) Clears BSR entries from the PIM RP group mapping cache.
Step 13	**show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr candidate-rp** **Example:**	(Optional) Displays PIM candidate RP information for the BSR.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show pim bsr candidate-rp`	
Step 14	**show pim** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **bsr election** **Example:** `RP/0/RSP0/CPU0:router# show pim bsr election`	(Optional) Displays PIM candidate election information for the BSR.
Step 15	**show pim** [**vrf** *vrf-name*][**ipv4** | **ipv6**] **bsr rp-cache** **Example:** `RP/0/RSP0/CPU0:router# show pim bsr rp-cache`	(Optional) Displays PIM RP cache information for the BSR.
Step 16	**show pim** [**vrf** *vrf-name*][**ipv4** | **ipv6**] **group-map** [*ip-address-name*] [**info-source**] **Example:** `RP/0/RSP0/CPU0:router# show pim ipv4 group-map`	(Optional) Displays group-to-PIM mode mapping.

Calculating Rates per Route

This procedure enables multicast hardware forward-rate counters on a per-VRF-family basis.

SUMMARY STEPS

1. **configure**
2. **multicast-routing** [**vrf** *vrf-name*] [**address-family** {**ipv4** | **ipv6**}]
3. **rate-per-route**
4. **interface** {*type interface-path-id* | **all**} **enable**
5. Do one of the following:
   - **accounting per-prefix**
   - **accounting per-prefix forward-only**
6. **commit**
7. **show mfib** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **route** [**rate** | **statistics**] [***** | *source-address*] [*group-address* [/*prefix-length*] [**detail** | **old-output**] | **summary**] [**location** *node-id*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	**multicast-routing** [**vrf** *vrf-name*] [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing address-family ipv4`	Enters multicast routing configuration mode. • The following multicast processes are started: MRIB, MFWD, PIM, and IGMP. • For IPv4, IGMP version 3 is enabled by default.
Step 3	**rate-per-route** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# rate-per-route`	Enables a per (S,G) rate calculation for a particular route.

Step 4	**interface** {*type interface-path-id* | **all**} **enable**
	Example:
	`RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`
	or
	`RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface FastEthernet0/3/3/1 enable`
		Enables multicast routing on all interfaces.

Step 5	Do one of the following: • **accounting per-prefix** • **accounting per-prefix forward-only** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv)# accounting per-prefix`	• Enables per-prefix counters present in hardware, assigning every existing and new (S, G) route forward, punt, and drop counters on the ingress route and forward and punt counters on the egress route. The (*, G) routes are assigned a single counter. • Enables per-prefix counters present in hardware accounting per-prefix—Enables three counters on ingress (forward, punt and drop, and two on egress (forward and punt) on every existing and new (S, G) route. The (*, G) routes are assigned a single counter. • accounting per-prefix forward-only—Enables one counter on ingress and one on egress in hardware to conserve hardware statistics resources. (Recommended for configuration of multicast VPN routing or for any line card that has a route-intensive configuration.)
Step 6	**commit**	
Step 7	**show mfib** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **route** [**rate** | **statistics**] [***** | *source-address*] [*group-address* [*/prefix-length*] [**detail** | **old-output**] | **summary**] [**location** *node-id*]	Displays route entries in the Multicast Forwarding Information Base (MFIB) table. • When the **rate** keyword is used with the *source-* and *group-address*, the command displays the cumulative

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router# show mfib vrf 12 route statistics location 0/1/cpU0`	rates per route for all line cards in the Multicast Forwarding Information Base (MFIB) table. • When the **statistics** keyword is used, the command displays the rate per route for one line card in the Multicast Forwarding Information Base (MFIB) table.

Configuring Multicast Nonstop Forwarding

This task configures the nonstop forwarding (NSF) feature for multicast packet forwarding for the purpose of alleviating network failures, or software upgrades and downgrades.

Although we strongly recommend that you use the NSF lifetime default values, the optional Step 3, on page 139 through Step 6, on page 139 allow you to modify the NSF timeout values for Protocol Independent Multicast (PIM) and Internet Group Management Protocol (IGMP) or Multicast Listener Discovery (MLD). Use these commands when PIM and IGMP (or MLD) are configured with nondefault interval or query intervals for join and prune operations.

Generally, configure the IGMP NSF and PIM NSF lifetime values to equal or exceed the query or join query interval. For example, if you set the IGMP query interval to 120 seconds, set the IGMP NSF lifetime to 120 seconds (or greater).

If the Cisco IOS XR Software control plane does not converge and reconnect after NSF is enabled on your router, multicast packet forwarding continues for up to 15 minutes, then packet forwarding stops.

Before you begin

For NSF to operate in your multicast network, you must also enable NSF for the unicast protocols (such as IS-IS, OSPF, and BGP) that PIM relies on for Reverse Path Forwarding (RPF) information. See the appropriate configuration modules to learn how to configure NSF for unicast protocols.

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family** {**ipv4** | **ipv6**}]
3. **nsf lifetime** *seconds*
4. **exit**
5. **router** {**igmp** | **mld**}
6. **nsf lifetime** *seconds*
7. **commit**
8. **show** {**igmp nsf**
9. **show mfib** [**ipv4** | **ipv6**] **nsf** [**location** *node-id*]
10. **show mrib** [**ipv4** | **ipv6**] **nsf**
11. **show pim** [**ipv4** | **ipv6**] **nsf**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim address-family ipv4`	(Optional) Enters PIM address-family configuration submode.
Step 3	**nsf lifetime** *seconds* **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# nsf lifetime 30`	(Optional) Configures the NSF timeout value for multicast forwarding route entries under the PIM process. **Note** If you configure the PIM hello interval to a nondefault value, configure the PIM NSF lifetime to a value less than the hello hold time. Typically the value of the hold-time field is 3.5 times the interval time value, or 120 seconds if the PIM hello interval time is 30 seconds.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv4)# exit`	(Optional) Exits PIM configuration mode and returns the router to the source configuration mode.
Step 5	**router** {**igmp** | **mld**} **Example:** `RP/0/RSP0/CPU0:router(config)# router igmp`	(Optional) Enters router IGMP or MLD configuration mode.
Step 6	**nsf lifetime** *seconds* **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# nsf lifetime 30`	(Optional) Configures the NSF timeout value for multicast forwarding route entries under the IGMP process.
Step 7	**commit**	
Step 8	**show** {**igmp nsf** **Example:** `RP/0/RSP0/CPU0:router# show igmp nsf`	(Optional) Displays the state of NSF operation in IGMP.

	Command or Action	Purpose
Step 9	**show mfib** [**ipv4** | **ipv6**] **nsf** [**location** *node-id*] **Example:** `RP/0/RSP0/CPU0:router# show mfib nsf`	(Optional) Displays the state of NSF operation for the MFIB line cards.
Step 10	**show mrib** [**ipv4** | **ipv6**] **nsf** **Example:** `RP/0/RSP0/CPU0:router# show mrib nsf`	(Optional) Displays the state of NSF operation in the MRIB.
Step 11	**show pim** [**ipv4** | **ipv6**] **nsf** **Example:** `RP/0/RSP0/CPU0:router# show pim nsf`	(Optional) Displays the state of NSF operation for PIM.

Configuring Multicast VPN

- Enabling a VPN for Multicast Routing, on page 141 (required)
- "Configuring BGP to Advertise VRF Routes for Multicast VPN from PE to PE" (required)

  See the module "Implementing BGP on Cisco IOS XR Software in *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- Configuring an MDT Address Family Session in BGP as a PE-to- PE Protocol (optional for PIM-SM MDT groups; required for PIM-SSM MDT groups)

  See the "Configuring an MDT Address Family Session in BGP" section in *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- Configuring a provider-edge-to-customer-edge protocol (optional)

  See the "Configuring BGP as a PE-CE Protocol," "Configuring OSPF as a PE-to-CE Protocol," and "Configuring EIGRP as a PE-to CE Protocol" sections in *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- Specifying the PIM VRF Instance, on page 144 (optional)

Prerequisites for Multicast VPN

- PIM and multicast forwarding must be configured on all interfaces used by multicast traffic. In an MVPN, you must enable PIM and multicast forwarding for the following interfaces:
  - Physical interface on a provider edge (PE) router that is connected to the backbone.
  - Interface used for BGP peering source address.
  - Any interfaces configured as PIM rendezvous points.

Note PIM and multicast forwarding are enabled in multicast routing configuration mode. No additional configuration is required in router pim mode to enable the PIM protocol.

- Interfaces in the VPN intended for use in forwarding multicast traffic must be enabled for PIM and multicast forwarding.
- BGP should already be configured and operational on all routers that are sending or receiving multicast traffic.
- To enable MVPN, you must include a VPN IPv4 address-family (AFI) in your BGP configuration. See Restrictions for Multicast VPN for Multicast Routing, on page 141. (See also the "Enabling BGP Routing" section in Cisco IOS XR Routing Configuration Guide.)
- All PE routers in the multicast domain must be running a Cisco IOS XR Software image that supports MVPN.
- Multicast forwarding must be configured for the global IPv4 address family.
- Each multicast SM VRF domain must have an associated PIM rendezvous point (RP) definition. Using Auto-RP and the bootstrap router (BSR), you may configure RP services in the MVPN on the customer-edge (CE) device because the MVPN learns about the RP dynamically. The VRF interface can be used as a listener on the PE device.

  To enable static RP services, you must configure every device in the domain for this purpose.

Restrictions for Multicast VPN for Multicast Routing

- Configuration of the MDT source on a per-VRF basis is only supported on IPv4.
- The MDT group address should be the same for both the address families in the same VRF.
- The nV satellite access interfaces are expected to tbe deployed as the access or edge interfaces and hence do not support functionalities of core interfaces on multicast topologies.

Enabling a VPN for Multicast Routing

This task enables multicast VPN routing for IPv4.

The MDT group address is used by provider edge (PE) routers to form a virtual PIM "neighborship" for the MDT. This enables the PEs to communicate with other PEs in the VRF as if they shared a LAN.

When sending customer VRF traffic, PEs encapsulate the traffic in their own (S,G) state, where the G is the MDT group address, and the S is the MDT source for the PE. By joining the (S,G) MDT of its PE neighbors, a PE router is able to receive the encapsulated multicast traffic for that VRF.

Although the VRF itself may have many multicast sources sending to many groups, the provider network needs only to install state for one group per VRF, in other words, the MDT group.

SUMMARY STEPS

1. **configure**
2. **multicast-routing**

3. **address-family ipv4**
4. **nsf**
5. **mdt source** *type interface-path-id*
6. **interface all enable**
7. **vrf** *vrf-name*
8. **address-family** {**ipv4**}]
9. **mdt default** *mdt-group-address*
10. **mdt data** *mdt-group-address/prefix-length* **threshold** *threshold acl-name*
11. **mdt mtu** *size*
12. **interface all enable**
13. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode.
Step 3	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 address-family submode.
Step 4	**nsf** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# nsf`	Specifies that nonstop forwarding (NSF) maintains the forwarding state in case of a disruption to a multicast process.
Step 5	**mdt source** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt source GigE 0/1/0/0`	Specifies the MDT source address. **Note** The MDT source interface name should be the same as the one used for BGP.

	Command or Action	Purpose
Step 6	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces. **Caution** To avoid any possibility of a reverse-path forwarding (RPF) failure, you should proactively enable any interfaces that might possibly carry multicast traffic.
Step 7	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-)# vrf vrf_A`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode.
Step 8	**address-family** {**ipv4**}]	Specifies the virtual routing and forwarding instance for the ipv4 address family.
Step 9	**mdt default** *mdt-group-address* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf_A-ipv4)# mdt default 239.23.2.1`	Specifies the multicast distribution tree (MDT) default group address.
Step 10	**mdt data** *mdt-group-address/prefix-length* **threshold** *threshold acl-name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf_A-ipv4)# mdt data 239.23.3.0/24 threshold 1200 acl-A`	(IPv4 MVPN configuration only) Specifies the multicast group address range to be used for data MDT traffic. **Note** This group range should not overlap the MDT default group. This is an optional command. The default threshold beyond which traffic is sent using a data MDT group is 1 kbps. However, you may configure a higher threshold, if desired. You may also, optionally, configure an access list to limit the number of groups to be tunneled through a data MDT group. Traffic from groups not on the access-list continues to be tunneled using the default MDT group.
Step 11	**mdt mtu** *size* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt mtu 1550`	This is an optional step. Specifies the MTU size. It is recommended to configure a high value, to accommodate the maximum multicast packet size. **Note** The default MTU for PIM/GRE MDT is 1376 and the default value for mLDP/P2MP-TE MDT is 9000 for Multicast VPN.

	Command or Action	Purpose
Step 12	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces.
Step 13	**commit**	

Specifying the PIM VRF Instance

If you are configuring Protocol Independent Multicast in sparse mode (PIM-SM) in the MVPN, you may also need to configure a rendezvous point (RP). This task specifies the optional PIM VPN instance.

SUMMARY STEPS

1. **configure**
2. **router pim vrf** *vrf-name* **address-family** {**ipv4** | **ipv6**}
3. **rp-address** *ip-address* [*group-access-list-name*] [**bidir**] [**override**]
4. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim vrf** *vrf-name* **address-family** {**ipv4**	**ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config)# router pim vrf vrf_A address-family ipv4`	Enters PIM address-family configuration submode and configures the PIM VRF for either an IPv4 or IPv6 address family.
Step 3	**rp-address** *ip-address* [*group-access-list-name*] [**bidir**] [**override**] **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf_A-ipv4)# rp-address 10.0.0.0`	Configures the PIM rendezvous point (RP) address: • **group-access-list-name** = Specifies an access list of groups to be mapped to a given RP. • **bidir** = Specifies a bidirectional RP. • **override** = Specifies that a static RP configuration should override auto-RP and the bootstrap router (BSR).
Step 4	**commit**	

Specifying the IGMP VRF Instance

SUMMARY STEPS

1. **configure**
2. **router igmp**
3. **vrf** *vrf-name*
4. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router igmp** **Example:** `RP/0/RSP0/CPU0:router(config)# router igmp`	Enters IGMP configuration mode.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# vrf vrf_B`	Configures a VRF instance.
Step 4	**commit**	

Configuring the MDT Source per VRF

This optional feature lets you change the default routing mechanism in a multicast VPN network topology, which routes all unicast traffic through a BGP peering loopback configured on a default VRF. Instead, you may configure a loopback that allows you to specify the MDT source using a specific VRF, as opposed to the default VRF. This overrides the current behavior and updates BGP as part of a MDT group. BGP then modifies the source and connector attributes in the MDT SAFI and VPN IPv4 updates.

For VRFs on which the MDT source is not configured, the MDT source for the default VRF is applied. Also, when the MDT source on a VRF is unconfigured, the configuration of the MDT source default VRF takes effect.

Note In the configuration below, the default VRF does not require explicit reference in Step 5.

SUMMARY STEPS

1. **configure**
2. **multicast-routing**

3. **address-family** [**ipv4** | **ipv6**]
4. **mdt source loopback 0**
5. **exit**
6. **vrf 101**
7. **address-family ipv4**
8. **mdt source loopback 1**
9. Repeat the steps 6 to 8, as many times as needed to create other VRFs.
10. **commit**
11. **show pim vrf all mdt interface**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**multicast-routing**
	Example:
	`RP/0/RSP0/CPU0:router(config)# multicast-routing`
	`RP/0/RSP0/CPU0:router(config-mcast)#`
		Enables IP multicast routing and forwarding.

Step 3	**address-family** [**ipv4**	**ipv6**]
		Example:
		`RP/0/RSP0/CPU0:router(config-mcast)#`
		`address-family ipv4`
			Enters ipv4 (or ipv6) address-family submode.

Step 4	**mdt source loopback 0**
	Example:
	`RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)#`
	` mdt source loopback 0`
		Configures the interface used to set the MDT source address for MVPN, using the default VRF.

Step 5	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)#`
	` exit`
		Exits from the current mode.

Step 6	**vrf 101** **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# vrf 101`	

	Command or Action	Purpose
Step 7	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-101)# address-family ipv4`	Enters the IPv4 address-family mode.
Step 8	**mdt source loopback 1** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-101-ipv4)# mdt source loopback 1`	Configures the interface used to set the MDT source address for MVPN.
Step 9	Repeat the steps 6 to 8, as many times as needed to create other VRFs. **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# vrf 102 mdt source loopback 2`	—
Step 10	**commit**	

Step 11	**show pim vrf all mdt interface**
	Example:
	`RP/0/RSP0/CPU0:router# show pim vrf all mdt interface`
	`GroupAddress Interface Source Vrf`
	`239.0.0.239 mdtVRF_NAME Loopback1 VRF_NAME`
		To verify the MDT source per VRF configuration, use the **show pim vrf all mdt interface** command.

Configuring Label Switched Multicast

Deployment of an LSM MLDP-based MVPN involves configuring a default MDT and one or more data MDTs. A static default MDT is established for each multicast domain. The default MDT defines the path used by PE routers to send multicast data and control messages to other PE routers in the multicast domain. A default MDT is created in the core network using a single MP2MP LSP.

An LSP MLDP-based MVPN also supports dynamic creation of the data MDTs for high-bandwidth transmission. For high-rate data sources, a data MDT is created using the P2MP LSPs to off-load the traffic from the default MDT to avoid unnecessary waste of bandwidth to PEs that are not part of the stream. You can configure MLDP MVPN for both the intranet or extranet. This configuration section covers the rosen based MLDP profile. For configuration examples of other MLDP profiles, see Configuring LSM based MLDP: Examples, on page 264.

Note Before configuring MLDP based MVPN, ensure that the MPLS is enabled on the core facing interface. For information in MPLS configuration, see Cisco IOS XR MPLS Configuration Guide. Also, ensure that BGP and any interior gateway protocol (OSPF or ISIS) is enabled on the core router. For more information on BGP and route-policy configuration, see Cisco IOS XR Routing Configuration Guide.

Perform this task to configure label switched multicast:

SUMMARY STEPS

1. **configure**
2. **mpls ldp mldp**
3. **root**
4. **vrf** *vrf_name*
5. **vpn id** *vpn-id*
6. **address-family** [**ipv4** | **ipv6**] **unicast**
7. **import route-target** [*xx.yy.nn* | *as-number:nn* | *ip-address:nn*]
8. **export route-target** [*xx.yy.nn* | *as-number:nn* | *ip-address:nn*]
9. **root**
10. **multicast-routing vrf** *vrf_name*
11. **address-family** [**ipv4** | **ipv6**]
12. **mdt default mldp ipv4** *root-node*
13. **mdt data** *mdt-group-address* **threshold** *value*
14. **root**
15. **router bgp** *as-number* **vrf** *vrf-name*
16. **rd** *route-distinguisher*
17. **address-family ipv4 mdt**
18. **address-family vpnv4 unicast**
19. **root**
20. **router pim**
21. **vrf** *vrf_name*
22. **address-family** [**ipv4** | **ipv6**]
23. **rpf topology route-policy** *route_policy_name*
24. **root**
25. **route-policy** *route_policy_name*
26. **set core-tree** *tree_type*
27. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	**mpls ldp mldp** **Example:** `RP/0/RSP0/CPU0:router(config)# mpls ldp mldp`	Enables MPLS MLDP support.
Step 3	**root** **Example:** `RP/0/RSP0/CPU0:router(config-ldp-mldp)# root`	Takes the user to the global configuration level.
Step 4	**vrf** *vrf_name* **Example:** `RP/0/RSP0/CPU0:router(config)# vrf vrf1`	Configures a VRF instance. The vrf-name argument is the name assigned to a VRF.
Step 5	**vpn id** *vpn-id* **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# vpn id 1:1`	Sets or updates a VPN identifier on a VRF.
Step 6	**address-family** [**ipv4** | **ipv6**] **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Enters the address-family submode.
Step 7	**import route-target** [*xx.yy.nn* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target import 1:1`	Imports the selected route target, optionally expressed as one of the following: • 4-byte AS number of the route target in xx.yy:nn format. Range is 0-65535.0-65535:0-65535 • AS number of the route target in nn format. Range is 0-65535. • IP address of the route target in A.B.C.D. format.
Step 8	**export route-target** [*xx.yy.nn* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target export 1:1`	Exports the selected route target, optionally expressed as one of the following: • 4-byte AS number of the route target in xx.yy:nn format. Range is 0-65535.0-65535:0-65535 • AS number of the route target in nn format. Range is 0-65535. • IP address of the route target in A.B.C.D. format.

