



EVPN Features

This chapter describes how to configure Layer 2 Ethernet VPN (EVPN) features on the router.

- [EVPN Overview, on page 1](#)
- [EVPN Concepts, on page 2](#)
- [EVPN Operation, on page 3](#)
- [EVPN Route Types, on page 4](#)
- [Configure EVPN L2 Bridging Service, on page 5](#)
- [EVPN Software MAC Learning , on page 7](#)
- [EVPN Out of Service, on page 14](#)
- [CFM Support for EVPN, on page 18](#)
- [EVPN Multiple Services per Ethernet Segment, on page 18](#)
- [EVPN MPLS Seamless Integration with VPLS , on page 24](#)
- [Configure EVPN on the Existing VPLS Network, on page 25](#)
- [EVI Configuration Under L2VPN Bridge-Domain, on page 27](#)
- [Verify EVPN Configuration, on page 28](#)
- [Network Convergence using Core Isolation Protection, on page 32](#)
- [Conditional Advertisement of Default-Originate, on page 38](#)
- [EVPN Core Isolation Protection, on page 42](#)
- [EVPN Routing Policy, on page 44](#)
- [Support for DHCPv4 and DHCPv6 Client over BVI, on page 59](#)

EVPN Overview

Ethernet VPN (EVPN) is a next generation solution that provides Ethernet multipoint services over MPLS networks. EVPN operates in contrast to the existing Virtual Private LAN Service (VPLS) by enabling control-plane based MAC learning in the core. In EVPN, PEs participating in the EVPN instances learn customer MAC routes in control-plane using MP-BGP protocol. Control-plane MAC learning brings a number of benefits that allow EVPN to address the VPLS shortcomings, including support for multi-homing with per-flow load balancing.

EVPN provides the solution for network operators for the following emerging needs in their network:

- Data center interconnect operation (DCI)
- Cloud and services virtualization

- Remove protocols and network simplification
- Integration of L2 and L3 services over the same VPN
- Flexible service and workload placement
- Multi-tenancy with L2 and L3 VPN
- Optimal forwarding and workload mobility
- Fast convergence
- Efficient bandwidth utilization

EVPN Benefits

The EVPN provides the following benefits:

- **Integrated Services:** Integrated L2 and L3 VPN services, L3VPN-like principles and operational experience for scalability and control, all-active multi-homing and PE load-balancing using ECMP, and enables load balancing of traffic to and from CEs that are multihomed to multiple PEs.
- **Network Efficiency:** Eliminates flood and learn mechanism, fast-reroute, resiliency, and faster reconvergence when the link to dual-homed server fails, optimized Broadcast, Unknown-unicast, Multicast (BUM) traffic delivery.
- **Service Flexibility:** MPLS data plane encapsulation, support existing and new services types (E-LAN, E-Line), peer PE auto-discovery, and redundancy group auto-sensing.

EVPN Modes

The following EVPN modes are supported:

- **Single-homing** - This enables you to connect a customer edge (CE) device to one provider edge (PE) device.
- **Multihoming** - This enables you to connect a customer edge (CE) device to more than one provider edge (PE) device. Multihoming ensures redundant connectivity. The redundant PE device ensures that there is no traffic disruption when there is a network failure. Following are the types of multihoming:
 - **All-Active** - In all-active mode all the PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

EVPN Concepts

To implement EVPN features, you need to understand the following concepts:

- **Ethernet Segment (ES):** An Ethernet segment is a set of Ethernet links that connects a multihomed device. If a multi-homed device or network is connected to two or more PEs through a set of Ethernet links, then that set of links is referred to as an Ethernet segment. The Ethernet segment route is also referred to as Route Type 4. This route is used for designated forwarder (DF) election for BUM traffic.
- **Ethernet Segment Identifier (ESI):** Ethernet segments are assigned a unique non-zero identifier, which is called an Ethernet Segment Identifier (ESI). ESI represents each Ethernet segment uniquely across the network.

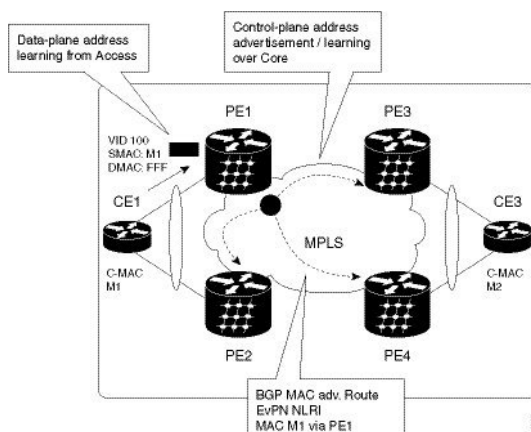
- **EVI:** The EVPN instance (EVI) is represented by the virtual network identifier (VNI). An EVI represents a VPN on a PE router. It serves the same role of an IP VPN Routing and Forwarding (VRF), and EVIs are assigned import/export Route Targets (RTs). Depending on the service multiplexing behaviors at the User to Network Interface (UNI), all traffic on a port (all-to-one bundling), or traffic on a VLAN (one-to-one mapping), or traffic on a list/range of VLANs (selective bundling) can be mapped to a Bridge Domain (BD). This BD is then associated to an EVI for forwarding towards the MPLS core.
- **EAD/ES:** Ethernet Auto Discovery Route per ES is also referred to as Route Type 1. This route is used to converge the traffic faster during access failure scenarios. This route has Ethernet Tag of 0xFFFFFFFF.
- **EAD/EVI:** Ethernet Auto Discovery Route per EVI is also referred to as Route Type 1. This route is used for aliasing and load balancing when the traffic only hashes to one of the switches. This route cannot have Ethernet tag value of 0xFFFFFFFF to differentiate it from the EAD/ES route.
- **Aliasing:** It is used for load balancing the traffic to all the connected switches for a given Ethernet segment using the Route Type 1 EAD/EVI route. This is done irrespective of the switch where the hosts are actually learned.
- **Mass Withdrawal:** It is used for fast convergence during the access failure scenarios using the Route Type 1 EAD/ES route.
- **DF Election:** It is used to prevent forwarding of the loops. Only a single router is allowed to decapsulate and forward the traffic for a given Ethernet Segment.

EVPN Operation

At startup, PEs exchange EVPN routes in order to advertise the following:

- **VPN membership:** The PE discovers all remote PE members of a given EVI. In the case of a multicast ingress replication model, this information is used to build the PEs flood list associated with an EVI. BUM labels and unicast labels are exchanged when MAC addresses are learned.
- **Ethernet segment reachability:** In multihoming scenarios, the PE auto-discovers remote PE and their corresponding redundancy mode (all-active or single-active). In case of segment failures, PEs withdraw the routes used at this stage in order to trigger fast convergence by signaling a MAC mass withdrawal on remote PEs.
- **Redundancy Group membership:** PEs connected to the same Ethernet segment (multihoming) automatically discover each other and elect a Designated Forwarder (DF) that is responsible for forwarding Broadcast, Unknown unicast and Multicast (BUM) traffic for a given EVI.

Figure 1: EVPN Operation



EVPN can operate in single-homing or dual-homing mode. Consider single-homing scenario, when EVPN is enabled on PE, Route Type 3 is advertised where each PE discovers all other member PEs for a given EVPN instance. When an unknown unicast (or BUM) MAC is received on the PE, it is advertised as EVPN Route Type 2 to other PEs. MAC routes are advertised to the other PEs using EVPN Route Type 2. In multihoming scenarios, Route Types 1, 3, and 4 are advertised to discover other PEs and their redundancy modes (single-active or all-active). Use of Route Type 1 is to auto-discover other PE which hosts the same CE. The other use of this route type is to fast route unicast traffic away from a broken link between CE and PE. Route Type 4 is used for electing designated forwarder. For instance, consider the topology when customer traffic arrives at the PE, EVPN MAC advertisement routes distribute reachability information over the core for each customer MAC address learned on local Ethernet segments. Each EVPN MAC route announces the customer MAC address and the Ethernet segment associated with the port where the MAC was learned from and its associated MPLS label. This EVPN MPLS label is used later by remote PEs when sending traffic destined to the advertised MAC address.

EVPN Route Types

The EVPN network layer reachability information (NLRI) provides different route types.