	Command or Action	Purpose
Step 9	**root** **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# root`	Takes the user to the global configuration level.
Step 10	**multicast-routing vrf** *vrf_name* **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing vrf vrf1`	Enables multicast routing for the specified VRF.
Step 11	**address-family** [**ipv4** | **ipv6**] **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1)# address-family ipv4`	Enters the address-family submode.
Step 12	**mdt default mldp ipv4** *root-node* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1-ipv4)# mdt default mldp ipv4 2.2.2.2`	Configures MLDP MDT for a VRF. The root node can be IP address of a loopback or physical interface on any router (source PE, receiver PE or core router) in the provider network. The root node address should be reachable by all the routers in the network. The router from where the signalling occurs functions as the root node. The default MDT must be configured on each PE router to enable the PE routers to receive multicast traffic for this particular MVRF. **Note** By default MPLS MLDP is enabled. To disable, use the **no mpls ldp mldp** command. **Note** LSPVIF tunnel is created as a result of **mdt default mldp** *root-node* command.
Step 13	**mdt data** *mdt-group-address* **threshold** *value* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1-ipv4)# mdt data 239.0.0.0/24 threshold 1000`	Configures the threshold value for data MDT.
Step 14	**root** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1-ipv4)#`	Takes the user to the global configuration mode.

	Command or Action	Purpose
	`root`	
Step 15	**router bgp** *as-number* **vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 1 vrf vrf1`	Enters the BGP configuration mode.
Step 16	**rd** *route-distinguisher* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 10.0.0.4:1`	Creates routing and forwarding tables. Specify the route-distinguisher argument to add an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter an RD value in either of these formats: • 16-bit autonomous system number. For example, 101:3. • 32-bit IP address: your 16-bit number. For example, 192.168.122.15:1.
Step 17	**address-family ipv4 mdt** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 mdt`	Configures the BGP MDT address family.
Step 18	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# address-family vpnv4 unicast`	Configures the BGP vpnv4 address family.
Step 19	**root** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# root`	Takes the user to the global configuration mode.
Step 20	**router pim** **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters the PIM configuration mode.
Step 21	**vrf** *vrf_name* **Example:**	Specifies the VRF instance. .

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pim)# vrf vrf1`	
Step 22	**address-family** [**ipv4** | **ipv6**] **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf1)# address-family ipv4`	Enters the address-family submode.
Step 23	**rpf topology route-policy** *route_policy_name* **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf1-af)# rpf topology route-policy FOO`	Assigns a given routing policy to an RPF topology table.
Step 24	**root** **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf1-af)# root`	Takes the user to the global configuration mode.
Step 25	**route-policy** *route_policy_name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy FOO`	Configures the route policy for a profile. For more information about configuring route policy, see *Cisco IOS XR Routing Configuration Guide*.
Step 26	**set core-tree** *tree_type* **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# set core-tree mldp-rosen`	Specifies the MDT type for the route policy.
Step 27	**commit**	

Verification of LSM mLDP based MVPN Configuration

Use these commands to verify the LSM mLDP based MVPN intranet configuration:

- To check the MLDP neighbors, use the **show mpls mldp neighbors** command:

```
Router# show mpls mldp neighbors
mLDP neighbor database
 MLDP peer ID      : 1.0.0.1:0, uptime 15:36:30 Up,
  Capabilities     : GR, Typed Wildcard FEC, P2MP, MP2MP, MBB
  Target Adj       : No
  Upstream count   : 0
  Branch count     : 0
  LDP GR           : Enabled
```

```
                  : Instance: 1
 Label map timer  : never
 Policy filter in : None
 Path count       : 1
 Path(s)          : 11.11.11.10       GigabitEthernet0/2/0/0 LDP
 Adj list         : 11.11.11.10       GigabitEthernet0/2/0/0
 Peer addr list   : 8.39.21.2
                  : 10.0.0.1
                  : 10.1.1.1
                  : 10.2.2.1
                  : 10.3.3.1
                  : 10.4.4.1
                  : 10.5.5.1
                  : 10.6.6.1
                  : 10.7.7.1
                  : 10.8.8.1
                  : 10.9.9.1
                  : 10.10.10.1
                  : 1.11.11.1
                  : 1.12.12.1
                  : 1.13.13.1
                  : 1.14.14.1
                  : 1.15.15.1
                  : 1.16.16.1
                  : 1.17.17.1
                  : 1.18.18.1
                  : 1.19.19.1
                  : 1.20.20.1
                  : 1.21.21.1
                  : 1.22.22.1
                  : 1.23.23.1
                  : 1.24.24.1
                  : 1.25.25.1
                  : 1.26.26.1
                  : 1.27.27.1
                  : 1.28.28.1
                  : 1.29.29.1
                  : 1.30.30.1
                  : 11.11.11.10
                  : 111.113.1.5
                  : 111.112.1.1
                  : 8.39.21.222

MLDP peer ID      : 3.0.0.1:0, uptime 15:36:31 Up,
 Capabilities     : GR, Typed Wildcard FEC, P2MP, MP2MP, MBB
 Target Adj       : No
 Upstream count   : 334
 Branch count     : 328
 LDP GR           : Enabled
                  : Instance: 1
 Label map timer  : never
 Policy filter in : None
 Path count       : 1
 Path(s)          : 11.113.1.2        GigabitEthernet0/2/0/3 LDP
 Adj list         : 11.113.1.2        GigabitEthernet0/2/0/3
 Peer addr list   : 8.39.15.2
                  : 3.0.0.1
                  : 189.189.189.189
                  : 13.13.13.18
                  : 11.113.1.2
                  : 22.113.1.2
                  : 111.113.1.6
                  : 112.113.1.6
```

- To check the PIM neighbors, use the **show pim vrf vrf-name neighbor** command:

```
Router# show pim vrf A1_MIPMSI neighbor
PIM neighbors in VRF A1_MIPMSI

Neighbor Address             Interface              Uptime    Expires  DR pri  s

101.2.2.101*                 Loopback2              15:54:43  00:00:02 1 (DR) BP
101.0.0.101*                 LmdtA1/MIPMSI          15:54:43  00:00:02 1      B
102.0.0.102                  LmdtA1/MIPMSI          03:52:08  00:00:02 1      B
103.0.0.103                  LmdtA1/MIPMSI          15:28:13  00:00:02 1 (DR) B
60.3.0.1                     Multilink0/2/1/0/3     15:54:39  00:01:21 1      B
60.3.0.2*                    Multilink0/2/1/0/3     15:54:43  00:00:02 1 (DR) BP
60.1.0.5                     Serial0/2/2/0/1:1.16   15:54:42  00:01:42 1      B
60.1.0.6*                    Serial0/2/2/0/1:1.16   15:54:43  00:00:02 1 (DR) BP
60.2.0.1                     Serial0/5/0/0/1        15:54:42  00:01:17 1      B
60.2.0.2*                    Serial0/5/0/0/1        15:54:43  00:00:02 1 (DR) BP
```

- To check the multicast routes for a given VRF, use **show mrib vrf vrf_name route** command:

```
Router# show mrib vrf A1_MIPMSI route
IP Multicast Routing Information Base
Entry flags: L - Domain-Local Source, E - External Source to the Domain,
    C - Directly-Connected Check, S - Signal, IA - Inherit Accept,
    IF - Inherit From, D - Drop, MA - MDT Address, ME - MDT Encap,
    MD - MDT Decap, MT - MDT Threshold Crossed, MH - MDT interface handle
    CD - Conditional Decap, MPLS - MPLS Decap, MF - MPLS Encap, EX - Extranet
    MoFE - MoFRR Enabled, MoFS - MoFRR State
Interface flags: F - Forward, A - Accept, IC - Internal Copy,
    NS - Negate Signal, DP - Don't Preserve, SP - Signal Present,
    II - Internal Interest, ID - Internal Disinterest, LI - Local Interest,
    LD - Local Disinterest, DI - Decapsulation Interface
    EI - Encapsulation Interface, MI - MDT Interface, LVIF - MPLS Encap,
    EX - Extranet, A2 - Secondary Accept

(*,224.0.0.0/24) Flags: D
  Up: 15:57:19


(*,224.0.1.39) Flags: S
  Up: 15:57:19


(*,224.0.1.40) Flags: S
  Up: 15:57:19

  Outgoing Interface List
    Serial0/5/0/0/1 Flags: II LI, Up: 15:57:12

(*,225.0.0.0/19) RPF nbr: 101.2.2.101 Flags: L C
  Up: 15:57:19

  Outgoing Interface List
    Decapstunnel98 Flags: NS DI, Up: 15:57:10

(*,225.0.32.0/19) RPF nbr: 102.0.0.102 Flags: C
  Up: 15:57:19


(*,225.0.32.1) RPF nbr: 102.0.0.102 Flags: C
  Up: 04:08:30

  Incoming Interface List
```

```
    LmdtA1/MIPMSI Flags: A LMI, Up: 04:08:30
  Outgoing Interface List
    Serial0/2/2/0/1:1.16 Flags: F NS, Up: 04:08:30

(*,225.0.32.2) RPF nbr: 102.0.0.102 Flags: C
  Up: 04:08:30

  Incoming Interface List
    LmdtA1/MIPMSI Flags: A LMI, Up: 04:08:30
  Outgoing Interface List
    Serial0/2/2/0/1:1.16 Flags: F NS, Up: 04:08:30

(*,225.0.32.3) RPF nbr: 102.0.0.102 Flags: C
  Up: 04:08:30

  Incoming Interface List
    LmdtA1/MIPMSI Flags: A LMI, Up: 04:08:30
  Outgoing Interface List
    Serial0/2/2/0/1:1.16 Flags: F NS, Up: 04:08:30

(*,225.0.32.4) RPF nbr: 102.0.0.102 Flags: C
  Up: 04:08:30

  Incoming Interface List
    LmdtA1/MIPMSI Flags: A LMI, Up: 04:08:30
  Outgoing Interface List
    Serial0/2/2/0/1:1.16 Flags: F NS, Up: 04:08:30
```

- To check the MPLS forwarding status, use **show mpls forwarding** command:

```
Router# show mpls forwarding
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16000  16255       MLDP LSM ID: 0x1   Gi0/2/0/3    11.113.1.2      348727240
16001  16254       MLDP LSM ID: 0x3   Gi0/2/0/3    11.113.1.2      348727234
16002  16253       MLDP LSM ID: 0x5   Gi0/2/0/3    11.113.1.2      348727234
16003  16252       MLDP LSM ID: 0x7   Gi0/2/0/3    11.113.1.2      348727234
16004  16251       MLDP LSM ID: 0x9   Gi0/2/0/3    11.113.1.2      421876882
16005  16250       MLDP LSM ID: 0xb   Gi0/2/0/3    11.113.1.2      348726916
```

Configuring RPF Vector (IETF Standard Encoding)

This example shows how to enable RPF encoding using IETF standard:

```
(config)# router pim
(config-pim-default-ipv4)# address-family ipv4
(config-pim-default-ipv4)# rpf-vector use-standard-encoding
!
(config)# multicast-routing
(config-mcast)# interface TenGigE
(config-mcast)# interface TenGigE
```

Verification

```
Router#show pim neighbor
Tue Apr 17 10:15:40.961 PDT

PIM neighbors in VRF default
Flag: B - Bidir capable, P - Proxy capable, DR - Designated Router,
      E - ECMP Redirect capable
```

```
      * indicates the neighbor created for this router

Neighbor Address             Interface              Uptime    Expires  DR pri   Flags

25.25.25.1                   TenGigE       1w3d      00:01:36 1      B P
25.25.25.2*                  TenGigE       1w3d      00:01:41 1 (DR) B P E
32.32.32.2*                  TenGigE
       1w4d      00:01:40 1      B P E
32.32.32.3                   TenGigE
       1w4d      00:01:42 1 (DR) B P
```

In the above output, you can see "P" tag on the multicast enabled interfaces.

Configuring MVPN Static P2MP-TE

Perform these steps to configure the multicast VPN for static point-to-multipoint (P2MP) traffic engineering (TE).

Configuring MVPN P2MP on Ingress PE

Perform this task to configure MVPN P2MP on Ingress PE.

SUMMARY STEPS

1. **configure**
2. **multicast-routing**
3. **address-family** {**ipv4**|**ipv6**}
4. **mdt source** *type interface-path-id*
5. **interface all enable**
6. **vrf** *vrf-name*
7. **address-family** {**ipv4** | **ipv6**}
8. **bgp auto-discovery rsvpte**
9. **mdt static p2mp-te tunnel-mte** *value*
10. **interface all enable**
11. **router igmp**
12. **vrf** *name*
13. **interface** *type interface-path-id*
14. **static-group** *ip_group_address source-address*
15. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:**	Enters multicast routing configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# multicast-routing`	
Step 3	**address-family** {**ipv4**|**ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 or ipv6 address-family submode.
Step 4	**mdt source** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt source Loopback 0`	Specifies the MDT source address. **Note** The MDT source interface name should be the same as the one used for BGP peerings.
Step 5	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 6	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# vrf vrf1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode.
Step 7	**address-family** {**ipv4** | **ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1)# address-family ipv4`	Enters ipv4 (or ipv6)address-family submode.
Step 8	**bgp auto-discovery rsvpte** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1-ipv4)# bgp auto-discovery rsvp-te`	Enables the RSVP-TE I-PMSI core tree.
Step 9	**mdt static p2mp-te tunnel-mte** *value* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt static p2mp-te tunnel-mte 1`	Specifies the static p2mp-te mpls traffic engineering P2MP tunnel interface.

	Command or Action	Purpose
Step 10	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 11	**router igmp** **Example:** `RP/0/RSP0/CPU0:router(config)# router igmp`	Configures router igmp and enters the igmp configuration mode.
Step 12	**vrf** *name* **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# vrf vrf1`	Sets conditions for the access list to recognize the source as part of the specified access list set, in which each ACL describes a set of SSM groups to be mapped.
Step 13	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-igmp)# interface tunnel-mte1`	Configures the MPLS Traffic Engineering P2MP tunnel interface.
Step 14	**static-group** *ip_group_address source-address* **Example:** `RP/0/RSP0/CPU0:router(config-igmp-default-if)# static-group 232.1.1.1 192.1.1.2`	Configures the IGMP static multicast group.
Step 15	**commit**	

Configuring MVPN P2MP BGP

Perform this task to configure MVPN P2MP BGP.

SUMMARY STEPS

1. **configure**
2. **router bgp 100**
3. **bgp router-id** *ip_address*
4. **address-family** {**ipv4** | **ipv6**} **unicast**
5. **address-family** {**vpnv4** | **vpnv6**} **unicast**
6. **address-family** {**ipv4** | **ipv6**} **mvpn**
7. **neighbor** *address*
8. **remote-as** *2-byte AS number*

9. **update-source interface** *type interface-path-id*
10. **address-family** {**ipv4** | **ipv6**} **unicast**
11. **address-family** {**vpnv4** | **vpnv6**} **unicast**
12. **address-family** {**ipv4** | **ipv6**} **mvpn**
13. **vrf** *name*
14. **rd** *x.y format*
15. **bgp router-id** *ip_address*
16. **address-family** {**ipv4** | **ipv6**} **unicast**
17. **redistribute connected**
18. **address-family** {**ipv4** | **ipv6**} **mvpn**
19. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp 100** **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Configures the BGP routing process.
Step 3	**bgp router-id** *ip_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 12.33.42.34`	Configures the router-id for the BGP protocol.
Step 4	**address-family** {**ipv4**	**ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures ipv4 address-family for unicast and enters the address family command mode.
Step 5	**address-family** {**vpnv4**	**vpnv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures vpnv4 address-family for unicast and enters the address family command mode.
Step 6	**address-family** {**ipv4**	**ipv6**} **mvpn** **Example:**	Configures ipv4 address-family for mvpn and enters the address family command mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	
Step 7	**neighbor** *address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 1.3.45.6`	Specifies a neighbor router.
Step 8	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100`	Set remote AS with the specified 2-byte AS number.
Step 9	**update-source interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 1`	Set remote AS with the specified 2-byte AS number.
Step 10	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures ipv4 address-family for unicast and enters the address family command mode.
Step 11	**address-family** {**vpnv4** | **vpnv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures vpnv4 address-family for unicast and enters the address family command mode.
Step 12	**address-family** {**ipv4** | **ipv6**} **mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures ipv4 address-family for mvpn and enters the address family command mode.
Step 13	**vrf** *name* **Example:**	Configures the VRF.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf1`	
Step 14	**rd** *x.y format* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 1:1`	Configures the route distinguisher.
Step 15	**bgp router-id** *ip_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 12.33.42.34`	Configures the router-id for the BGP protocol.
Step 16	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures ipv4 address-family for unicast and enters the address family command mode.
Step 17	**redistribute connected** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# redistribute connected`	Redistributes information from another routing protocol through connected routes.
Step 18	**address-family** {**ipv4** | **ipv6**} **mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures ipv4 address-family for mvpn and enters the address family command mode.
Step 19	**commit**	

Configuring MVPN P2MP on Egress PE

Perform this task to configure MVPN P2MP on egress PE.

SUMMARY STEPS

1. **configure**
2. **multicast-routing**
3. **address-family** {**ipv4**|**ipv6**}
4. **mdt source** *type interface-path-id*
5. **interface all enable**

6. **vrf** *vrf-name*
7. **address-family** {**ipv4** | **ipv6**}
8. **core-tree-protocol rsvp-te group-list** *name*
9. **interface all enable**
10. **ipv4 access-list** *acl-name*
11. [*sequence-number*] **permit ipv4 host** *source_address* **host** [*destination_address*]
12. [*sequence-number*] **permit ipv4 any host** *destination_address*
13. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode.
Step 3	**address-family** {**ipv4**|**ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 or ipv6 address-family submode.
Step 4	**mdt source** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt source Loopback 1`	Specifies the MDT source address. **Note** The MDT source interface name should be the same as the one used for BGP peerings.
Step 5	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 6	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# vrf vrf1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode.