Table 1: EVPN Route Types

Route Type	Name	Usage
1	Ethernet Auto-Discovery (AD) Route	Few routes are sent per ES, carries the list of EVIs that belong to ES
2	MAC/IP Advertisement Route	Advertise MAC, address reachability, advertise IP/MAC binding
3	Inclusive Multicast Ethernet Tag Route	Multicast Tunnel End point discovery
4	Ethernet Segment Route	Redundancy group discovery, DF election

Route Type	Name	Usage
5	IP Prefix Route	Advertise IP prefixes.

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet Auto-Discovery (AD) routes are advertised on per EVI and per ESI basis. These routes are sent per ES. They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed. This route type is used for mass withdrawal of MAC addresses and aliasing for load balancing.

Route Type 2: MAC/IP Advertisement Route

These routes are per-VLAN routes, so only PEs that are part of a VNI require these routes. The host's IP and MAC addresses are advertised to the peers within NRLI. The control plane learning of MAC addresses reduces unknown unicast flooding.

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

Route Type 5: IP Prefix Route

The IP prefixes are advertised independently of the MAC-advertised routes. With EVPN IRB, host route /32 is advertised using RT-2 and subnet /24 is advertised using RT-5.



Note With EVPN IRB, host route /32 are advertised using RT-2 and subnet /24 are advertised using RT-5.

Configure EVPN L2 Bridging Service

Perform the following steps to configure EVPN L2 bridging service.



Note Always ensure to change the label mode from per-prefix to per-VRF label mode. Since L2FIB and VPNv4 route (labels) shares the same resource, BVI ping fails when you exhaust the resources.



Note Flooding disable isn't supported on EVPN bridge domains.

```
/* Configure address family session in BGP */
```

```

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

/* Configure EVI and define the corresponding BGP route targets */

Router# configure
Router(config)# evpn
Router(config-evpn)# evi 6005
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 200:50
Router(config-evpn-evi-bgp)# route-target import 100:6005
Router(config-evpn-evi-bgp)# route-target export 100:6005
Router(config-evpn-evi-bgp)# exit
Router(config-evpn-evi)# advertise-mac

/* Configure a bridge domain */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 1
Router(config-l2vpn-bg)# bridge-domain 1-1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpn-bg-bd-ac-evi)# commit
Router(config-l2vpnbg-bd-ac-evi)# exit

```

Running Configuration

```

router bgp 200 bgp
  router-id 209.165.200.227
  address-family l2vpn evpn
  neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  updatesource Loopback0
  addressfamily l2vpn evpn
!

configure
evpn
  evi 6005
  bgp
    rd 200:50
    route-target import 100:6005
    route-target export 100:6005
!
  advertise-mac

configure
l2vpn
  bridge group 1
  bridge-domain 1-1
  interface GigabitEthernet 0/0/0/1

```

```

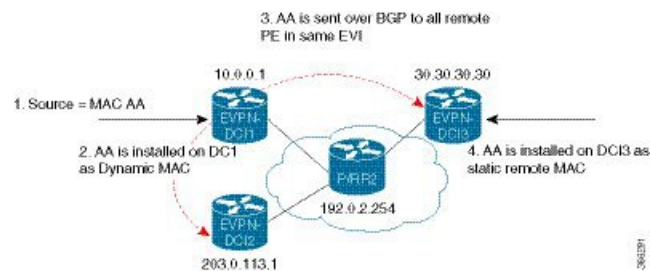
    evi 6005
    !

```

EVPN Software MAC Learning

The MAC addresses learned on one device needs to be learned or distributed on the other devices in a VLAN. EVPN Software MAC Learning feature enables the distribution of the MAC addresses learned on one device to the other devices connected to a network. The MAC addresses are learnt from the remote devices using BGP.

Figure 2: EVPN Software MAC Learning



The above figure illustrates the process of software MAC learning. The following are the steps involved in the process:

1. Traffic comes in on one port in the bridge domain.
2. The source MAC address (AA) is learnt on the PE and is stored as a dynamic MAC entry.
3. The MAC address (AA) is converted into a type-2 BGP route and is sent over BGP to all the remote PEs in the same EVI.
4. The MAC address (AA) is updated on the PE as a remote MAC address.