	Command or Action	Purpose
Step 7	**address-family** {**ipv4** | **ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1)# address-family ipv4`	Enters ipv4 (or ipv6)address-family submode.
Step 8	**core-tree-protocol rsvp-te group-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vrf1-ipv4)# core-tree-protocol rsvp-te group-list mvpn_acl`	Configures RSVP-TE as the core-tree-protocol and enters the core-tree-protocol configuration mode.
Step 9	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 10	**ipv4 access-list** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list mvpn_acl`	Enters IPv4 ACL configuration submode and creates a name for an IPv4 access list.
Step 11	[*sequence-number*] **permit ipv4 host** *source_address* **host** [*destination_address*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit 1 host 232.1.1.2 any`	Sets conditions for the access list to recognize the source as part of the specified access list set.
Step 12	[*sequence-number*] **permit ipv4 any host** *destination_address* **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 permit ipv4 any host 232.1.1.2`	Sets conditions for the access list to recognize the source as part of the specified access list set.
Step 13	**commit**	

Configuring MVPN InterAS Options

Perform these steps to configure the various MVPN InterAS options:

Configuring a PE Router for MVPN InterAS Option B or C

Perform this step to configure a PE router for MVPN InterAS option B or C:

SUMMARY STEPS

1. **configure**
2. **vrf vpn1**
3. **address-family ipv4 unicast**
4. **import route-target** *2-byte AS number*
5. **export route-target** *2-byte AS number*
6. **router bgp** *2-byte AS number*
7. **bgp router-id** *ipv4 address*
8. **address-family ipv4 unicast**
9. **allocate-label all**
10. **address-family vpnv4 unicast**
11. **address-family ipv4 mvpn**
12. **neighbor** *neighbor_address*
13. **remote-as** *2-byte AS number*
14. **update-source Loopback** *0-655335*
15. **address-family ipv4 labeled-unicast**
16. **address-family vpnv4 unicast**
17. **inter-as install**
18. **address-family ipv4 mvpn**
19. **vrf** *vpn1*
20. **rd** *2-byte AS number*
21. **address-family ipv4 unicast**
22. **route-target download**
23. **address-family ipv4 mvpn**
24. **inter-as install**
25. **mpls ldp**
26. **router-id** *ip address*
27. **mldp recursive-fec**
28. **interface** *type interface-path-id*
29. **multicast-routing**
30. **address-family ipv4**
31. **mdt source** *type interface-path-id*
32. **interface all enable**
33. **vrf vpn1**
34. **address-family ipv4**
35. **bgp auto-discovery mldp inter-as**
36. **mdt partitioned mldp ipv4 mp2mp**
37. **interface all enable**
38. **router pim**
39. **vrf vrf1**

40. **address-family ipv4**
41. **rpf toplogy route-policy** *policy_name*
42. **route-policy** *policy_name*
43. **set core-tree mldp-partitioned-mp2mp**
44. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**vrf vpn1** **Example:** `RP/0/RSP0/CPU0:router(config)# vrf vpn1`	Configures the vrf and enters the vrf configuration mode.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Configures the ipv4 address-family for a unicast topology and enters the ipv4 address-family submode.
Step 4	**import route-target** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 20:1`	Specifies the 2-byte AS number for the import route target extended communities.
Step 5	**export route-target** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 10:1`	Specifies the 2-byte AS number for the export route target extended communities.
Step 6	**router bgp** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Configures the router bgp and enters the router bgp configuration mode.
Step 7	**bgp router-id** *ipv4 address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id`	Configures the bgp router id with the ipv4 address. **Note** Steps 1 to 7 are common for both Option B and C.

	Command or Action	Purpose
	`10.10.10.1`	
Step 8	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures the ipv4 address-family with the unicast topology.
Step 9	**allocate-label all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# allocate-label all`	Allocates label for all prefixes. **Note** Steps 8 and 9 are part of the Option C configuration.
Step 10	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology.
Step 11	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 12	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.02`	Specifies and configures a neighbor router with the neighbor address.
Step 13	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100`	Sets remote AS with the mentioned 2-byte AS number.
Step 14	**update-source Loopback** *0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0`	Specifies the source of routing updates using the Loopback interface. **Note** Steps 10 to 14 are common to both Option B and C.

	Command or Action	Purpose
Step 15	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the ipv4 address-family with the labeled unicast topology. **Note** Step 15 is only performed for Option C configuration.
Step 16	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology. **Note** Step 16 is common to both Option B and C.
Step 17	**inter-as install** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# inter-as install`	Installs Inter-AS option. **Note** Step 17 is only for Option B configuration.
Step 18	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 19	**vrf** *vpn1* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# vrf vpn1`	Configures the vrf and enters the vrf configuration mode.
Step 20	**rd** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 10:1`	Configures the route distinguisher with a 2-byte AS number.
Step 21	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast`	Configures the ipv4 address-family for a unicast topology and enters the ipv4 address-family submode.
Step 22	**route-target download** **Example:**	Installs and configures the route-targets in RIB.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# route-target download`	
Step 23	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn. **Note** Steps 18 to 23 are common to both Option B and C.
Step 24	**inter-as install** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# inter-as install`	Installs Inter-AS option. **Note** Step 24 is only for Option C configuration.
Step 25	**mpls ldp** **Example:** `RP/0/RSP0/CPU0:router(config)# mpls ldp`	Configures the MPLS label distribution protocol (ldp). **Note** Steps 25 till 44 are common to both Option B and C.
Step 26	**router-id** *ip address* **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# router-id 10.10.10.1`	Configures the router id with the ip address.
Step 27	**mldp recursive-fec** **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# mldp recursive-fec`	Configures the mLDP recursive FEC support.
Step 28	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# interface GigabitEthernet 0/1/0/0`	Configures the GigabitEthernet/IEEE 802.3 interface(s).
Step 29	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enbales IP Multicast forwarding and enters the multicast routing configuration mode.

	Command or Action	Purpose
Step 30	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Configures the ipv4 address-family and enters ipv4 address-family submode.
Step 31	**mdt source** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mdt source Loopback 0`	Configures mvpn and specifies the interface used to set MDT source address.
Step 32	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 33	**vrf vpn1** **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# vrf vpn1`	Configures the vrf and enters the vrf configuration mode.
Step 34	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vpn1)# address-family ipv4`	Configures the ipv4 address-family and enters the ipv4 address-family submode.
Step 35	**bgp auto-discovery mldp inter-as** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vpn1-ipv4)# bgp auto-discovery mldp inter-as`	enables BGP MVPN auto-discovery.
Step 36	**mdt partitioned mldp ipv4 mp2mp** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vpn1-ipv4)# mdt partitioned mldp ipv4 mp2mp`	Enables MLDP MP2MP signaled partitioned distribution tree for ipv4 core. **Note** This configuration varies depending on what core tree option is being used. For example, the above step enables MLDP MP2MP core tree. Instead, if you select, P2MP core tree, the configuration enables MLDP P2MP core tree.

	Command or Action	Purpose
Step 37	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-vpn1-ipv4)# interface all enable`	Enables multicast routing and forwarding on all new and existing interfaces. You can also enable individual interfaces.
Step 38	**router pim** **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Configures the router pim and enters the pim configuration mode.
Step 39	**vrf vrf1** **Example:** `RP/0/RSP0/CPU0:router(config-pim)# vrf vrf1`	Configures the vrf and enters the vrf configuration mode.
Step 40	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf1)# address-family ipv4`	Configures the ipv4 address-family and enters the ipv4 address-family submode.
Step 41	**rpf toplogy route-policy** *policy_name* **Example:** `RP/0/RSP0/CPU0:router(config-pim-vrf1-ipv4)# rpf topology route-policy MSPMSI_MP2MP`	Configures the route-policy to select RPF topology.
Step 42	**route-policy** *policy_name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy MSPMSI_MP2MP`	Configures the route-policy to select RPF topology.
Step 43	**set core-tree mldp-partitioned-mp2mp** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# set core-tree mldp-partitioned-mp2mp`	Sets a MLDP Partitioned MP2MP core multicast distribution tree type.
Step 44	**commit**	

Configuring ASBR Router for MVPN InterAS Option B or C

Perform this step to configure ASBR router for MVPN InterAS Option B or C:

Before you begin

Perform these steps prior to starting the Configuring ASBR Router for MVPN InterAS Option B or C:

```
prefix-set IGP_leaks
  10.10.10.1/32,
  10.10.10.2/32,
  10.10.10.3/32
  end-set
!
route-policy IGP_INTER_AS_C_OUT
  if destination in IGP_leaks then
    pass
  else
    drop
  endif
  end-policy
!
```

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast** *destination prefix interface-type interface-path-id*
4. **router bgp** *2-byte AS number*
5. **bgp router-id** *ipv4 address*
6. **address-family vpnv4 unicast**
7. **retain route-target all**
8. **address-family ipv4 mvpn**
9. **retain route-target all**
10. **address-family ipv4 unicast**
11. **redistribute ospf** *router_tag*
12. **route-policy** *policy_name*
13. **allocate-label all**
14. **neighbor** *neighbor_address*
15. **remote-as** *2-byte AS number*
16. **update-source** *interface 0-655335*
17. **address-family vpnv4 unicast**
18. **address-family ipv4 labeled-unicast**
19. **route-policy** *policy_name* **in**
20. **route-policy** *policy_name* **out**
21. **neighbor** *neighbor_address*
22. **remote-as** *2-byte AS number*
23. **update-source Loopback** *0-655335*
24. **address-family vpnv4 unicast**
25. **address-family ipv4 labeled-unicast**

26. **next-hop-self**
27. **address-family ipv4 mvpn**
28. **next-hop-self**
29. **mpls ldp**
30. **router-id** *ip address*
31. **mldp recursive-fec**
32. **interface** *type interface-path-id*
33. **discovery transport-address** *ip_address*
34. **interface** *type interface-path-id*
35. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router static** **Example:** `RP/0/RSP0/CPU0:router(config)# router static`	Enables a static routing process.
Step 3	**address-family ipv4 unicast** *destination prefix interface-type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast 3.3.3.3/32 GigabitEthernet 0/1/0/1`	Configures the ipv4 address-family for the unicast topology with a destination prefix.
Step 4	**router bgp** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Configures the router bgp and enters the router bgp configuration mode.
Step 5	**bgp router-id** *ipv4 address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.10.10.1`	Configures the bgp router id with the ipv4 address. **Note** Steps 1 to 5 are common for both Option B and C.
Step 6	**address-family vpnv4 unicast** **Example:**	Configures the vpnv4 address-family with the unicast topology.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	
Step 7	**retain route-target all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# retain route-target all`	Accepts or retains the received updates containing at least one route target.
Step 8	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 9	**retain route-target all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# retain route-target all`	Accepts or retains the received updates containing at least one route target. **Note** Steps 6 to 9 are only for Option B Configuration.
Step 10	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures the ipv4 address-family for a unicast topology and enters the ipv4 address-family submode.
Step 11	**redistribute ospf** *router_tag* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# redistribute ospf 100`	Redistributes information from another routing protocol.
Step 12	**route-policy** *policy_name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy IGP_INTER_AS_C_OUT`	Configures the route-policy to select RPF topology.
Step 13	**allocate-label all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# allocate-label`	Allocates label for all prefixes. **Note** Steps 10 and 13 are part of the Option C configuration.

	Command or Action	Purpose
	`all`	
Step 14	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.02`	Specifies and configures a neighbor router with the neighbor address.
Step 15	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100`	Sets remote AS with the mentioned 2-byte AS number. **Note** Steps 14 and 15 are common to both Option B and C.
Step 16	**update-source** *interface 0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source GigabitEthernet 0/1/0/1`	Specifies the source of routing updates using the GigabitEthernet interface.
Step 17	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology. **Note** Steps 16 and 17 are for Option B configuration.
Step 18	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the ipv4 address-family with the labeled unicast topology. **Note** Step 18 is only performed for Option C configuration.
Step 19	**route-policy** *policy_name* **in** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies route-policy to inbound routes.
Step 20	**route-policy** *policy_name* **out** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies route-policy to outbound routes. **Note** Steps 19 and 20 are common to both Option B and C.

	Command or Action	Purpose
Step 21	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.02`	Specifies and configures a neighbor router with the neighbor address.
Step 22	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100`	Sets remote AS with the mentioned 2-byte AS number.
Step 23	**update-source Loopback** *0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0/1/0/1`	Specifies the source of routing updates using the Loopback interface.
Step 24	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology.
Step 25	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the ipv4 address-family with the labeled unicast topology. **Note** Step 25 is only performed for Option C configuration.
Step 26	**next-hop-self** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	Disables the next hop calculation for this neighbor. **Note** Steps 21 to 26 are common to both Option B and C, except 25 that is only applicable for Option C.
Step 27	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 28	**next-hop-self**	Disables the next hop calculation for this neighbor.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	**Note** Steps 25 and 26 are applicable only for Option B configuration.
Step 29	**mpls ldp** **Example:** `RP/0/RSP0/CPU0:router(config)# mpls ldp`	Configures the MPLS label distribution protocol (ldp). **Note** Steps 27 to 33 are common to both Option B and C.
Step 30	**router-id** *ip address* **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# router-id 10.10.10.1`	Configures the router id with the ip address.
Step 31	**mldp recursive-fec** **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# mldp recursive-fec`	Configures the mLDP recursive FEC support.
Step 32	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# interface GigabitEthernet 0/1/0/0`	Configures the GigabitEthernet/IEEE 802.3 interface(s).
Step 33	**discovery transport-address** *ip_address* **Example:** `RP/0/RSP0/CPU0:router(config-ldp-if)# discovery transport-address 3.3.3.2`	Configures interface LDP discovery parameters by specifying the interface LDP transport address.
Step 34	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-ldp)# interface GigabitEthernet 0/1/0/1`	Configures the GigabitEthernet/IEEE 802.3 interface(s).
Step 35	**commit**	

Configuring RR for MVPN InterAS Option C

Perform this step to configure RR for MVPN InterAS Option C:

SUMMARY STEPS

1. **configure**
2. **router bgp** *2-byte AS number*
3. **bgp router-id** *ipv4 address*
4. **address-family ipv4 unicast**
5. **allocate-label all**
6. **address-family vpnv4 unicast**
7. **address-family ipv4 mvpn**
8. **neighbor** *neighbor_address*
9. **remote-as** *2-byte AS number*
10. **update-source Loopback** *0-655335*
11. **address-family ipv4 labeled-unicast**
12. **route-reflector-client**
13. **address-family vpnv4 unicast**
14. **route-reflector-client**
15. **address-family ipv4 mvpn**
16. **route-reflector-client**
17. **neighbor** *neighbor_address*
18. **remote-as** *2-byte AS number*
19. **update-source Loopback** *0-655335*
20. **address-family ipv4 labeled-unicast**
21. **route-reflector-client**
22. **neighbor** *neighbor_address*
23. **remote-as** *2-byte AS number*
24. **update-source Loopback** *0-655335*
25. **address-family vpnv4 unicast**
26. **route-policy** *policy_name* **in**
27. **route-policy** *policy_name* **out**
28. **next-hop-unchanged**
29. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *2-byte AS number* **Example:**	Configures the router bgp and enters the router bgp configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# router bgp 100`	
Step 3	**bgp router-id** *ipv4 address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.10.10.1`	Configures the bgp router id with the ipv4 address.
Step 4	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Configures the ipv4 address-family for a unicast topology and enters the ipv4 address-family submode.
Step 5	**allocate-label all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates label for all prefixes.
Step 6	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology.
Step 7	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 8	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.1`	Specifies and configures a neighbor router with the neighbor address.
Step 9	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as`	Sets remote AS with the mentioned 2-byte AS number.

	Command or Action	Purpose
	`100`	
Step 10	**update-source Loopback** *0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0`	Specifies the source of routing updates using the Loopback interface.
Step 11	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the ipv4 address-family with the labeled unicast topology.
Step 12	**route-reflector-client** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# route-reflector-client`	Configures a neighbor as route reflector client.
Step 13	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Configures the vpnv4 address-family with the unicast topology.
Step 14	**route-reflector-client** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures a neighbor as route reflector client.
Step 15	**address-family ipv4 mvpn** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 mvpn`	Configures the ipv4 address-family with the mvpn.
Step 16	**route-reflector-client** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures a neighbor as route reflector client.

	Command or Action	Purpose
Step 17	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.2`	Specifies and configures a neighbor router with the neighbor address.
Step 18	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100`	Sets remote AS with the mentioned 2-byte AS number.
Step 19	**update-source Loopback** *0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0`	Specifies the source of routing updates using the Loopback interface.
Step 20	**address-family ipv4 labeled-unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the ipv4 address-family with the labeled unicast topology.
Step 21	**route-reflector-client** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# route-reflector-client`	Configures a neighbor as route reflector client.
Step 22	**neighbor** *neighbor_address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 20.20.20.3`	Specifies and configures a neighbor router with the neighbor address.
Step 23	**remote-as** *2-byte AS number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200`	Sets remote AS with the mentioned 2-byte AS number.

	Command or Action	Purpose
Step 24	**update-source Loopback** *0-655335* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0`	Specifies the source of routing updates using the Loopback interface.
Step 25	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`	Configures the vpnv4 address-family with the unicast topology.
Step 26	**route-policy** *policy_name* **in** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies route-policy to inbound routes.
Step 27	**route-policy** *policy_name* **out** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies route-policy to outbound routes.
Step 28	**next-hop-unchanged** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged`	Indicates that the next hop should be kept as is and not overwritten, before advertising to eBGP peers.
Step 29	**commit**	

Configuring Multitopology Routing

This set of procedures configures multitopology routing, which is used by PIM for reverse-path forwarding (RPF) path selection.

- "Configuring a Global Topology and Associating It with an Interface" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Enabling an IS-IS Topology" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Placing an Interface in a Topology in IS-IS" (required)

For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Configuring a Routing Policy" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Restrictions for Configuring Multitopology Routing

- Only the default VRF is currently supported in a multitopology solution.
- Only protocol-independent multicast (PIM) and intermediate system-intermediate system (IS-IS) routing protocols are currently supported.
- Topology selection is restricted solely to (S, G) route sources for both SM and SSM. Static and IS-IS are the only interior gateway protocols (IGPs) that support multitopology deployment.

  For non-(S, G) route sources like a rendezvous point or bootstrap router (BSR), or when a route policy is not configured, the current policy default remains in effect. In other words, either a unicast-default or multicast-default table is selected for all sources based on any of the following configurations:

  - Open Shortest Path First (OSPF)
  - Intermediate System-to-Intermediate System (IS-IS)
  - Multiprotocol Border Gateway Protocol (MBGP)

Note Although both **multicast** and **unicast** keywords are available when using the **address-family** {**ipv4** | **ipv6**} command in routing policy language (RPL), only topologies under multicast SAFI can be configured globally.

Information About Multitopology Routing

Configuring multitopology networks requires the following tasks:

- "Configuring a Global Topology and Associating It with an Interface" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Enabling an IS-IS Topology" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Placing an Interface in a Topology in IS-IS" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

- "Configuring a Routing Policy" (required)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Configuring an RPF Topology in PIM

SUMMARY STEPS

1. **configure**

2. **router pim address-family** {**ipv4** | **ipv6**}
3. **rpf topology route-policy** *policy-name*
4. **exit**
5. **multicast-routing address-family** {**ipv4** | **ipv6**}
6. **interface all enable**
7. **commit**
8. **show pim [vrf** *vrf-name*] [**ipv4** | **ipv6**] [{**unicast** | **multicast** | **safi-all**} **topology** {*table-name* | **all**}] **rpf** [*ip-address* | **hash** | **summary** | **route-policy**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**router pim address-family** {**ipv4**	**ipv6**}
		Example:
		`RP/0/RSP0/CPU0:router(config)#`
		`RP/0/RSP0/CPU0:router(config-pim-default-ipv4)#`
			Enters PIM address-family configuration submode for the IP prefix you select.

Step 3	**rpf topology route-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pim-default-ipv)# rpf topology route-policy mtpolicy`	Assigns a given routing policy to an RPF topology table.

Step 4	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pim-default-ipv6)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		Exits pim address-family configuration submode.

Step 5	**multicast-routing address-family** {**ipv4**	**ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing address-family ipv4`	Enters multicast address-family configuration submode.
Step 6	**interface all enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default- ipv4))#`	Enables multicast routing and forwarding on all new and existing interfaces.

	Command or Action	Purpose
	`interface all enable`	
Step 7	**commit**	
Step 8	**show pim [vrf** *vrf-name*] [**ipv4** | **ipv6**] [{**unicast** | **multicast** | **safi-all**} **topology** {*table-name* | **all**}] **rpf** [*ip-address* | **hash** | **summary** | **route-policy**] **Example:** `RP/0/RSP0/CPU0:router# show pim vrf mtt rpf ipv4 multicast topology all rpf`	Shows PIM RPF entries for one or more tables.

Configuring MVPN Extranet Routing

To be able to import unicast routes from source VRFs to receiver VRFs, the import route targets of receiver VRFs must match the export route targets of a source VRF. Also, all VRFs on the PEs where the extranet source-receiver switchover takes place should be added to the BGP router configuration on those PEs.

Configuring MVPN extranet routing consists of these mandatory and optional tasks, which should be performed in the sequence shown:

- "Configuring a Routing Policy" (required only if performing the following task)

  For information, see *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

For examples of an end-to-end configuration of each of the two available MVPN extranet topology solutions, see Configuring MVPN Extranet Routing: Example, on page 241.

Prerequisites for MVPN Extranet Routing

- PIM-SM and PIM-SSM are supported. You must configure the multicast group range in the source and receiver VRFs with a matching PIM mode.
- Because only static RP configuration is currently supported for a given multicast group range, both source and receiver MVRFs must be configured with the same RP.
- In the IPv6 Connectivity over MVPN topology model, the data MDT encapsulation range should be large enough to accommodate extranet streams without any aggregation. This prevents extranet traffic, flowing to multiple VRFs, from being carried into only one data MDT.
- Data MDT configuration is required on only the Source VRF and Source PE Router.

Restrictions for MVPN Extranet Routing

- PIM-DM and PIM-BIDIR are not supported.
- Cisco IOS XR Software software supports only IPv4 extranet multicast routing over IPv4 core multicast routing.
- Any PE can be configured as an RP except a PE in the "Receiver VRF on the Source PE Router" model where the extranet switchover occurs, and where the source VRF has no interfaces. This is because the

source VRF must have some physical interface to signal the data packets being received from the first hop.

- Cisco IOS XR Software currently supports only one encapsulation of VRF traffic on an extranet. This means that only one encapsulation interface (or MDT) is allowed in the outgoing forwarding interface list of the multicast route. If, for a given stream, there are multiple receiver VRFs joining the same source VRF, only the first receiver VRF receives traffic; other receiver VRF joins are discarded.

Note This limitation applies only to IPv6 Connectivity over MVPN topology model.

Configuring VPN Route Targets

This procedure demonstrates how to configure a VPN route target for each topology.

Note Route targets should be configured so that the receiver VRF has unicast reachability to prefixes in the source VRF. These configuration steps can be skipped if prefixes in the source VRF are already imported to the receiver VRF.

SUMMARY STEPS

1. **configure**
2. **vrf** *source-vrf*
3. **address-family** [**ipv4** | **ipv6**} **unicast**
4. **import route-target** [*xx.yy:nn* | *as-number:nn* | *ip-address:nn*]
5. **export route-target** [*xx.yy:nn* | *as-number:nn* | *ip-address:nn*]
6. **commit**
7. **configure**
8. **vrf** *receiver-vrf*
9. Repeat Step 3 through Step 6.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**vrf** *source-vrf*
	Example:
	`RP/0/RSP0/CPU0:router(config)# vrf green`
	`RP/0/RSP0/CPU0:router(config-vrf)#`
		Configures a VRF instance for the source PE router.

	Command or Action	Purpose
Step 3	**address-family** [**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies a unicast IPv4 or IPv6 address family and enters address family configuration submode. **Note** Only IPv4 addressing is supported for extranet.

Step 4	**import route-target** [*xx.yy:nn* | *as-number:nn* | *ip-address:nn*]
	Example:
	`RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 234:222`
	`RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 100:100`
		Imports the selected route target, optionally expressed as one of the following :
		• 4-byte AS number of the route target in *xx.yy:nn* format. Range is 0-65535.0-65535:0-65535
		• AS number of the route target in *nn* format. Range is 0-65535.
		• IP address of the route target in *A.B.C.D.* format.

Step 5	**export route-target** [*xx.yy:nn* | *as-number:nn* | *ip-address:nn*] **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 100:100`	Exports the selected route target, optionally expressed as one of the following: • 4-byte AS number of the route target in *xx.yy:nn* format. Range is 0-65535.0-65535:0-65535 • AS number of the route target in *nn* format. Range is 0-65535. • IP address of the route target in *A.B.C.D.* format.
Step 6	**commit**	
Step 7	**configure**	

Step 8	**vrf** *receiver-vrf*
	Example:
	`RP/0/RSP0/CPU0:router(config)# vrf red`
	`RP/0/RSP0/CPU0:router(config-vrf)#`
		Configures a VRF instance for the receiver PE router.

Step 9	Repeat Step 3 through Step 6.	—

Interconnecting PIM-SM Domains with MSDP

To set up an MSDP peering relationship with MSDP-enabled routers in another domain, you configure an MSDP peer to the local router.

If you do not want to have or cannot have a BGP peer in your domain, you could define a default MSDP peer from which to accept all Source-Active (SA) messages.

Finally, you can change the Originator ID when you configure a logical RP on multiple routers in an MSDP mesh group.

Before you begin

You must configure MSDP default peering, if the addresses of all MSDP peers are not known in BGP or multiprotocol BGP.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 address** *address mask*
4. **exit**
5. **router msdp**
6. **default-peer** *ip-address* [**prefix-list** *list*]
7. **originator-id** *type interface-path-id*
8. **peer** *peer-address*
9. **connect-source** *type interface-path-id*
10. **mesh-group** *name*
11. **remote-as** *as-number*
12. **commit**
13. **show msdp** [**ipv4**] **globals**
14. **show msdp** [**ipv4**] **peer** [*peer-address*]
15. **show msdp** [**ipv4**] **rpf** *rpf-address*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface loopback 0`	(Optional) Enters interface configuration mode to define the IPv4 address for the interface. **Note** This step is required if you specify an interface type and number whose primary address becomes the source IP address for the TCP connection.
Step 3	**ipv4 address** *address mask* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0`	(Optional) Defines the IPv4 address for the interface. **Note** This step is required only if you specify an interface type and number whose primary address becomes the source IP address for the TCP connection. See optional for information about configuring the **connect-source** command.
Step 4	**exit** **Example:**	Exits interface configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# end`	
Step 5	**router msdp** **Example:** `RP/0/RSP0/CPU0:router(config)# router msdp`	Enters MSDP protocol configuration mode.
Step 6	**default-peer** *ip-address* [**prefix-list** *list*] **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# default-peer 172.23.16.0`	(Optional) Defines a default peer from which to accept all MSDP SA messages.
Step 7	**originator-id** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# originator-id GigabitEthernet0/1/1/0`	(Optional) Allows an MSDP speaker that originates a (Source-Active) SA message to use the IP address of the interface as the RP address in the SA message.
Step 8	**peer** *peer-address* **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# peer 172.31.1.2`	Enters MSDP peer configuration mode and configures an MSDP peer. • Configure the router as a BGP neighbor. • If you are also BGP peering with this MSDP peer, use the same IP address for MSDP and BGP. You are not required to run BGP or multiprotocol BGP with the MSDP peer, as long as there is a BGP or multiprotocol BGP path between the MSDP peers.
Step 9	**connect-source** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-msdp-peer)# connect-source loopback 0`	(Optional) Configures a source address used for an MSDP connection.
Step 10	**mesh-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-msdp-peer)# mesh-group internal`	(Optional) Configures an MSDP peer to be a member of a mesh group.
Step 11	**remote-as** *as-number* **Example:**	(Optional) Configures the remote autonomous system number of this peer.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-msdp-peer)# remote-as 250`	
Step 12	**commit**	
Step 13	**show msdp** [**ipv4**] **globals** **Example:** `RP/0/RSP0/CPU0:router# show msdp globals`	Displays the MSDP global variables.
Step 14	**show msdp** [**ipv4**] **peer** [*peer-address*] **Example:** `RP/0/RSP0/CPU0:router# show msdp peer 172.31.1.2`	Displays information about the MSDP peer.
Step 15	**show msdp** [**ipv4**] **rpf** *rpf-address* **Example:** `RP/0/RSP0/CPU0:router# show msdp rpf 172.16.10.13`	Displays the RPF lookup.

Controlling Source Information on MSDP Peer Routers

Your MSDP peer router can be customized to control source information that is originated, forwarded, received, cached, and encapsulated.

When originating Source-Active (SA) messages, you can control to whom you will originate source information, based on the source that is requesting information.

When forwarding SA messages you can do the following:

- Filter all source/group pairs
- Specify an extended access list to pass only certain source/group pairs
- Filter based on match criteria in a route map

When receiving SA messages you can do the following:

- Filter all incoming SA messages from an MSDP peer
- Specify an extended access list to pass certain source/group pairs
- Filter based on match criteria in a route map

In addition, you can use time to live (TTL) to control what data is encapsulated in the first SA message for every source. For example, you could limit internal traffic to a TTL of eight hops. If you want other groups to go to external locations, you send those packets with a TTL greater than eight hops.

By default, MSDP automatically sends SA messages to peers when a new member joins a group and wants to receive multicast traffic. You are no longer required to configure an SA request to a specified MSDP peer.

SUMMARY STEPS

1. **configure**
2. **router msdp**
3. **sa-filter** {**in** | **out**} {*ip-address* | *peer-name*} [**list** *access-list-name*] [**rp-list** *access-list-name*]
4. **cache-sa-state** [**list** *access-list-name*] [**rp-list** *access-list-name*]
5. **ttl-threshold** *ttl-value*
6. **exit**
7. **ipv4 access-list** *name* [*sequence-number*] **permit** *source* [*source-wildcard*]
8. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router msdp** **Example:** `RP/0/RSP0/CPU0:router(config)# router msdp`	Enters MSDP protocol configuration mode.
Step 3	**sa-filter** {**in** | **out**} {*ip-address* | *peer-name*} [**list** *access-list-name*] [**rp-list** *access-list-name*] **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# sa-filter out router.cisco.com list 100`	Configures an incoming or outgoing filter list for messages received from the specified MSDP peer. • If you specify both the **list** and **rp-list** keywords, all conditions must be true to pass any source, group (S, G) pairs in outgoing Source-Active (SA) messages. • You must configure the **ipv4 access-list** command in Step 7, on page 191. • If all match criteria are true, a **permit** from the route map passes routes through the filter. A **deny** filters routes. • This example allows only (S, G) pairs that pass access list 100 to be forwarded in an SA message to the peer named router.cisco.com.
Step 4	**cache-sa-state** [**list** *access-list-name*] [**rp-list** *access-list-name*] **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# cache-sa-state`	Creates and caches source/group pairs from received Source-Active (SA) messages and controls pairs through access lists.

	Command or Action	Purpose
	`list 100`	
Step 5	**ttl-threshold** *ttl-value* **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# ttl-threshold 8`	(Optional) Limits which multicast data is sent in SA messages to an MSDP peer. • Only multicast packets with an IP header TTL greater than or equal to the *ttl-value* argument are sent to the MSDP peer specified by the IP address or name. • Use this command if you want to use TTL to examine your multicast data traffic. For example, you could limit internal traffic to a TTL of 8. If you want other groups to go to external locations, send those packets with a TTL greater than 8. • This example configures a TTL threshold of eight hops.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# exit`	Exits the current configuration mode.
Step 7	**ipv4 access-list** *name* [*sequence-number*] **permit** *source* [*source-wildcard*] **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list 100 20 permit 239.1.1.1 0.0.0.0`	Defines an IPv4 access list to be used by SA filtering. • In this example, the access list 100 permits multicast group 239.1.1.1. • The **ipv4 access-list** command is required if the keyword **list** is configured for SA filtering in Step 3, on page 190.
Step 8	**commit**	

Configuring MSDP MD5 Password Authentication

SUMMARY STEPS

1. **configure**
2. **router msdp**
3. **peer** *peer-address*
4. **password** {**clear** | **encrypted**} *password*
5. **commit**
6. **show mfib** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **hardware route** {***** | *source-address* | *group-address*[/*prefix-length*]} **location** *node-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router msdp** **Example:** `RP/0/RSP0/CPU0:router(config)# router msdp`	Enters MSDP configuration mode.
Step 3	**peer** *peer-address* **Example:** `RP/0/RSP0/CPU0:router(config-msdp)# peer 10.0.5.4`	Configures the MSDP peer.
Step 4	**password** {**clear** | **encrypted**} *password* **Example:** `RP/0/RSP0/CPU0:router(config-msdp-peer)# password encrypted a34bi5m`	Configures the password.
Step 5	**commit**	
Step 6	**show mfib** [**vrf** *vrf-name*] [**ipv4** | **ipv6**] **hardware route** {***** | *source-address* | *group-address*[/*prefix-length*]} **location** *node-id* **Example:** `RP/0/RSP0/CPU0:router# show mfib hardware route * location 0/1/cpu0`	Displays multicast routes configured with multicast QoS and the associated parameters.

Configuring VRF for MSDP

Use the **vrf** keyword in the MSDP configuration mode to enable VRF for MSDP.

SUMMARY STEPS

1. **configure**
2. **router msdp**
3. **vrf** *vrf-name*
4. **peer** *peer-address*
5. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router msdp** **Example:** `RP/0/RSP0/CPU0:router(config)# router msdp`	Enters MSDP configuration mode.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-msdp) # vrf vrf1`	Enables VRF configuration for MSDP.
Step 4	**peer** *peer-address* **Example:** `RP/0/RSP0/CPU0:router(config-msdp) # peer 10.0.0.2`	Configures the VRF MSDP peer .
Step 5	**commit**	

Multicast only fast reroute (MoFRR)

MoFRR allows fast reroute for multicast traffic on a multicast router. MoFRR minimizes packet loss in a network when node or link failures occur(at the topology merge point). It works by making simple enhancements to multicast routing protocols.

MoFRR involves transmitting a multicast join message from a receiver towards a source on a primary path and transmitting a secondary multicast join message from the receiver towards the source on a backup path. Data packets are received from the primary and secondary paths. The redundant packets are discarded at topology merge points with the help of Reverse Path Forwarding (RPF) checks. When a failure is detected on the primary path, the repair occurs locally by changing the interface on which packets are accepted to the secondary interface, thus improving the convergence times in the event of a node or link failure on the primary path.

MoFRR supports ECMP (Equal Cost Multipath) and non-ECMP topologies as well.

TI (Topology Independent) MoFRR is a multicast feature that performs fast convergence (Fast ReRoute) for specified routes/flows when failure is detected on one of the paths between the router and the source.

Operating Modes of MoFRR

- Flow-based MoFRR— exposes the primary and secondary RPF interfaces to the forwarding plane, with switchover occurring entirely at the hardware level.

Faster convergence is obtainable in Flow-based MoFRR by monitoring the packet counts of the primary stream. If no activity is detected for 30 ms, the switch over is triggered to the backup stream and the traffic loss is within 50 ms.

Restrictions

These limitations apply to MoFRR deployments when the Cisco ASR 9000 Series SPA Interface Processor-700 linecard is used in the Cisco ASR 9000 Series Router chassis.

- Cisco ASR 9000 Series SPA Interface Processor-700 cannot be used on ingress interface as either the primary or backup (ECMP paths) path back to the multicast source.
- The egress interfaces on Cisco ASR 9000 Series SPA Interface Processor-700 may lead to duplicate multicast streams for short periods of time (the time between the switch from Trident primary to Trident backup paths on ingress).
- Flow-based MoFRR is not supported on 5th Generation ASR 9000 line cards.
- When using 3rd Generation and 5th Generation ASR 9000 line cards in the same chassis, primary or backup FRR path should not use 5th Generation ASR 9000 line cards.

Non-ECMP MoFRR

TI (Topology-Independent) MoFRR is a multicast feature that performs fast convergence (Fast ReRoute) for specified routes/flows when failure is detected on one of the paths between the router and the source.

Flow based non-ECMP approach uses a mechanism where two copies of the same multicast stream flow through disjoint paths in the network. At the point in the network (usually the tail PE that is closer to the receivers) where the two streams merge, one of the streams is accepted and forwarded on the downstream links, while the other stream is discarded. When a failure is detected in the primary stream due to a link or node failure in the network, MoFRR instructs the forwarding plane to start accepting packets from the backup stream (which now becomes the primary stream).

For more information about topology independent MoFRR, refer the *Multicast Configuration Guide for Cisco ASR 9000 Series Routers*.

Implementing Non-ECMP MoFRR

The config handler in PIM creates a mapping between (S1, G) and (S2, G) in an internal mapping database. No explicit route is created till a downstream join / data signal is received for (S1, G).

Downstream (S, G) join

The tail PE on receipt of (S, G) JOIN looks up the mapping database and,

- Creates the (S1, G) route entry with proxy info and marks it as primary mofrr route.
- Creates the (S2, G) route entry with the proxy info and marks it as backup mofrr route.
- Creates reference to (S2, G) from (S1, G) route and vice versa.
- Redistributes route with MoFRR primary & backup flags to PD.

Downstream (S,G) prune

The tail PE on receipt of (S, G) PRUNE looks up the mapping database and,

- • Deletes the (S1, G) route entry with proxy info and redistributes the route delete.
  • Deletes the (S2, G) route entry with proxy info and redistributes the route delete.

Data Signaling

Head PE on receipt of (S, G) traffic will clone the traffic as (S1, G) and (S2, G) and send it out on the interfaces on which (S1, G) / (S2, G) join has been received. This is done because the (S, G) entry is created with an encap-id with 2 encap-oles corresponding to (S1, G) and (S2, G).

On Tail PE on receipt of (S1, G) traffic the header is replaced as (S, G) and sent out on the interfaces on which (S, G) join has been received. If traffic is not received on (S1, G) on tail node for 50 ms then ucode initiates a switchover event and starts accepting traffic on (S2, G) and sends switchover notifications to control plane.

Configuring MoFRR

RIB-based MoFRR

SUMMARY STEPS

1. **configure**
2. **router pim**
3. **mofrr rib** *acl-name*
4. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters the PIM configuration mode.
Step 3	**mofrr rib** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(pim)# mofrr rib acl1`	Enter the ACL name.
Step 4	**commit**	

Flow-based MoFRR

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *acl-name*
3. *sequence number* [**permit**|**deny**] **ipv4 host** *address* [**host** *address* | **any**]
4. **exit**
5. **router pim**
6. **mofrr** *acl-name*
7. **commit**
8. **show mfib hardware route summary location**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**ipv4 access-list** *acl-name* **Example:** `RP/0/RSP0/CPU0:router (config)# ipv4 access-list flow_mofrr`	Enters IPv4 access list configuration mode and configures the named access list.
Step 3	*sequence number* [**permit**|**deny**] **ipv4 host** *address* [**host** *address*	**any**] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl) #10 permit ipv4 host 20.0.0.2 any`	Specifies one or more conditions allowed or denied in the created IPv4 access list.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# exit`	Saves the MoFRR acl configuration and exists the IPv4 acl configuration mode. You need to **exit** twice here.
Step 5	**router pim** **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters the PIM configuration mode.

	Command or Action	Purpose
Step 6	**mofrr** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(pim)# mofrr flow_mofrr`	Enables MoFRR for the specified access list source group with hardware switchover triggers. This is supported on IPv4 only.
Step 7	**commit**	
Step 8	**show mfib hardware route summary location** **Example:** `RP/0/RSP0/CPU0:router# show mfib hardware route 4`	Displays the number of enabled MoFRR routes.

Configuring Head PE Router (for MoFRR)

Pre-requisites

- ACL configurations. (for detailed information on how to configure ACLs, refer the Configuring ACLs chapter of the IP Addresses and Services configuration guide.)

The head PE router can be configured as follows:

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family ipv4**]
3. **mofrr**
4. **mofrr** *acl-name*
5. **clone source S to S1 S2masklen***n*
6. **commit**
7. **show pim topology** *route*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family ipv4**] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters PIM configuration mode, or PIM address-family configuration submode.