Configure EVPN Software MAC Learning

The following section describes how you can configure EVPN Software MAC Learning:



Note The router does not support flow-aware transport (FAT) pseudowire.

```

/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_SH
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface BundleEther 20.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 10000 ← Enabling
storm-control is optional
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit

```

```

RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-evi)# commit

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn

RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# description MPLSFACINGPEER
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn

```

Supported Modes for EVPN Software MAC Learning

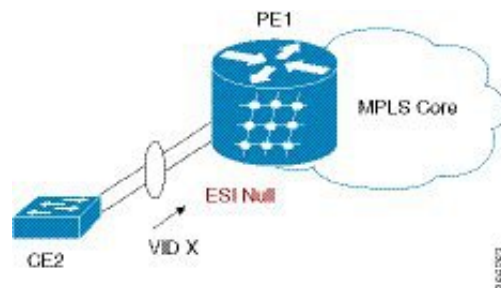
The following are the modes in which EVPN Software MAC Learning is supported:

- Single Home Device (SHD) or Single Home Network (SHN)
- Dual Home Device (DHD)—All Active Load Balancing

Single Home Device or Single Home Network Mode

The following section describes how you can configure EVPN Software MAC Learning feature in single home device or single home network (SHD/SHN) mode:

Figure 3: Single Home Device or Single Home Network Mode



In the above figure, the PE (PE1) is attached to Ethernet Segment using bundle or physical interfaces. Null Ethernet Segment Identifier (ESI) is used for SHD/SHN.

Configure EVPN in Single Home Device or Single Home Network Mode

This section describes how you can configure EVPN Software MAC Learning feature in single home device or single home network mode.

```

/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001

```



```

/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 09.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

```

Running Configuration

```

l2vpn
bridge group EVPN_ALL_ACTIVE
  bridge-domain EVPN_2001
  interface BundleEther1.2001
  evi 2001
!
evpn
  evi 2001
  advertise-mac
!
router bgp 200 bgp
  router-id 40.40.40.40
  address-family l2vpn evpn
  neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  updatesource Loopback0
  addressfamily l2vpn evpn

```

Verification

Verify EVPN in single home devices.

```

RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Te0/4/0/10 detail

```

Ethernet Segment Id	Interface	Nexthops
N/A	Te0/4/0/10	20.20.20.20

```

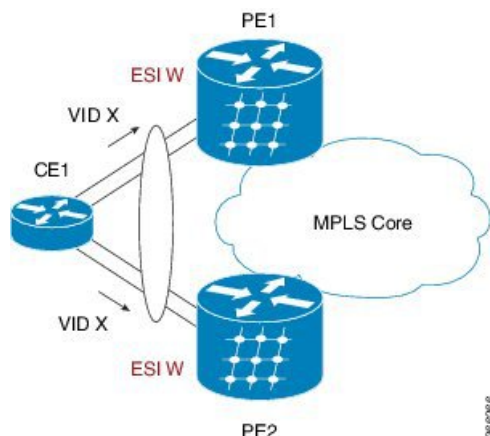
.....
Topology :
Operational : SH
Configured : Single-active (AaPS) (default)

```

Dual Home Device—All-Active Load Balancing Mode

The following section describes how you can configure EVPN Software MAC Learning feature in dual home device (DHD) in all-active load balancing mode:

Figure 4: Dual Home Device —All-Active Load Balancing Mode



All-active load-balancing is known as Active/Active per Flow (AApF). In the above figure, identical Ethernet Segment Identifier is used on both EVPN PEs. PEs are attached to Ethernet Segment using bundle interfaces. In the CE, single bundles are configured towards two EVPN PEs. In this mode, the MAC address that is learnt is stored on both PE1 and PE2. Both PE1 and PE2 can forward the traffic within the same EVI.

Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode

This section describes how you can configure EVPN Software MAC Learning feature in dual home device—all-active mode:

```

/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001

/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# exit
RP/0/RSP0/CPU0:router(config-evpn)# interface bundle-ether1
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.01

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn

```

```

/* Configure Link Aggregation Control Protocol (LACP) bundle. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1.300
RP/0/RSP0/CPU0:router(config-if)# lacp switchover suppress-flaps 300
RP/0/RSP0/CPU0:router(config-if)# exit

/* Configure VLAN Header Rewrite.*/

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface bundle-Ether1.2001 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-if)# rewrite ingress tag pop 1 symmetric

```

Running Configuration

```

l2vpn
bridge group EVPN_ALL_ACTIVE
  bridge-domain EVPN_2001
  interface Bundle-Ether1.2001
  !
  evi 2001
  !
!
evpn
  evi 2001
  !
  advertise-mac
  !
  interface bundle-ether1
    ethernet-segment
      identifier type 0 01.11.00.00.00.00.00.01
  !
!
router bgp 200
  bgp router-id 209.165.200.227
  address-family l2vpn evpn
  !
  neighbor 10.10.10.10
    remote-as 200
    description MPLS-FACING-PEER
    update-source Loopback0
    address-family l2vpn evpn
  !
  interface Bundle-Ether1
    lacp switchover suppress-flaps 300
    load-interval 30
  !
  interface bundle-Ether1.2001 l2transport
    encapsulation dot1q 2001
    rewrite ingress tag pop 1 symmetric
  !

```

Verification

Verify EVPN in dual home devices in All-Active mode.



Note With the EVPN IRB, the supported label mode is per-VRF.

```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface bundle-Ether 1 carvin$

Ethernet Segment Id      Interface  Nexthops
-----
0100.211b.fce5.df00.0b00  BE11      10.10.10.10
209.165.201.1
Topology :
Operational : MHN
Configured : All-active (AApF) (default)
Primary Services : Auto-selection
Secondary Services: Auto-selection
Service Carving Results:
Forwarders : 4003
Elected : 2002
EVI E : 2000, 2002, 36002, 36004, 36006, 36008
.....
Not Elected : 2001
EVI NE : 2001, 36001, 36003, 36005, 36007, 36009

MAC Flushing mode : Invalid

Peering timer : 3 sec [not running]
Recovery timer : 30 sec [not running]
Local SHG label : 34251
Remote SHG labels : 1
38216 : nexthop 209.165.201.1
```

Verify EVPN Software MAC Learning

Verify the packet drop statistics.



Note Disable CW configuration if any in EVPN peer nodes, as CW is not supported in EVPN Bridging.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name EVPN_2001 details

Bridge group: EVPN_ALL_ACTIVE, bridge-domain: EVPN_2001, id: 1110,
state: up, ShgId: 0, MSTi: 0
List of EVPNs:
EVPN, state: up
evi: 2001
XC ID 0x80000458
Statistics:
packets: received 28907734874 (unicast 9697466652), sent
76882059953
bytes: received 5550285095808 (unicast 1861913597184), sent
14799781851396
MAC move: 0
List of ACs:
AC: TenGigE0/0/0/1, state is up
Type VLAN; Num Ranges: 1
...
Statistics:
packets: received 0 (multicast 0, broadcast 0, unknown
unicast 0, unicast 0), sent 45573594908
bytes: received 0 (multicast 0, broadcast 0, unknown unicast
0, unicast 0), sent 8750130222336
MAC move: 0
.....
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 neighbor
```

```
Neighbor IP      vpn-id
-----
209.165.200.225  2001
209.165.201.30   2001
```

Verify the BGP L2VPN EVPN summary.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn summary
```

```
...
Neighbor      Spk  AS      MsgRcvd  MsgSent  TblVer   InQ   OutQ   Up/Down  St/PfxRcd
209.165.200.225  0    200     216739  229871   200781341  0     0     3d00h   348032
209.165.201.30   0    200     6462962 4208831  200781341 10     0     2d22h   35750
```

Verify the MAC updates to the L2FIB table in a line card.

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location
```

```
Topo ID Producer Next Hop(s)      Mac Address      IP Address
-----
1112     0/6/CPU0 Te0/0/0/1 00a3.0001.0001
```

Verify the MAC updates to the L2FIB table in a route switch processor (RSP).

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location
```

```
Topo ID  Producer Next Hop(s)      Mac Address      IP Address
-----
1112     0/6/CPU0 0/0/0/1 00a3.0001.0001
```