	Command or Action	Purpose
Step 3	**mofrr** **Example:** `RP/0/RSP0/CPU0:router(config-pim)# mofrr`	Enters PIM Multicast only FRR configuration mode.
Step 4	**mofrr** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-mofrr)# flow acl1`	Enables MoFRR with hardware switchover triggers for the specified access-list.
Step 5	**clone source S to S1 S2masklen***n* **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-mofrr)# clone source 10.1.1.1 to 20.2.2.2 50.5.5.5 masklen 32`	Duplicates source (S) to S1 and S2 with the specified mask length. A mapping is created between (S,G) , (S1,G) and (S2,G). S1 is the primary path and S2 is the secondary path.
Step 6	**commit**	
Step 7	**show pim topology** *route* **Example:** `RP/0/RSP0/CPU0:router# show pim topology 232.0.0.1`	This command verifies the mapping between the source S and S1 and S2. S, S1, S2 entries are updated in the displayed MoFRR details.

Configuring Tail PE Router (for MoFRR)

-

SUMMARY STEPS

1. **configure**
2. **router pim** [**address-family ipv4**]
3. **mofrr**
4. **mofrr** *acl-name*
5. **clone join S to S1 S2masklen***length*
6. **rpf-vector** *source***masklen***length*
7. **commit**
8. **show mfib hardware router mofrr** *route***location***interface-path-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim** [**address-family ipv4**] **Example:** `RP/0/RSP0/CPU0:router(config)# router pim`	Enters PIM configuration mode, or PIM address-family configuration submode.
Step 3	**mofrr** **Example:** `RP/0/RSP0/CPU0:router(config-pim)# mofrr`	Enters PIM Multicast only FRR configuration mode.
Step 4	**mofrr** *acl-name* **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-mofrr)# flow acl1`	Enables MoFRR with hardware switchover triggers for the specified access-list.
Step 5	**clone join S to S1 S2masklen***length* **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-mofrr)# clone join 10.1.1.1 to 20.2.2.2 50.5.5.5 masklen 32`	Duplicates source to S1 and S2 with the specified mask length. A mapping is created between (S,G) ,(S1,G) and (S2,G).
Step 6	**rpf-vector** *source***masklen***length* **Example:** `RP/0/RSP0/CPU0:router(config-pim-ipv4-mofrr)# rpf-vector 10.1.1.1 masklen 10 R1, R2.....`	Configures the reachability of the tail-node.
Step 7	**commit**	
Step 8	**show mfib hardware router mofrr** *route***location***interface-path-id* **Example:** `RP/0/RSP0/CPU0:router# show mfib hardware router mofrr 232.0.0.1 location 0/1/1/1`	This command verifies the mapping between the source S and S1 and S2. S, S1, S2 entries are updated in the displayed MoFRR details.

Enabling multicast on PW-HE interfaces

This task enables multicast on PW-HE interfaces.

SUMMARY STEPS

1. **configure**
2. **multicast-routing** [**address-family** {**ipv4** | **ipv6**}]
3. **interface pw-ether1**
4. **enable**
5. exit
6. **vrf** *vrf-name*
7. **address-family ipv4**
8. **interface** *type interface path-id*
9. **enable**
10. exit
11. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** [**address-family** {**ipv4** | **ipv6**}] **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing address-family ipv4`	Enters multicast routing configuration mode.
Step 3	**interface pw-ether1** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# interface pw-ether1`	Enters the pseudowire interface configuration mode.
Step 4	**enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# enable`	Enables multicast routing on pseudowire interfaces.
Step 5	exit **Example:**	Exits the current configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-mcast-ipv4)# exit`	
Step 6	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router (config-mcast) # vrf v1`	Enters the vrf configuration mode.
Step 7	**address-family ipv4** **Example:** `RP/0/RSP0/CPU0:router(config)# address-family ipv4`	Enters the IPv4 address-family configuration mode.
Step 8	**interface** *type interface path-id* **Example:** `RP/0/RSP0/CPU0:router (config-mcast-vrf-v4) # interface pw-ether2`	Enters the vrf mode for the specified pw interface.
Step 9	**enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# enable`	Enables multicast routing on the pw interface in the vrf.
Step 10	exit **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# exit`	Exits the current configuration mode. **Note** This step can be used, more than once.
Step 11	**commit**	

Static join

The static join can be achieved with IGMP or MLD. The **router mld** or **router igmp** commands can be used to enter the MLD or IGMP modes respectively. The examples section (later in this chapter) includes the examples for both the cases.

SUMMARY STEPS

1. **configure**
2. **router mld**
3. **interface** *type interface-path-id*
4. **static-group** *ip-group-address source-address*
5. exit
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router mld** **Example:** `RP/0/RSP0/CPU0:router(config)# router mld`	Enters the MLD multicast routing configuration mode.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-mld)# interface pw-ether1`	Enters the pseudowire interface configuration mode.
Step 4	**static-group** *ip-group-address source-address* **Example:** `RP/0/RSP0/CPU0:router(config-mld-default-if)# static-group ff35::e100 2000:10::1`	Enables pw-ether1 interface to statistically join a multicast group.
Step 5	exit **Example:** `RP/0/RSP0/CPU0:router(config-mcast-ipv4)# exit`	Exits the current configuration mode. **Note** This step can be used, more than once.
Step 6	**commit**	

Configuring Route Policy for Static RPF

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family**[**ipv4** | **ipv6**][**multicast** | **unicast**]*destination prefix interface-typeinterface-path-id*
4. **exit**
5. **route-policy***policy-name*
6. **set rpf-topology** *policy-name***address-family**[**ipv4** | **ipv6**]**multicast** | **unicasttopology***name*
7. **end route-policy**
8. **router pim address-family**[**ipv4** | **ipv6**]
9. **rpf topology route-policy***policy-name***pim policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router static** **Example:** `RP/0/RSP0/CPU0:router(config) # router static`	Enables a static routing process.
Step 3	**address-family**[**ipv4** | **ipv6**][**multicast** | **unicast**]*destination prefix interface-typeinterface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-static) # address-family ipv4 multicast 202.93.100.4/ 32 202.95.1.1`	Configures the ipv4 multicast address-family topology with a destination prefix.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-afi) # exit`	Exits from the address family configuration mode.
Step 5	**route-policy***policy-name* **Example:** `RP/0/RSP0/CPU0:router(config) # route-policy r1`	Configures the route policy to select the RPF topology.
Step 6	**set rpf-topology** *policy-name***address-family**[**ipv4** | **ipv6**]**multicast** | **unicasttopology***name* **Example:** `RP/0/RSP0/CPU0:router(config-rpl) # set rpf-topology p1 ipv4 multicast topology t1`	Configures the PIM rpf-topology attributes for the selected multicast address-family.
Step 7	**end route-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl) # end route-policy r1`	Ends the route policy.
Step 8	**router pim address-family**[**ipv4** | **ipv6**] **Example:** `RP/0/RSP0/CPU0:router(config) # router pim address-family ipv4`	Enters the PIM address-family configuration sub-mode.
Step 9	**rpf topology route-policy***policy-name***pim policy** **Example:** `RP/0/RSP0/CPU0:router(config) # rpf topology route-policy r1 pim policy`	Selects the RPF topology for the configured route-policy.

Point-to-Multipoint Traffic Engineering Label-Switched Multicast

IP multicast was traditionally used for IPTV broadcasting and content delivery services. MPLS-TE (traffic engineering) is fast replacing the IP multicast technique because of the various advantages of MPLS-TE, such as:

- Fast re-routing and restoration in case of link/ node failure
- Bandwidth guarantee
- Explicit path setting along with off-line computation

MPLS supports point-to-point path. However, in order to use MPLS for multicast service, MPLS has to be extended to handle point-to-multipoint paths. A reliable solution to signal Point-to-Multipoint (P2MP) label switched paths(LSP) is the Point-to-Multipoint TE LSP. This solution uses the Resource Reservation Protocol-Traffic Engineering (RSVP-TE) extension as the signaling protocol for establishing P2MP TE LSPs.

Point to Multipoint LSP(P2MP)

P2MP LSP is unidirectional. In case of native IP multicast, the multicast forwarding always has to perform an acceptance check. This check ensures all multicast packets undergo a RPF check to ensure that the packets have arrived on the correct interface in the direction of the source. However, the acceptance check with MPLS forwarding may be different in case of an unicast or upstream label.

Depending on the multicast signaling protocol, the labeled packet may require an additional L3 lookup at the P and PE routers in order to forward the multicast packet to the physical interfaces according to multicast routing. In this case, the incoming P2MP LSP as the incoming interface for the received multicast packet must also be available to the multicast forwarding plane during the L3 lookup. For more details on RSVP-TE and P2MP LSP, refer the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers*

Multicast Routing Protocol support for P2MP

All multicast routing protocols support P2MP TE LSP. At ingress node, a multicast protocol must make a mapping between the multicast traffic and the P2MP TE LSP with the configuration of static-join. At egress node, the multicast protocol must conduct a special RPF check for the multicast packet which is received from MPLS core and forward it to the customer facing interface. The RPF check is based on the configuration of static-rpf. These multicast groups which are forwarded over the P2MP TE LSPs can be specified with the static-rpf configuration in case of PIM-SSM.

Enabling Multicast Forwarding Over Tunnel Interface (at Ingress Node)

This configuration is used for allowing the forwarding of the multicast packet over the specified interface.

SUMMARY STEPS

1. **configure**
2. **multicast-routing**
3. **address-family** {**ipv4**|**ipv6**}

4. **interface tunnel-mte** *range*
5. **enable** | **disable**
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode.
Step 3	**address-family** {**ipv4**|**ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 or ipv6 address-family submode.
Step 4	**interface tunnel-mte** *range* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface tunnel-mte 100`	Specify the range. The range is 0 to 65535.
Step 5	**enable** | **disable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# enable`	If enable is set, MFIB forwards multicast packets over the interface. If disable is set, MFIB stops forwarding multicast packets over the interface.
Step 6	**commit**	

P2MP configurations at egress node and bud node

Configuring Static Reverse Path Forwarding (RPF)

SUMMARY STEPS

1. **configure**
2. **multicast-routing**
3. **address-family** {**ipv4** | **ipv6**}

4. **static-rpf** *address range prefix*
5. **mpls address**
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode.
Step 3	**address-family** {**ipv4** | **ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 (or ipv6) address-family submode.
Step 4	**static-rpf** *address range prefix* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# static-rpf 10.1.1.1 32`	Enter the source and prefix length.
Step 5	**mpls address** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# mpls 10.2.2.2`	Enter the source PE address of the MPLS P2MP tunnel.
Step 6	**commit**	

Configuring Core Tree Protocol

SUMMARY STEPS

1. **configure**
2. **multicast-routing**
3. **address-family** {**ipv4** | **ipv6**}
4. **core-tree-protocol rsvp-te group-list** *name*
5. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing** **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing`	Enters multicast routing configuration mode.
Step 3	**address-family** {**ipv4** | **ipv6**} **Example:** `RP/0/RSP0/CPU0:router(config-mcast)# address-family ipv4`	Enters ipv4 (or ipv6)address-family submode.
Step 4	**core-tree-protocol rsvp-te group-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# core-tree-protocol rsvp-te group-list acl1`	Enters the core-tree-protocol configuration mode.
Step 5	**commit**	

Configuring IGMP VRF Override

This process consists of the following tasks:

Specifying VRF definition

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-target 1:1**
5. **export route-target 1:1**
6. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config)# vrf name1`	Enters the VRF configuration sub mode.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	AFI configuration for IPv4. This is supported on unicast topologies only.
Step 4	**import route-target 1:1** **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 1:1`	Enables VRF import.
Step 5	**export route-target 1:1** **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 1:1`	Enables VRF export.
Step 6	**commit**	

Enabling Multicast Routing on default and non-default VRFs

This task enables multicast routing and forwarding on all new and existing interfaces. For the VRF override feature, multicast routing needs to be enabled on both, the default and the non-default VRFs.

SUMMARY STEPS

1. **configure**
2. **multicast-routing vrf** [*vrf-name* | *default]*
3. **interface** {*type interface-path-id* | **all**} **enable**
4. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**multicast-routing vrf** [*vrf-name* | *default]* **Example:** `RP/0/RSP0/CPU0:router(config)# multicast-routing vrf green`	Enters multicast configuration mode for the specified VRF. Note that the default configuration mode for multicast routing is default vrf (if the non-default VRF name is not specified).
Step 3	**interface** {*type interface-path-id* | **all**} **enable** **Example:** `RP/0/RSP0/CPU0:router(config-mcast-green)# interface all enable`	Enables multicast routing and forwarding on one or on all new and existing interfaces.
Step 4	**commit**	

Configuring an Interface for a Non-default VRF Instance

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *address mask*
5. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface tengige 0/1/0/0`	Enters PIM address-family IPv4 submode.

	Command or Action	Purpose
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-if)# vrf name1`	Sets the VRF for the interface.
Step 4	**ipv4 address** *address mask* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.0.0.0`	Sets the IPv4 address for the interface.
Step 5	**commit**	

Configuring route-policy

SUMMARY STEPS

1. **configure**
2. **route-policy** *policy-name*
3. **set rpf-topology vrf default**
4. **end-policy**
5. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**route-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy policy1`	Defines a route policy.
Step 3	**set rpf-topology vrf default** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# set rpf-topology vrf default`	Sets the PIM RPF topology attributes for the default VRF.

	Command or Action	Purpose
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	Ends the route-policy definition configuration.
Step 5	**commit**	

Associating a route policy to PIM configuration for the VRF receiving IGMP reports

SUMMARY STEPS

1. **configure**
2. **router pim vrf** *vrf-name* **address-family ipv4**
3. **rpf-topology route-policy** *policy-name*
4. **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router pim vrf** *vrf-name* **address-family ipv4**	Enters PIM address-family IPv4 submode.
Step 3	**rpf-topology route-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# rpf-topology route-policy policy1`	Associates a previously defined route-policy with the non-default VRF that receives the IGMP reports.
Step 4	**commit**	

MVPN GRE over PWHE with CSI

MVPN GRE over PWHE is supported on CSI interface.

The Multicast VPN (MVPN) feature provides the ability to support multicast over a Layer 3 VPN. Whereas, Pseudowire Headend (PWHE) allows termination of access pseudowires (PWs) into a Layer 3 (VRF or global) domain or into a Layer 2 domain.

Restrictions

- Only SSM is supported on PE-CE multicast
- Only IPv4 is supported on PE-CE multicast over PWHE interfaces
- Only IPv4 SM is supported on provider multicast
- Does not support SM on PE-CE multicast
- Does not support ISSU
- IPv6 is not supported

Configuration Example

```
interface PW-Ether1 vrf vrf1
 ipv4 address 192.0.2.1 255.255.255.252
 ipv6 address 2001:DB8:1::1/32
 attach generic-interface-list Bundle311
!
```

Multicast Traceroute v2

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Mtrace version 2 - Traceroute facility for IP Multicast	Release 7.7.1	This feature enables you to: • Trace the path a packet would take from a source to the receiver. • To isolate packet-loss problems. • To isolate configuration problems. Mtrace version2, or Mtrace2, is an IP multicast traceroute facility which allows the tracing of an IP multicast routing path. Based on RFC 8487, Mtrace2 is usually initiated from an Mtrace2 client by sending a Mtracev2 query to a Last-Hop Router (LHR) or to a Rendezvous Point (RP). This feature introduces the **mtrace2 IPv4** and **mtrace2 IPv6** commands.

Mtrace version2, or Mtrace2 is a diagnostic tool to trace the route taken by a packet on an IP mutlicast network. Mtrace2 is usually initiated from a Mtrace2 client by sending a Mtrace2 query to a Last-Hop Router (LHR) or a Rendezvous Point (RP). From the LHR/RP receiving the query, the tracing is directed towards a specified source if a source address is specified, and a source-specific state exists on the receiving router. If a source address is not specified, or if a source-specific state does not exist on a receiving router, the tracing is directed towards the RP for the specified group address. Mtrace2 also provides additional information such as the packet rates and losses as well as other diagnostic information.

This feature has three main benefits:

- To trace the path that a packet takes from a source to a receiver.
- To isolate packet-loss problems.
- To isolate configuration problems.

The advantages Mtrace2 exhibits over previous versions of Mtrace are:

- It is based on RFC 8487.
- It is UDP based, and not IGMP based.
- Though IANA (Internet Assigned Numbers Authority) has allocated a specific UDP port number for this feature, the source port can be assigned by the local host system as well.
- IANA (Internet Assigned Numbers Authority) has allocated a specific UDP port number for this feature.
- It supports both IPv4 and IPv6.

Mtrace2 Packet Flow Over a Network

All Mtrace2 messages are UDP packets that are encoded in the Type/Length/Value (TLV) format. The first TLV of a message is a message header TLV which specifies the type of message and additional context of information that is required to process the message and to parse subsequent TLVs in the message. Subsequent TLVs in a message are referred to as Blocks and are appended after the header TLV to provide additional information associated with the message.

A Mtrace2 client creates a query packet and forwards this to a LHR. The LHR modifies the Type field of the query TLV to turn it into a request before appending a standard response block and forwarding it upstream to a First-Hop Router (FHR), which is a router that is connected to the Mtrace2 client. Each node in the path attaches its response to this packet. If a node finds an error, it sets an appropriate forwarding error code to the packet. Once the request packet reaches the FHR, the FHR converts it to a reply and sends it back to the Mtrace2 client.

Table 5: Mtrace2 Defined TLV Types

Code	Type
0x00	Reserved
0x01	Mtrace2 Query
0x02	Mtrace2 Request
0x03	Mtrace2 Reply
0x04	Standard Response Block

Code	Type
0x05	Augmented Response Block
0x06	Extended Query Block

Mtrace2 Guidelines

- For IPv4, Mtrace2 messages must not be fragmented. Mtrace2 clients, LHRs and RPs must set the IP header *do-not-fragment (DF)* bit for all Mtrace2 messages.
- For IPv6, the packet size for the Mtrace2 messages must not exceed 1280 bytes, which is the smallest Maximum Transmission Unit (MTU) for an IPv6 interface.
- Mtrace2 does not support mixed IPv4 and IPv6 packets.

Configuration Examples for Implementing Multicast Routing on Software

This section provides the following configuration examples:

DNS-based SSM Mapping: Example

The following example illustrates DNS-based SSM Mapping configuration.

```
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback5
  maximum disable
  interface all enable
  accounting per-prefix
 !
 address-family ipv6
  nsf
  maximum disable
  interface all enable
  accounting per-prefix
 !
 vrf p11_1
  address-family ipv4
   ssm range ssm_acl
   interface all enable
   mdt default ipv4 235.1.1.1
  !



ipv4 access-list ssm_acl
 10 permit ipv4 225.11.1.0 0.0.0.255 any
 20 permit ipv4 225.11.2.0 0.0.0.255 any
!

router mld
```

```
 vrf p11_1
  ssm map query dns


router igmp

 !
 vrf p11_1
  ssm map query dns
 !


domain vrf p11_1 name-server 100.1.1.2
domain multicast cisco.com
domain name-server 10.10.10.1
```

Calculating Rates per Route: Example

The following example illustrates output from hardware counters based on rate per route for a specific source and group address location:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# multicast-routing vrf vpn12 address-family ipv4
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# rate-per-route
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# interface all enable
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# accounting per-prefix
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# commit
RP/0/RSP0/CPU0:router(config-mcast-default-ipv4)# exit
RP/0/RSP0/CPU0:router(config-mcast)# exit
RP/0/RSP0/CPU0:router(config)# exit
RP/0/RSP0/CPU0:router# show mfib route rate

IP Multicast Forwarding Rates Source Address, Group Address HW Forwarding Rates: bps In/pps
 In/bps Out/pps Out

(*,224.0.0.0/24)
bps_in /pps_in /bps_out /pps_out
N/A / N/A / N/A / N/A

(*,224.0.1.39)
bps_in /pps_in /bps_out /pps_out
N/A / N/A / N/A / N/A

(*,224.0.1.40)
bps_in /pps_in /bps_out /pps_out
N/A / N/A / N/A / N/A

(*,232.0.0.0/8)
bps_in /pps_in /bps_out /pps_out
N/A / N/A / N/A / N/A
(10.0.70.2,225.0.0.0)
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.1)
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.2)
```

```
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.3)
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.4)
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.5)
bps_in /pps_in /bps_out /pps_out
22649 / 50 / 22951 / 50

(10.0.70.2,225.0.0.6)
bps_in /pps_in /bps_out /pps_out
```

Preventing Auto-RP Messages from Being Forwarded on Software: Example

This example shows that Auto-RP messages are prevented from being sent out of the GigabitEthernet interface 0/3/0/0. It also shows that access list 111 is used by the Auto-RP candidate and access list 222 is used by the **boundary** command to contain traffic on GigabitEthernet interface 0/3/0/0.

```
ipv4 access-list 111
 10 permit 224.1.0.0 0.0.255.255 any
 20 permit 224.2.0.0 0.0.255.255 any
!
!Access list 111 is used by the Auto-RP candidate.
!
ipv4 access-list 222
 10 deny any host 224.0.1.39
 20 deny any host 224.0.1.40
!
!Access list 222 is used by the boundary command to contain traffic (on
GigabitEthernet0/3/0/0) that is sent to groups 224.0.1.39 and 224.0.1.40.
!
router pim
 auto-rp mapping-agent loopback 2 scope 32 interval 30
 auto-rp candidate-rp loopback 2 scope 15 group-list 111 interval 30
multicast-routing
 interface GigabitEthernet0/3/0/0
 boundary 222
!
```

Inheritance in MSDP on Software: Example

The following MSDP commands can be inherited by all MSDP peers when configured under router MSDP configuration mode. In addition, commands can be configured under the peer configuration mode for specific peers to override the inheritance feature.

- **connect-source**
- **sa-filter**
- **ttl-threshold**

If a command is configured in both the router msdp and peer configuration modes, the peer configuration takes precedence.

In the following example, MSDP on Router A filters Source-Active (SA) announcements on all peer groups in the address range 226/8 (except IP address 172.16.0.2); and filters SAs sourced by the originator RP 172.16.0.3 to 172.16.0.2.

MSDP peers (172.16.0.1, 172.16.0.2, and 172.17.0.1) use the loopback 0 address of Router A to set up peering. However, peer 192.168.12.2 uses the IPv4 address configured on the GigabitEthernet interface to peer with Router A.

Router A

```
!
ipv4 access-list 111
 10 deny ip host 172.16.0.3 any
 20 permit any any
!

ipv4 access-list 112
 10 deny any 226.0.0.0 0.255.255.255
 30 permit any any
!
router msdp
 connect-source loopback 0
 sa-filter in rp-list 111
 sa-filter out rp-list 111
 peer 172.16.0.1
!
peer 172.16.0.2
 sa-filter out list 112
!
peer 172.17.0.1
!
peer 192.168.12.2
 connect-source GigabitEthernet0/2/0/0
!
```

MSDP-VRF: Example

This is an example, where peer 10.0.0.4 is configured in the VRF context for vrf1.

```
config
 router msdp
  vrf vrf1
    peer 10.0.0.4
  exit
end
!
```

MoFRR Provider Edge Configuration: Example

The following example shows Tail PE configuration details. Here, joins for (10.0.0.4, 232.1.1.1) will be sent as joins for (10.0.0.4, 232.1.1.1) and joins for (3.3.1.1, 232.1.1.1).

```
config
router pim
mofrr
flow mofrr_acl
join source 10.0.0.4 to 3.3.0.0 masklen 16
rpf-vector 10.0.0.4 masklen 16 10.1.1.1 20.1.1.1
rpf-vector 3.3.1.1 masklen 16 30.1.1.1 40.1.1.1
ipv4 access-list extended mofrr_acl
! 10 permit ipv4 any 232.1.1.1
```

Configuring Route Policy for Static RPF: Example

```
router static
 address-family ipv4 multicast
  202.93.192.74 /32 202.40.148.11

!
route-policy pim-policy
 set rpf-topology ipv4 multicast topology default

end-policy
!
router pim
 address-family ipv4
  rpf topology route-policy pim-policy
```

Configuring IPv4 Multicast VPN: Example

Cisco ASR 9000 Series Routers support only IPv4 addressing.

This end-to-end configuration example shows how to establish a multicast VPN topology (Figure 21: Topology in CE4PE1PE2 CE3MVPN Configuration, on page 218), using two different routing protocols (OSPF or BGP) to broadcasting traffic between customer-edge(CE) routers and provider-edge (PE) routers:

Figure 21: Topology in CE4PE1PE2 CE3MVPN Configuration

CE4---------- PE1 ---------- PE2 ---------- CE3

For more configuration information, see the Configuring Multicast VPN, on page 140 of this module and also related configuration information in *Routing Configuration Guide for Cisco ASR 9000 Series Routers* .

Configuring MVPN to Advertise Routes Between the CE and the PE Using OSPF: Example

PE1:

```
!
vrf vpn1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
interface Loopback0
```

```
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback1
 vrf vpn1
 ipv4 address 2.2.2.2 255.255.255.255
!
interface GigabitEthernet0/5/0/0
 vrf vpn1
 ipv4 address 101.1.1.1 255.255.255.0
!
interface TenGigE0/6/0/0
 ipv4 address 12.1.1.1 255.255.255.0
!
mpls ldp
 router-id 10.0.0.4
 interface TenGigE0/6/0/0
 !
!
multicast-routing
 vrf vpn1 address-family ipv4
  mdt data 233.1.0.0/16 threshold 3
  mdt default ipv4 232.1.1.1
  rate-per-route
  interface all enable
  accounting per-prefix
 !
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
!
router bgp 100
 bgp router-id 10.0.0.4
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv4 mdt
 !
 neighbor 9.9.9.9
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv4 mdt
  !
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute ospf 1
  !
 !
!
router ospf 1
 vrf vpn1
  router-id 2.2.2.2
  redistribute bgp 100
  area 0
   interface Loopback1
```

```
   !
   interface GigabitEthernet0/5/0/0
   !
  !
 !
!
router ospf 100
 router-id 10.0.0.4
 area 0
  interface Loopback0
  !
  interface TenGigE0/6/0/0
  !
 !
!
router pim vrf vpn1 address-family ipv4
 rp-address 2.2.2.2
 log neighbor changes
!
router pim vrf default address-family ipv4
 rp-address 10.0.0.4
!
end
```

PE2:

```
!
vrf vpn1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
interface Loopback0
 ipv4 address 9.9.9.9 255.255.255.255
!
interface Loopback1
 vrf vpn1
 ipv4 address 10.10.10.10 255.255.255.255
!
interface GigabitEthernet0/2/2/7
 vrf vpn1
 ipv4 address 122.1.1.1 255.255.255.0
 negotiation auto
!
interface TenGigE0/3/0/0
 ipv4 address 12.1.1.2 255.255.255.0
!
mpls ldp
 router-id 9.9.9.9
 interface TenGigE0/3/0/0
 !
!
multicast-routing
 vrf vpn1 address-family ipv4
  mdt data 233.1.0.0/16 threshold 3
  mdt default ipv4 232.1.1.1
  rate-per-route
```

```
  interface all enable
  accounting per-prefix
 !
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
!
router bgp 100
 bgp router-id 9.9.9.9
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv4 mdt
 !
 neighbor 10.0.0.4
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv4 mdt
  !
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute ospf 1
  !
 !
!
router ospf 1
 vrf vpn1
  router-id 10.10.10.10
  redistribute bgp 100
  area 0
   interface Loopback1
   !
   interface GigabitEthernet0/2/2/7
   !
  !
 !
!
router ospf 100
 router-id 9.9.9.9
 area 0
  interface Loopback0
  !
  interface TenGigE0/3/0/0
  !
 !
!
router pim vrf vpn1 address-family ipv4
 rp-address 2.2.2.2
!
router pim vrf default address-family ipv4
 rp-address 10.0.0.4
!
end
```

CE4:

For information about configuring the CE router, using Cisco IOS software, see the appropriate Cisco IOS software configuration documentation.

```
!
interface Loopback0
 ipv4 address 101.101.101.101 255.255.255.255
!
interface GigabitEthernet0/0/0/0
 ipv4 address 101.1.1.2 255.255.255.0
!
interface GigabitEthernet0/0/0/3
 ipv4 address 11.1.1.1 255.255.255.0
!
multicast-routing
 address-family ipv4
  interface all enable
 !
!
router ospf 1
 router-id 101.101.101.101
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
  !
  interface GigabitEthernet0/0/0/3
  !
 !
!
router pim vrf default address-family ipv4
 rp-address 2.2.2.2
 interface Loopback0
 !
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/3
 !
!
end
```

CE3:

For information about configuring the CE router, using Cisco IOS software, see the appropriate Cisco IOS software configuration documentation.

```
interface Loopback0
 ipv4 address 122.122.122.122 255.255.255.255
!

interface GigabitEthernet0/1/3/0
 ipv4 address 22.1.1.1 255.255.255.0
!

interface GigabitEthernet0/2/3/0
 ipv4 address 122.1.1.2 255.255.255.0

multicast-routing
 address-family ipv4
  interface all enable
```

```
!
router ospf 1
 router-id 122.122.122.122
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/1/3/0
  !
  interface GigabitEthernet0/2/3/0
  !
 !
!
router pim vrf default address-family ipv4
 rp-address 2.2.2.2
 interface Loopback0
 !
 interface GigabitEthernet0/1/3/0
 !
 interface GigabitEthernet0/2/3/0
 !
!
end
```

Configuring MVPN to Advertise Routes Between the CE and the PE Using BGP: Example

PE1:

```
vrf vpn1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback1
 vrf vpn1
 ipv4 address 2.2.2.2 255.255.255.255
!
interface GigabitEthernet0/5/0/0
 vrf vpn1
 ipv4 address 101.1.1.1 255.255.255.0
!
interface TenGigE0/6/0/0
 ipv4 address 12.1.1.1 255.255.255.0
!
mpls ldp
 router-id 10.0.0.4
 interface TenGigE0/6/0/0
 !
!
multicast-routing
 vrf vpn1 address-family ipv4
  mdt data 233.1.0.0/16 threshold 3
  mdt default ipv4 232.1.1.1
  rate-per-route
```

```
  interface all enable
  accounting per-prefix
 !
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
!
!
route-policy pass-all
  pass
end-policy
!
router bgp 100
 bgp router-id 10.0.0.4
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv4 mdt
 !
 neighbor 9.9.9.9
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv4 mdt
  !
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 101.1.1.2
   remote-as 400
   address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
   !
  !
 !
!
router ospf 100
 router-id 10.0.0.4
 area 0
  interface Loopback0
  !
  interface TenGigE0/6/0/0
  !
 !
!
router pim vrf vpn1 address-family ipv4
 rp-address 2.2.2.2
 log neighbor changes
!
router pim vrf default address-family ipv4
 rp-address 10.0.0.4
!
```

```
end
```

PE2:

```
!
vrf vpn1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
interface Loopback0
 ipv4 address 9.9.9.9 255.255.255.255
!
interface Loopback1
 vrf vpn1
 ipv4 address 10.10.10.10 255.255.255.255
!
interface GigabitEthernet0/2/2/7
 vrf vpn1
 ipv4 address 122.1.1.1 255.255.255.0
 negotiation auto
!
interface TenGigE0/3/0/0
 ipv4 address 12.1.1.2 255.255.255.0
!
mpls ldp
 router-id 9.9.9.9
 interface TenGigE0/3/0/0
 !
!
multicast-routing
 vrf vpn1 address-family ipv4
  mdt data 233.1.0.0/16 threshold 3
  mdt default ipv4 232.1.1.1
  rate-per-route
  interface all enable
  accounting per-prefix
 !
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
!
!
route-policy pass-all
  pass
end-policy
!
router bgp 100
 bgp router-id 9.9.9.9
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv4 mdt
```

```
 !
 neighbor 10.0.0.4
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv4 mdt
  !
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 122.1.1.2
   remote-as 500
   address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
   !
  !
 !
!
router ospf 100
 router-id 9.9.9.9
 area 0
  interface Loopback0
  !
  interface TenGigE0/3/0/0
  !
 !
!
router pim vrf vpn1 address-family ipv4
 rp-address 2.2.2.2
!
router pim vrf default address-family ipv4
 rp-address 10.0.0.4
!
end
```

CE4:

For information about configuring the CE router, using Cisco IOS software, see the appropriate Cisco IOS software configuration documentation.

```
interface Loopback0
 ipv4 address 101.101.101.101 255.255.255.255
!
interface GigabitEthernet0/0/0/0
 ipv4 address 101.1.1.2 255.255.255.0
!
interface GigabitEthernet0/0/0/3
 ipv4 address 11.1.1.1 255.255.255.0
!
multicast-routing
 address-family ipv4
  interface all enable
 !
!
!
```

```
route-policy pass-all
  pass
end-policy
!
router bgp 400
 bgp router-id 101.101.101.101
 address-family ipv4 unicast
  redistribute connected
 !
 neighbor 101.1.1.1
  remote-as 100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
!
router pim vrf default address-family ipv4
 rp-address 2.2.2.2
 interface Loopback0
 !
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/3
 !
!
end
```

CE3:

For information about configuring the CE router, using Cisco IOS software, see the appropriate Cisco IOS software configuration documentation.

```
interface Loopback0
 ipv4 address 122.122.122.122 255.255.255.255
!

interface GigabitEthernet0/1/3/0
 ipv4 address 22.1.1.1 255.255.255.0
!

interface GigabitEthernet0/2/3/0
 ipv4 address 122.1.1.2 255.255.255.0

multicast-routing
 address-family ipv4
  interface all enable
!
!
!
route-policy pass-all
  pass
end-policy
!
router bgp 500
 bgp router-id 122.122.122.122
 address-family ipv4 unicast
  redistribute connected
 !
 neighbor 122.1.1.1
  remote-as 100
  address-family ipv4 unicast
```

```
   route-policy pass-all in
   route-policy pass-all out
  !
 !
!
!
router pim vrf default address-family ipv4
 rp-address 2.2.2.2
 interface Loopback0
 !
 interface GigabitEthernet0/1/3/0
 !
 interface GigabitEthernet0/2/3/0
 !
!
end
```

Configuration Examples for MVPN Profiles

This section provides profile-wise configuration examples for the various MVPN profiles.

Configuration Examples for Inband mLDP Profiles

Profile-6: VRF Inband mLDP

```
router bgp 100
 mvpn
!
multicast-routing
 vrf v61
 address-family ipv4
    mdt source Loopback0
    mdt mtu 1600
    mdt mldp in-band-signaling ipv4
    interface all enable
   !
  address-family ipv6
    mdt mtu 1600
    mdt mldp in-band-signaling ipv4
   interface all enable
 !
!
router pim
 vrf v61
  address-family ipv4
   rpf topology route-policy mldp-inband
  !
  address-family ipv6
   rpf topology route-policy mldp-inband
  !
!
route-policy mldp-inband
  set core-tree mldp-inband
end-policy
!
```

Profile-7: Global Inband mLDP

```
multicast-routing
 address-family ipv4
   mdt source Loopback0
   mdt mldp in-band-signaling ipv4
```

```
   ssm range Global-SSM-Group
  interface all enable
!
 address-family ipv6
  mdt source Loopback0
  mdt mldp in-band-signaling ipv4
  ssm range Global-SSM-Group-V6
  interface all enable
!
 router pim
  address-family ipv4
   rpf topology route-policy mldp-inband
  !
  address-family ipv6
   rpf topology route-policy mldp-inband
 !
!
route-policy mldp-inband
  set core-tree mldp-inband
end-policy
!
```

Configuration Examples for P2MP-TE profiles

Profile-8: Global Static P2MP-TE

```
interface Loopback0
 ipv4 address 200.200.1.1 255.255.255.255
!
multicast-routing
 address-family ipv4
  mdt source Loopback0
   ssm range Global-SSM-Group
  interface all enable
!
 address-family ipv6
  mdt source Loopback0
  ssm range Global-SSM-Group-V6
  interface all enable
!
router igmp
 interface tunnel-mte1
   static-group 228.1.1.1 2.2.2.1
 !
router mld
 interface tunnel-mte1
  static-group ff3e:0:228::1 2001:2:2:2::1
```

Profile-10: VRF static P2MP-TE with BGP-AD

```
!
multicast-routing
 mdt source Loopback0
 vrf v101
  address-family ipv4
   mdt static p2mp-te tunnel-mte10
    interface all enable
   bgp auto-discovery pim
  !
!
router igmp
 vrf v101
  interface tunnel-mte10
```

```
   static-group 227.101.1.1 101.7.1.1
 !
!
```

Profile-18: Rosen Static P2MP-TE with BGP-AD and PIM signaling

```
multicast-routing
 mdt source Loopback0
 vrf v181
  address-family ipv4
    mdt default p2mp-te static tunnel-mte18
   interface all enable
   bgp auto-discovery pim
   !
  address-family ipv6
   mdt default p2mp-te static tunnel-mte181
    interface all enable
   bgp auto-discovery pim
 !
!
router pim
 vrf v181
  address-family ipv4
   rpf topology route-policy p2mp-te-default
    !
  address-family ipv6
   rpf topology route-policy p2mp-te-default
  !
!
route-policy p2mp-te-default
  set core-tree p2mp-te-default
end-policy
!
```

Profile-16: Rosen Static P2MP-TE with BGP-Ad and BGP signaling

```
!
multicast-routing
 mdt source Loopback0
 vrf v161
  address-family ipv4
   mdt default p2mp-te static tunnel-mte16
    interface all enable
   bgp auto-discovery pim
   !
  address-family ipv6
   mdt default p2mp-te static tunnel-mte161
    interface all enable
   bgp auto-discovery pim
 !
!
router pim
 vrf v161
  address-family ipv4
   rpf topology route-policy p2mp-te-default
   mdt c-multicast-routing bgp
   !
  address-family ipv6
   rpf topology route-policy p2mp-te-default
   mdt c-multicast-routing bgp
 !
!
route-policy p2mp-te-default
```

```
  set core-tree p2mp-te-default
end-policy
!
```

Profile-20: Default MDT - P2MP-TE - BGP-AD - PIM

```
multicast-routing
 mdt source Loopback0
vrf p20_vrf1
 address-family ipv4 unicast
  import route-target
   20:1
  !
  export route-target
   20:1
  !
route-policy rpf-for-p20_vrf1
  set core-tree p2mp-te-default
end-policy
!
vrf p20_vrf1
  rd 20:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
rsvp
 interface Bundle-Ether1
  bandwidth
 !
 !
 vrf p20_vrf1
  address-family ipv4
   mdt source Loopback0
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery p2mp-te
   !
   mdt default p2mp-te
  !
 !
vrf p20_vrf1
  address-family ipv4
   rpf topology route-policy rpf-for-p20_vrf1
   rp-address 100.1.20.1 static_p20vrf1_v4
   log neighbor changes
   interface GigabitEthernet0/0/0/0.202
    enable
   !
   interface GigabitEthernet0/0/0/0.203
    enable
   !
   interface GigabitEthernet0/0/0/1.202
    enable
   !
   interface GigabitEthernet0/0/0/1.203
    enable
   !
```

Configuration examples for Partitioned mLDP profiles

Profile-2: Partitioned mLDP MP2MP without BGP-AD

```
router bgp 100
  mvpn
!
multicast-routing
 vrf v21
  address-family ipv4
    mdt mtu 1600
    mdt partitioned mldp ipv4 mp2mp
    interface all enable
   !
  address-family ipv6
    mdt mtu 1600
    mdt partitioned mldp ipv4 mp2mp
    interface all enable
 !
!
router pim
 vrf v21
  address-family ipv4
   rpf topology route-policy mldp-partitioned-mp2mp
  !
  address-family ipv6
   rpf topology route-policy mldp-partitioned-mp2mp
 !
!
route-policy mldp-partitioned-mp2mp
  set core-tree mldp-partitioned-mp2mp
end-policy
!
```

Profile-4: Partitioned mLDP MP2MP with BGP-AD and PIM signaling

```
!
multicast-routing
 mdt source Loopback0
 vrf v41
  address-family ipv4
    mdt mtu 1600
mdt partitioned mldp ipv4 mp2mp
mdt data 255
   interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
mdt partitioned mldp ipv4 mp2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
   !
!
router pim
 vrf v41
  address-family ipv4
   rpf topology route-policy mldp-partitioned-mp2mp
  !
  address-family ipv6
   rpf topology route-policy mldp-partitioned-mp2mp
 !
!
```

```
route-policy mldp-partitioned-mp2mp
  set core-tree mldp-partitioned-mp2mp
end-policy
!
```

Profile-15: Partitioned mLDP MP2MP with BGP-AD and BGP signaling

```
!
multicast-routing
 mdt source Loopback0
 vrf v151
  address-family ipv4
    mdt mtu 1600
mdt partitioned mldp ipv4 mp2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
mdt partitioned mldp ipv4 mp2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
 !
!
router pim
 vrf v151
  address-family ipv4
   rpf topology route-policy mldp-partitioned-mp2mp
   mdt c-multicast-routing bgp
  !
  address-family ipv6
   rpf topology route-policy mldp-partitioned-mp2mp
   mdt c-multicast-routing bgp
  !
!
route-policy mldp-partitioned-mp2mp
  set core-tree mldp-partitioned-mp2mp
end-policy
!
```

Profile-5: Partitioned mLDP P2MP with BGP-AD and PIM signaling

```
!
multicast-routing
 mdt source Loopback0
vrf v51
  address-family ipv4
    mdt mtu 1600
mdt partitioned mldp ipv4 p2mp
mdt data 255
   interface all enable
   bgp auto-discovery mldp
  !
  address-family ipv6
    mdt mtu 1600
mdt partitioned mldp ipv4 p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
  !
!
router pim
 vrf v51
```

```
  address-family ipv4
   rpf topology route-policy mldp-partitioned-p2mp
  !
  address-family ipv6
   rpf topology route-policy mldp-partitioned-p2mp
 !
!
route-policy mldp-partitioned-p2mp
  set core-tree mldp-partitioned-p2mp
end-policy
```

Profile-14: Partitioned mLDP P2MP with BGP-AD and BGP siganling

```
!
multicast-routing
 mdt source Loopback0
 vrf v141
  address-family ipv4
    mdt mtu 1600
    mdt partitioned mldp ipv4 p2mp
    mdt data 255
   interface all enable
   bgp auto-discovery mldp
  !
  address-family ipv6
    mdt mtu 1600
mdt partitioned mldp ipv4 p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
 !
!
router pim
 vrf v141
  address-family ipv4
   rpf topology route-policy mldp-partitioned-p2mp
   mdt c-multicast-routing bgp
   !
  address-family ipv6
   rpf topology route-policy mldp-partitioned-p2mp
   mdt c-multicast-routing bgp
 !
!
route-policy mldp-partitioned-p2mp
  set core-tree mldp-partitioned-p2mp
end-policy
!
```

Configuration Examples for Rosen-mGRE profiles

Profile-0: Rosen mGRE with MDT SAFI

```
router bgp 100
 address-family ipv4 mdt
 !
 neighbor X.X.X.X   < -----RR or Remote PE ip address
 address-family ipv4 mdt
 !
!

multicast-routing
 address-family ipv4
  mdt source Loopback0
```

```
  interface all enable
!
 address-family ipv6
   mdt source Loopback0
  interface all enable
!
 vrf v1
  address-family ipv4
   mdt mtu 1600
   mdt data 231.1.1.2/32
   mdt default ipv4 231.1.1.1
   interface all enable
  !
  address-family ipv6
   mdt mtu 1600
   mdt data 231.1.1.2/32
   mdt default ipv4 231.1.1.1
   interface all enable
 !
!
```

Profile-3: Rosen mGRE with BGP-AD and PIM signaling

```
router bgp 100
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
 neighbor X.X.X.X   < -----RR or Remote PE ip address
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
 vrf v31
  rd 100:31
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
!
multicast-routing
 mdt source Loopback0
 vrf v31
  address-family ipv4
   mdt mtu 1600
   mdt data 232.31.1.2/32
   mdt default ipv4 232.31.1.1
   interface all enable
   bgp auto-discovery pim
  !
  address-family ipv6
   mdt mtu 1600
   mdt data 232.31.1.2/32
   mdt default ipv4 232.31.1.1
   interface all enable
   bgp auto-discovery pim
 !
!
```

Profile-11: Rosen mGRE with BGP-AD and BGP signaling

```
router bgp 100
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
 neighbor X.X.X.X   < -----RR or Remote PE ip address
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
 vrf v111
  rd 100:111
  address-family ipv4 mvpn
  !
  address-family ipv6 mvpn
  !
 !
!
multicast-routing
 mdt source Loopback0
 vrf v111
  address-family ipv4
   mdt mtu 1600
   mdt data 232.111.1.2/32
   mdt default ipv4 232.111.1.1
   interface all enable
   bgp auto-discovery pim
  !
  address-family ipv6
   mdt mtu 1600
   mdt data 232.111.1.2/32
   mdt default ipv4 232.111.1.1
   interface all enable
   bgp auto-discovery pim
 !
!
router pim
 vrf v111
  address-family ipv4
   mdt c-multicast-routing bgp
  !
  address-family ipv6
   mdt c-multicast-routing bgp
 !
```

Configuration Examples for Rosen mLDP profiles

Profile-1: Rosen mLDP with M2MP without BGP-AD

```
vrf v11
 vpn id 100:11
!
router bgp 100
  mvpn
!
multicast-routing
 mdt source Loopback0
 vrf v11
  address-family ipv4
    mdt mtu 1600
    mdt default mldp ipv4 100.100.1.1
```

```
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
  !
  address-family ipv6
    mdt mtu 1600
    mdt default mldp ipv4 100.100.1.1
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
 !
!
router pim
 vrf v11
  address-family ipv4
   rpf topology route-policy rosen-mldp
  !
  address-family ipv6
   rpf topology route-policy rosen-mldp
 !
!
route-policy rosen-mldp
  set core-tree mldp-default
end-policy!
```

Profile-9: Rosen mLDP MP2MP with BGP-AD and PIM signaling

```
vrf v91
 vpn id 100:91

!
!
multicast-routing
 mdt source Loopback0
 vrf v91
  address-family ipv4
    mdt mtu 1600
    mdt default mldp ipv4 100.100.1.1
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
   mdt default mldp ipv4 100.100.1.1
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
   bgp auto-discovery mldp
  !
!
router pim
 vrf v91
  address-family ipv4
   rpf topology route-policy rosen-mldp
  !
  address-family ipv6
   rpf topology route-policy rosen-mldp
  !
 !
route-policy rosen-mldp
  set core-tree mldp-default
end-policy
```

Profile-13: Rosen mLDP MP2MP with BGP-AD and BGP signaling

```
vrf v131
 vpn id 100:131
 !

!
multicast-routing
 mdt source Loopback0
 vrf v131
  address-family ipv4
    mdt mtu 1600
    mdt default mldp ipv4 100.100.1.1
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
    mdt default mldp ipv4 100.100.1.1
   mdt default mldp ipv4 100.100.1.2
   mdt data 255
   interface all enable
   bgp auto-discovery mldp
  !
!
router pim
 vrf v131
  address-family ipv4
   rpf topology route-policy rosen-mldp
   mdt c-multicast-routing bgp
  !
  address-family ipv6
   rpf topology route-policy rosen-mldp
   mdt c-multicast-routing bgp
  !
!
route-policy rosen-mldp
  set core-tree mldp-default
end-policy
```

Profile-17: Rosen mLDP P2MP with BGP-AD and PIM signaling

```
!
multicast-routing
 mdt source Loopback0
 vrf v171
  address-family ipv4
    mdt mtu 1600
mdt default mldp p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
mdt default mldp p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
 !
!
router pim
```

```
 vrf v171
  address-family ipv4
   rpf topology route-policy rosen-mldp
   !
  address-family ipv6
   rpf topology route-policy rosen-mldp
 !
!
route-policy rosen-mldp
  set core-tree mldp-default
end-policy
!
```

Profile-12: Rosen mLDP P2MP with BGP-AD and BGP signaling

```
!
multicast-routing
 mdt source Loopback0
 vrf v121
  address-family ipv4
    mdt mtu 1600
mdt default mldp p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
   !
  address-family ipv6
    mdt mtu 1600
mdt default mldp p2mp
mdt data 255
    interface all enable
   bgp auto-discovery mldp
 !
!
router pim
 vrf v121
  address-family ipv4
   rpf topology route-policy rosen-mldp
  mdt c-multicast-routing bgp
  !
  address-family ipv6
   rpf topology route-policy rosen-mldp
  mdt c-multicast-routing bgp
 !
!
route-policy rosen-mldp
  set core-tree mldp-default
end-policy
!
```

Configuration Examples for multicast support on PW-HE

Enabling multicast

```
configure
multicast-routing
address-family ipv4
interface pw-ether1
enable
vrf v1
address-family ipv4
interface pw-ether2
```

```
enable
 !
!
```

Configuring Static Join (with IGMP)

```
configure
router igmp
interface pw-ether1
static-group 225.0.0.1
static-group 225.0.0.2 10.1.1.1
 !
!
```

Configuring Static Join (with MLD)

```
configure
router mld
interface pw-ether2
static-group ff15::e100
static-group ff15::e100 2000:10::1
  !
!
```

Configuring MVPN Static P2MP TE: Examples

Configuring MVPN P2MP on Ingress PE: Example

```
multicast-routing
 address-family ipv4
  mdt source Loopback0
  interface all enable
 !
 vrf vrf1
  address-family ipv4
   bgp auto-discovery rsvpte
   mdt static p2mp-te tunnel-mte1
   interface all enable
  !
router igmp
 vrf vrf1
  interface tunnel-mte1
   static-group 232.1.1.1 192.1.1.2
  !
```

Configuring MVPN P2MP BGP: Example

```
router bgp 100
 bgp router-id 110.110.110.110
 address-family ipv4 unicast
 address-family vpnv4 unicast
 address-family ipv4 mvpn
 !
 neighbor 130.130.130.130
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  address-family vpnv4 unicast
```

```
 address-family ipv4 mvpn
!
vrf vrf1
 rd 1:1
 bgp router-id 110.110.110.110
 address-family ipv4 unicast
  redistribute connected
 address-family ipv4 mvpn
 !
!
```

Configuring MVPN P2MP on Egress PE: Example

```
multicast-routing
 address-family ipv4
  mdt source Loopback0
  interface all enable
 !
 vrf vrf1
  address-family ipv4
   core-tree-protocol rsvp-te group-list mvpn-acl
   interface all enable
  !

ipv4 access-list mvpn-acl
 10 permit ipv4 host 192.1.1.2 host 232.1.1.1
 20 permit ipv4 any host 232.1.1.2
```

Configuring MVPN Extranet Routing: Example

These examples describe two ways to configure MVPN extranet routing:

For the full set of configuration tasks, see Configuring MVPN Extranet Routing, on page 184.

Configuring the Source MVRF on the Receiver PE Router: Example

The following examples show how to configure MVPN extranet routing by specifying the source MVRF on the receiver PE router.

You must configure both the source PE router and the receiver PE router.

Configure the Source PE Router Using Route Targets

```
interface Loopback5
 ipv4 address 201.5.5.201 255.255.255.255
!
interface Loopback22
 vrf provider-vrf
 ipv4 address 201.22.22.201 255.255.255.255
!
interface GigabitEthernet0/6/0/0
 vrf provider-vrf
 ipv4 address 10.10.10.1 255.255.0.0
!
vrf provider-vrf
 address-family ipv4 unicast
 import route-target
 1100:1
```

```
 !
export route-target
 1100:1
 !
!
router bgp 1
 regular BGP MVPN config
vrf provider-vrf
 rd 1100:1
 address-family ipv4 unicast
 redistribute connected

 !
!
multicast-routing
 vrf provider-vrf address-family ipv4
 mdt data 226.1.4.0/24 threshold 3
 log-traps
 mdt default ipv4 226.0.0.4
 rate-per-route
 interface all enable
 accounting per-prefix
 !
!
address-family ipv4
 nsf
 mdt source Loopback5
 interface all enable
 !
!
router pim vrf provider-vrf address-family ipv4
 rp-address 201.22.22.201
!
```

Configure the Receiver PE Router Using Route Targets

```
interface Loopback5
 ipv4 address 202.5.5.202 255.255.255.255
!
interface GigabitEthernet0/3/0/2
 vrf receiver-vrf
 ipv4 address 20.20.20.1 255.255.0.0
!
vrf provider-vrf
 address-family ipv4 unicast
 import route-target
 1100:1
 !
 export route-target
 1100:1
 !
!
vrf receiver-vrf
 address-family ipv4 unicast
 import route-target
 1100:1
 1101:1
 !
 export route-target
 1101:1
 !
!
```

```
multicast-routing
 vrf provider-vrf address-family ipv4
 log-traps
 mdt default ipv4 226.0.0.4
 rate-per-route
 interface all enable
 accounting per-prefix
!

 vrf receiver_vrf address-family ipv4
 log-traps
 mdt default ipv4 226.0.0.5
 rate-per-route
 interface all enable
 accounting per-prefix
!
address-family ipv4
 nsf
 mdt source Loopback5
 interface all enable
!
router pim vrf provider-vrf address-family ipv4
 rp-address 201.22.22.201
!

router pim vrf receiver_vrf address-family ipv4
 rp-address 201.22.22.201
!
router bgp 1
 regular BGP MVPN config
vrf provider-vrf
 rd 1100:1
 address-family ipv4 unicast
 redistribute connected
 !

 vrf receiver_vrf
 rd 1101:1
 address-family ipv4 unicast
 redistribute connected
!
```

Configuring RPL Policies in Receiver VRFs to Propagate Joins to a Source VRF: Example

In addition to configuring route targets, Routing Policy Language (RPL) policies can be configured in receiver VRFs on receiver PE routers to propagate joins to a specified source VRF. However, this configuration is optional.

The following configuration example shows a policy where the receiver VRF can pick either “provider_vrf_1” or “provider_vrf_2” to propogate PIM joins.

In this example, provider_vrf_1 is used for multicast streams in the range of from 227.0.0.0 to 227.255.255.255, while provider_vrf_2 is being used for streams in the range of from 228.0.0.0 to 228.255.255.255.

```
route-policy extranet_streams_from_provider_vrf
 if destination in (227.0.0.0/32 ge 8 le 32) then
  set rpf-topology vrf provider_vrf_1
 elseif destination in (228.0.0.0/32 ge 8 le 32) then
  set rpf-topology vrf provider_vrf_2
 else
  pass
```

```
 endif
end-policy
!
router pim vrf receiver_vrf address-family ipv4
 rpf topology route-policy extranet_streams_from_provider_vrf
!
```

Configuring the Receiver MVRF on the Source PE Router: Example

The following examples show how to configure MVPN extranet routing by specifying the receiver MVRF on the source PE router.

Note You must configure both the source PE router and the receiver PE router.

Configure the Source PE Router Using Route Targets

```
interface Loopback5
 ipv4 address 202.5.5.202 255.255.255.255
!
interface GigabitEthernet0/3/0/2
 vrf provider-vrf
 ipv4 address 20.20.20.1 255.255.0.0
!
vrf provider-vrf
 address-family ipv4 unicast
 import route-target
 1100:1
 !
 export route-target
 1100:1
 !
!

vrf receiver-vrf
 address-family ipv4 unicast
 import route-target
 1100:1
 1101:1
 !
 export route-target
 1101:1
 !
!

router bgp 1
 regular BGP MVPN config
vrf provider-vrf
 rd 1100:1
 address-family ipv4 unicast
 redistribute connected
 !

 vrf receiver-vrf
 rd 1101:1
 address-family ipv4 unicast
 redistribute connected
 !
```

```
!

multicast-routing
 vrf provider-vrf address-family ipv4
 log-traps
 mdt default ipv4 226.0.0.4
 rate-per-route
 interface all enable
 accounting per-prefix
!

 vrf receiver_vrf address-family ipv4
 log-traps
 mdt default ipv4 226.0.0.5
 rate-per-route
 interface all enable
 accounting per-prefix
!
address-family ipv4
 nsf
 mdt source Loopback5
 interface all enable
!
router pim vrf provider-vrf address-family ipv4
 rp-address 201.22.22.201
!
router pim vrf receiver_vrf address-family ipv4
 rp-address 201.22.22.201
!
```

Configure the Receiver PE Router Using Route Targets

```
interface Loopback5
 ipv4 address 201.5.5.201 255.255.255.255
!
interface Loopback22
 vrf receiver_vrf
 ipv4 address 201.22.22.201 255.255.255.255
!
interface GigabitEthernet0/6/0/0
 vrf receiver_vrf
 ipv4 address 10.10.10.1 255.255.0.0
!

vrf receiver_vrf
 address-family ipv4 unicast
 import route-target
 1100:1
 1101:1
 !
 export route-target
 1101:1
 !
!

router bgp 1
 regular BGP MVPN config
vrf receiver_vrf
 rd 1101:1
 address-family ipv4 unicast
 redistribute connected
 !
```

```
multicast-routing
 vrf receiver_vrf address-family ipv4
 log-traps
 mdt default ipv4 226.0.0.5
 rate-per-route
 interface all enable
 accounting per-prefix
 !
address-family ipv4
 nsf
 mdt source Loopback5
 interface all enable
!

router pim vrf receiver_vrf address-family ipv4
 rp-address 201.22.22.201
!
```

Configuring RPL Policies in Receiver VRFs on Source PE Routers to Propagate Joins to a Source VRF: Example

In addition to configuring route targets , RPL policies can be configured in receiver VRFs on a ource PE router to propagate joins to a specified source VRF. However, this configuration is optional.

The configuration below shows a policy in which the receiver VRF can select either “provider_vrf_1” or “provider_vrf_2” to propagate PIM joins. Provider_vrf_1 will be selected if the rendezvous point (RP) for a multicast stream is 201.22.22.201, while provider_vrf_2 will be selected if the RP for a multicast stream is 202.22.22.201.

As an alternative, you can configure a multicast group-based policy as shown in the Configuring RPL Policies in Receiver VRFs to Propagate Joins to a Source VRF: Example, on page 243.

```
route-policy extranet_streams_from_provider_rp
 if source in (201.22.22.201) then
  set rpf-topology vrf provider_vrf_1
 else if source in (202.22.22.201) then
  set rpf-topology vrf provider_vrf_2
 else
  pass
 endif
end-policy
!
router pim vrf receiver_vrf address-family ipv4
 rpf topology route-policy extranet_streams_from_provider_rp
 rp-address 201.22.22.201 grange_227
 rp-address 202.22.22.201 grange_228
!
```

Configuring Multicast Hub and Spoke Topology: Example

These examples describe two ways to configure Multicast Hub and Spoke:

Figure 22: Example for CE1 PE1PE3 CE3Multicast Hub and Spoke Topology

CE1---------- PE1 ---------- PE3 ---------- CE3

CE1, PE1, and PE3 are all on Cisco IOS XR Software, CE3 has Cisco IOS Software in order to configure autorp on VRF interface. For information about configuring the CE router, using Cisco IOS software, see the appropriate Cisco IOS software documentation.

Hub and Spoke Non-Turnaround Configuration: Example

A1-Hub-1 (bsr RP) A1-Hub-4 (auto-rp RP)

A1-Spoke-3

No turnaround case with bsr and autorp relay

PE1:

```
vrf A1-Hub-1
address-family ipv4 unicast
import route-target

   1000:10

   1001:10

  !

  export route-target

   1000:10

  !

 !

vrf A1-Hub-Tunnel
address-family ipv4 unicast

  import route-target

   1000:10

  !

 !

!

vrf A1-Spoke-Tunnel
address-family ipv4 unicast

  import route-target

   1001:10

  !

 !

!

router pim

 vrf A1-Hub-1
```

```
  address-family ipv4
   rpf topology route-policy A1-Hub-Policy
   bsr relay vrf A1-Hub-Tunnel
   bsr candidate-bsr 201.10.10.201 hash-mask-len 30 priority 4
   bsr candidate-rp 201.10.10.201 group-list A1_PE1_RP_grange priority 4 interval 60
   auto-rp relay vrf A1-Hub-Tunnel
  !
 !
!
router pim
 vrf A1-Hub-Tunnel
  address-family ipv4
  !
 !
!
multicast-routing
 vrf A1-Hub-1
  address-family ipv4
   log-traps
   multipath
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!
multicast-routing
 vrf A1-Hub-Tunnel
  address-family ipv4
   mdt data 226.202.1.0/24 threshold 10
   log-traps
   mdt default ipv4 226.202.0.0
```

```
   rate-per-route
   accounting per-prefix
  !
 !
!
multicast-routing
 vrf A1-Spoke-Tunnel
  address-family ipv4
   mdt mtu 2000
   mdt data 226.202.2.0/24 threshold 5
   log-traps
   mdt default ipv4 226.202.0.1
   rate-per-route
   accounting per-prefix
  !
 !
!
router bgp 1
 vrf A1-Hub-1
  rd 1000:1
  address-family ipv4 unicast
   route-target download
   redistribute connected
   redistribute eigrp 20 match internal external metric 1000
  !
 !
!
router bgp 1
 vrf A1-Hub-Tunnel
  rd 1002:1
  address-family ipv4 unicast
   redistribute connected
  !
```

```
 !
!
router bgp 1
 vrf A1-Spoke-Tunnel
  rd 1002:2
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
route-policy A1-Hub-Policy
  if extcommunity rt matches-any (1000:10) then
    set rpf-topology vrf A1-Hub-Tunnel
  elseif extcommunity rt matches-any (1001:10) then
    set rpf-topology vrf A1-Spoke-Tunnel
  else
    pass
  endif
end-policy
!
route-policy A1-Spoke-Policy
  if extcommunity rt matches-any (1000:10) then
    set rpf-topology vrf A1-Hub-Tunnel
  else
    pass
  endif
end-policy
!
```

PE3:

```
vrf A1-Hub-4
address-family ipv4 unicast
import route-target
   1000:10
```

```
   1001:10
  !
  export route-target
   1000:10
  !
 !
!
vrf A1-Spoke-2
address-family ipv4 unicast
import route-target
   1000:10
  !
  export route-target
   1001:10
  !
 !
!
vrf A1-Hub-Tunnel
address-family ipv4 unicast
import route-target
   1000:10
  !
 !
!
vrf A1-Spoke-Tunnel
address-family ipv4 unicast
import route-target
   1001:10
  !
 !
!
router pim
 vrf A1-Hub-4
  address-family ipv4
   rpf topology route-policy A1-Hub-Policy
```

```
   bsr relay vrf A1-Hub-Tunnel listen
   auto-rp relay vrf A1-Hub-Tunnel
  !
 !
!
router pim
 vrf A1-Spoke-2
  address-family ipv4
   rpf topology route-policy A1-Spoke-Policy
   bsr relay vrf A1-Hub-Tunnel listen
   auto-rp relay vrf A1-Hub-4
  !
 !
!
multicast-routing
 vrf A1-Hub-4
  address-family ipv4
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!
multicast-routing
 vrf A1-Spoke-2
  address-family ipv4
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
```

```
!
multicast-routing
 vrf A1-Hub-Tunnel
  address-family ipv4
   mdt data 226.202.1.0/24 threshold 10
   log-traps
   mdt default ipv4 226.202.0.0
   rate-per-route
   accounting per-prefix
  !
 !
!
multicast-routing
 vrf A1-Spoke-Tunnel
  address-family ipv4
   mdt data 226.202.2.0/24 threshold 5
   log-traps
   mdt default ipv4 226.202.0.1
   rate-per-route
   accounting per-prefix
  !
 !
!
router bgp 1
 vrf A1-Hub-4
  rd 1000:4
  address-family ipv4 unicast
   route-target download
   redistribute connected
   redistribute eigrp 4 match internal external metric 1000
  !
 !
```

```
!
router bgp 1
 vrf A1-Spoke-2
  rd 1001:2
  address-family ipv4 unicast
   route-target download
   redistribute connected
   redistribute eigrp 6 match internal external metric 1000
  !
 !
router bgp 1
 vrf A1-Hub-Tunnel
  rd 1002:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
router bgp 1
 vrf A1-Spoke-Tunnel
  rd 1002:2
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
route-policy A1-Hub-Policy
  if extcommunity rt matches-any (1000:10) then
    set rpf-topology vrf A1-Hub-Tunnel
  elseif extcommunity rt matches-any (1001:10) then
    set rpf-topology vrf A1-Spoke-Tunnel
  else
    pass
  endif
```

```
end-policy
!
route-policy A1-Spoke-Policy
  if extcommunity rt matches-any (1000:10) then
    set rpf-topology vrf A1-Hub-Tunnel
  else
    pass
  endif
end-policy
!
```

CE1:

```
vrf A1-Hub-1
 address-family ipv4 unicast
  import route-target
   1000:10
   1001:10
  !
  export route-target
   1000:10
  !
 !
!
multicast-routing
 vrf A1-Hub-1
  address-family ipv4
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!
```

```
No router pim configuration required
```

CE3: Where autorp is configured (this is an Cisco IOS Software example, because auto-rp on vrf interface is not supported in Cisco IOS XR Software)

```
ip vrf A1-Hub-4
 rd 1000:4
 route-target export 1000:10
 route-target import 1000:10
 route-target import 1001:10
!
ip vrf A1-Spoke-2
 rd 1001:2
 route-target export 1001:10
 route-target import 1000:10
!
ip multicast-routing vrf A1-Hub-4
ip multicast-routing vrf A1-Spoke-2

interface Loopback10
 ip vrf forwarding A1-Hub-4
 ip address 103.10.10.103 255.255.255.255
 ip pim sparse-mode
!
ip pim vrf A1-Hub-4 autorp listener
ip pim vrf A1-Hub-4 send-rp-announce Loopback10 scope 32
ip pim vrf A1-Hub-4 send-rp-discovery Loopback10 scope 32
```

Hub and Spoke with Turnaround: Example

Multicast turnaround mandates a 2-interface connection to the hub site

To configure a CE as a turnaround router, it is connected to its respective PE through two interfaces and each interface is placed in a separate hub site vrf called **hub-x-in vrf** and **hub-x-out vrf**. Hub-x-in vrf carries joins that come from the receiver spoke site through the Hub Tunnel and hub-x-out vrf will carry the same joins towards the source spoke site through the Spoke Tunnel without violating the four basic rules below. The source spoke sends traffic to the spoke tunnel to hub-x-out which is turned around to the hub-tunnel on the hub-x-in interface.

1. Hub sites sends traffic only to MDTHub.
2. Spoke sites sends traffic only to MDTspoke.
3. Hub sites receives traffic from both tunnels.
4. Spoke sites receives traffic only from MDTHub.

A2-Spoke-1 A2-Hub-2

A2-Spoke-2 A2-Hub-3in

A2-Hub-2out

A2-Spoke-3 (spoke has auto-rp)

Figure 23: Example for CE1PE1PE2 CE2Multicast Hub and Spoke Topology with Turnaround

CE1---------- PE1 ---------- PE2 ---------- CE2

Routes exported by hub sites are imported by hub sites and spoke sites. Routes exported by spoke sites are imported by both **hub-x-out** and **hub-x-in** and hub site exports spoke routes back into the core by hub VRF route targets. This causes routes originated from one spoke site to be learned by all other spoke sites but with the nexthop of **hub-x-out**. For example, Spoke2 will see the RPF for Spoke1 reachable with nexthop of **A2-Hub-3in**. This is the fundamental difference in leaking of routes which helps in achieving turnaround of multicast traffic.

PE1:

```
vrf A2-Spoke-1
 address-family ipv4 unicast
  import route-target
   4000:1
   4000:2
   4000:3
   4000:4
  !
  export route-target
   4001:1
  !
 !
!
```

vrf A2-Spoke-2

```
address-family ipv4 unicast
  import route-target
```

```
   4000:1
   4000:2
   4000:3
   4000:4
  !
  export route-target
   4001:2
  !
 !
!
```

PE2:

```
vrf A2-Hub-2
 address-family ipv4 unicast
  import route-target
   4000:1
   4000:2
   4000:3
   4000:4
   4001:1
   4001:2
   4001:3
   4001:4
  !
  export route-target
   4000:2
  !
 !
!


vrf A2-Hub-3out
 address-family ipv4 unicast
```

```
  import route-target
   4000:1
   4000:2
   4000:3
   4000:4
   4001:1   --------à exports the spoke routes into CE2 into vrf default
   4001:2   --------à exports the spoke routes into CE2 into vrf default
   4001:3   --------à exports the spoke routes into CE2 into vrf default
   4001:4   --------à exports the spoke routes into CE2 into vrf default
  !
  export route-target
   4000:4
  !
 !
!
vrf A2-Hub-3in
 address-family ipv4 unicast
  import route-target
   4000:1
   4000:2
   4000:3
   4000:4
  !
  export route-target
   4000:3--------à selected spoke routes (in the prefix-set below) can be re-exported with
 hub route target so other spokes can reach them via A2-Hub-3in
  !
 !
!
prefix-set A2-Spoke-family
  112.31.1.0/24,
  112.32.1.0/24,
  152.31.1.0/24,
```

```
  132.30.1.0/24,
  102.9.9.102/32,
  103.31.31.103/32,
  183.31.1.0/24,
  183.32.1.0/24
end-set
!
route-policy A2-Spoke-family
  if destination in A2-Spoke-family then
    pass
  else
    drop
  endif
end-policy
!


router bgp 1
 vrf A2-Hub-3in
  rd 4000:3
  address-family ipv4 unicast
   route-target download
   redistribute connected
  !
  neighbor 113.113.114.9
   remote-as 12
   address-family ipv4 unicast
```

route-policy A2-Spoke-family in ------à leaking the selected spoke routes with hub route targets so they can be imported by the spoke sites with RPF A2-Hub-3in.

```
    route-policy pass-all out
   !
  !
 !
```

```
!
router bgp 1
 vrf A2-Hub-3out
  rd 4000:4
  address-family ipv4 unicast
   route-target download
   redistribute connected
  !
 !
!
router bgp 1
 vrf A2-Hub-2
  rd 4000:2
  address-family ipv4 unicast
   route-target download
   redistribute connected
   redistribute eigrp 20 match internal external metric 1000
  !
 !
!
multicast-routing
 vrf A2-Hub-2
  address-family ipv4
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!
multicast-routing
 vrf A2-Hub-3in
  address-family ipv4
```

```
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !
 !
!

multicast-routing
 vrf A2-Hub-3out
  address-family ipv4
   log-traps
   rate-per-route
   interface all enable
   accounting per-prefix
  !


 !
!
router pim
 vrf A2-Hub-2
  address-family ipv4
   rpf topology route-policy A2-Hub-Policy
   bsr relay vrf A2-Spoke-3 listen
   auto-rp relay vrf A2-Hub-Tunnel
  !
 !
!
router pim
vrf A2-Hub-3in
address-family ipv4
rpf topology route-policy A2-Hub-Policy
!
!
!
router pim
vrf A2-Hub-3out
address-family ipv4
```

```
rpf topology route-policy A2-Hub-Policy
!
!
!

route-policy A2-Hub-Policy
if extcommunity rt matches-any (4000:1, 4000:2, 4000:3, 4000:4) then
set rpf-topology vrf A2-Hub-Tunnel
elseif extcommunity rt matches-any (4001:1, 4001:2, 4001:3, 4001:4) then
set rpf-topology vrf A2-Spoke-Tunnel
else
pass
endif
end-policy

!
```

Any CE-PE protocol can be used. In this example, A2-Hub-3out exports all the hub and spoke routes to CE2 through EIGRP.

A2-Hub-3in uses route policy A2-Spoke-family to re-import selected spoke routes into PE2 through BGP.

```
router eigrp 20
vrf A2-Hub-3out
address-family ipv4
default-metric 1000 1 255 1 1500
autonomous-system 20
redistribute bgp 1
interface GigabitEthernet0/1/0/1.13
hold-time 60

   !

  !

 !

!
```

CE2:

Here A2-Hub-3in and A2-Hub-3out interfaces are in vrf default and not in a hub site vrf.

```
interface GigabitEthernet0/12/1/0.12
description To PE2 or vrf A2-Hub-3in
ipv4 address 113.113.114.9 255.255.255.252
encapsulation dot1q 3001

!
interface GigabitEthernet0/12/1/0.13
description To PE2 or vrf A2-Hub-3out
ipv4 address 113.113.114.13 255.255.255.252
encapsulation dot1q 3002
!
router bgp 12
nsr
bgp graceful-restart

 address-family ipv4 unicast
redistribute connected
```

```
redistribute eigrp 20
!
neighbor 113.113.114.10   --à this is the A2-Hub-3in neighbor on PE2.
remote-as 1
address-family ipv4 unicast
route-policy pass-all in
route-policy pass-all out
  !
 !
!
```

Configuring LSM based MLDP: Examples

These examples describe multiple profiles to configure MLDP based MVPN:

Rosen MLDP without BGP-Advertisement

```
vrf 1
 vpn id 1:1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
route-policy mldp-1
  set core-tree mldp-default
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
 !
!
router bgp 100 mvpn
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mdt
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
```

```
  address-family ipv4 mdt
  !
 !
 vrf 1
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
!
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 1
  address-family ipv4
   interface all enable
   mdt default mldp ipv4 10.0.0.4
   accounting per-prefix
  !
 !
router pim
 vrf 1
  address-family ipv4
   rpf topology route-policy mldp-1
   rp-address 10.1.1.1
  !
 !
```

Rosen MLDP with BGP Advertisement

```
vrf 101
  vpn id 101:101
 address-family ipv4 unicast
  import route-target
   101:101
  !
  export route-target
   101:101
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback101
 vrf 101
 ipv4 address 10.1.101.1 255.255.255.255
!
```

```
route-policy mldp-101
  set core-tree mldp-default
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100 mvpn
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
 !
 vrf 101
  rd 101:101
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
 !
!
```

```
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
!
router pim
 vrf 101
  address-family ipv4
   rpf topology route-policy mldp-101
   vpn-id 101
   rp-address 10.1.101.1
  !
 !
```

VRF In-band Profile

```
vrf 250
 address-family ipv4 unicast
  import route-target
   250:250
  !
  export route-target
   250:250
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback250
 vrf 250
 ipv4 address 10.1.250.1 255.255.255.255
!
route-policy mldp-250
  set core-tree mldp-inband
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
```

```
 address-family vpnv6 unicast
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
 !
 vrf 250
  rd 250:250
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
 !
!
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 250
  address-family ipv4
   mdt mldp in-band-signaling
   interface all enable
  !
 !
router pim
 vrf 250
  address-family ipv4
   rpf topology route-policy mldp-250
   rp-address 10.1.250.1
  !
 !
```

Partitioned-MDT MP2MP without BGP-AD

```
vrf 251
 address-family ipv4 unicast
  import route-target
   251:251
  !
  export route-target
   251:251
```

```
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback251
 vrf 251
 ipv4 address 10.11.1.1 255.255.255.255
!
route-policy mldp-251
  set core-tree mldp-partitioned-mp2mp
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
!
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
 !
 vrf 251
  rd 251:251
  address-family ipv4 unicast
   redistribute connected
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
```

```
  logging internal
 !
 <all core-facing interfaces>
 !
!
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 251
  address-family ipv4
   mdt partitioned mldp ipv4 mp2mp
   interface all enable
  !
 !
router pim
 vrf 251
  address-family ipv4
   rpf topology route-policy mldp-251
   rp-address 10.11.1.1
  !
 !
```

Partitioned-MDT MP2MP with BGP-AD

```
vrf 301
 address-family ipv4 unicast
  import route-target
   301:301
  !
  export route-target
   301:301
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback301
 vrf 301
 ipv4 address 10.11.51.1 255.255.255.255
!
route-policy mldp-301
  set core-tree mldp-partitioned-mp2mp
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
```

```
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
 !
 vrf 301
  rd 301:301
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
 !
!

multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 301
  address-family ipv4
   bgp auto-discovery mldp
   mdt partitioned mldp ipv4 mp2mp
   interface all enable
  !
 !
router pim
 vrf 301
  address-family ipv4
```

```
   rpf topology route-policy mldp-301
   rp-address 10.11.51.1
  !
 !
```

Multidirectional Selective Provider Multicast Service Instance mLDP-P2MP with BGP-Advertisement

```
vrf 401
 address-family ipv4 unicast
  import route-target
   401:401
  !
  export route-target
   401:401
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback401
 vrf 401
 ipv4 address 10.11.151.1 255.255.255.255
!
route-policy mldp-401
  set core-tree mldp-partitioned-p2mp
end-policy
!
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
```

```
  !
  address-family ipv4 mvpn
  !
 !
 vrf 401
  rd 401:401
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
 !
!
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 401
  address-family ipv4
   bgp auto-discovery mldp
   mdt partitioned mldp ipv4 p2mp
   interface all enable
  !
 !
router pim
 vrf 401
  address-family ipv4
   rpf topology route-policy mldp-401
   rp-address 10.11.151.1
  !
```

Rosen-GRE with BGP-Advertisement

```
vrf 501
  address-family ipv4 unicast
  import route-target
   501:501
  !
  export route-target
   501:501
  !
 !
!
interface Loopback0
 ipv4 address 10.0.0.4 255.255.255.255
!
interface Loopback501
 vrf 501
```

```
 ipv4 address 10.111.1.1 255.255.255.255
!

<no route policy?>

 vrf 501
  rd 501:501
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
router ospf 1
 address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface Loopback1
  !
  interface GigabitEthernet0/0/2/0
  !
  interface GigabitEthernet0/3/2/1
  !
  interface GigabitEthernet0/3/2/2
  !
 !
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 address-family ipv4 unicast
  redistribute connected
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 neighbor 5.5.5.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family vpnv6 unicast
  !
  address-family ipv4 mvpn
  !
 !
 vrf 501
  rd 501:501
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
mpls traffic-eng
 interface GigabitEthernet0/0/2/0
 !
!
```

```
mpls ldp
 router-id 10.0.0.4
 graceful-restart
 mldp
  logging internal
 !
 <all core-facing interfaces>
 !
!
multicast-routing
 address-family ipv4
  nsf
  mdt source Loopback0
  interface all enable
  accounting per-prefix
 !
 vrf 501
  address-family ipv4
   bgp auto-discovery pim
   mdt default ipv4 232.1.1.1
   interface all enable
  !
 !
router pim
 vrf 501
  address-family ipv4
   rp-address 10.111.1.1
  !
 !
```

Additional References

Related Documents

Related Topic	Document Title
Multicast command reference document	*Multicast Command Reference for Cisco ASR 9000 Series Routers*
Getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Modular quality of service command reference document	*Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*
Routing command reference and configuration documents	*Routing Command Reference for Cisco ASR 9000 Series Routers* *Routing Configuration Guide for Cisco ASR 9000 Series Routers*
Information about user groups and task IDs	*System Security Configuration Guide for Cisco ASR 9000 Series Routers*