Verify the summary information for the MAC address.

```
RP/0/RSP0/CPU0:router# show l2vpn forwarding bridge-domain EVPN_ALL_ACTIVE:EVPN_2001
mac-address location
```

```
.....
Mac Address      Type          Learned from/Filtered on  LC learned  Resync Age/Last Change
Mapped to
0000.2001.5555   dynamic      Te0/0/0/2                N/A         11 Jan 14:37:22
N/A <-- local dynamic
00bb.2001.0001   dynamic      Te0/0/0/2                N/A         11 Jan 14:37:22
N/A
0000.2001.1111   EVPN         BD id: 1110              N/A         N/A
N/A <-- remote static
00a9.2002.0001   EVPN         BD id: 1110              N/A         N/A
N/A
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac
```

```
EVI      MAC address      IP address      Nexthop      Label
-----

```

```

2001 00a9.2002.0001 :: 10.10.10.10 34226 <-- Remote MAC
2001 00a9.2002.0001 :: 209.165.201.30 34202

2001 0000.2001.5555 20.1.5.55 TenGigE0/0/0/2 34203 <-- local MAC

```

```

RP/0/RSP0/CPU0:router# RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac 00a9.2002.0001
detail

```

EVI	MAC address	IP address	Nexthop	Label
2001	00a9.2002.0001	::	10.10.10.10	34226
2001	00a9.2002.0001	::	209.165.201.30	34202

Ethernet Tag : 0

Multi-paths Resolved : True <--- aliasing to two remote PE with All-Active load balancing

Static : No

Local Ethernet Segment : N/A

Remote Ethernet Segment : 0100.211b.fce5.df00.0b00

Local Sequence Number : N/A

Remote Sequence Number : 0

Local Encapsulation : N/A

Remote Encapsulation : MPLS

Verify the BGP routes associated with EVPN with bridge-domain filter.

```

RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2

*> [2][0][48][00bb.2001.0001][0]/104
      0.0.0.0          0 i <----- locally learnt MAC
*>i[2][0][48][00a9.2002.00be][0]/104
      10.10.10.10 100 0 i <----- remotely learnt MAC
* i 209.165.201.30 100 0 i

```

EVPN Out of Service

The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

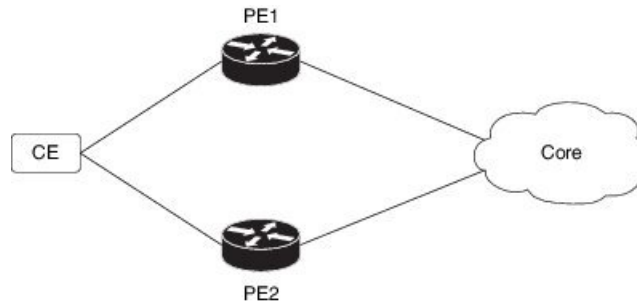
Use the **cost-out** command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.



Note EVPN cost-out is supported only on manually configured ESIs.

In the following topology, the CE is connected to PE1 and PE2. When you configure the **cost-out** command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

Figure 5: EVPN Out of Service



To bring up the node into service, use **no cost-out** command. This brings up all the bundle interfaces belonging to EVPN Ethernet segment on the PE and the corresponding bundle members on the CE.

When the node is in cost-out state, adding a new bundle Ethernet segment brings that bundle down. Similarly, removing the bundle Ethernet segment brings that bundle up.

Use **startup-cost-in** command to bring up the node into service after the specified time on reload. The node will cost-out when EVPN is initialized and remain cost-out until the set time. If you execute **evpn no startup-cost-in** command while timer is running, the timer stops and node is cost-in.

The 'cost-out' configuration always takes precedence over the 'startup-cost-in' timer. So, if you reload with both the configurations, cost-out state is controlled by the 'cost-out' configuration and the timer is not relevant. Similarly, if you reload with the startup timer, and configure 'cost-out' while timer is running, the timer is stopped and OOS state is controlled only by the 'cost-out' configuration.

If you do a proc restart while the startup-cost-in timer is running, the node remains in cost-out state and the timer restarts.

Configure EVPN Out of Service

This section describes how you can configure EVPN Out of Service.

```
/* Configuring node cost-out on a PE */
```

```
Router# configure
Router(config)# evpn
Router(config-evpn)# cost-out
Router(config-evpn) commit
```

```
/* Bringing up the node into service */
```

```
Router# configure
Router(config)# evpn
Router(config-evpn)# no cost-out
Router(config-evpn) commit
```

```
/* Configuring the timer to bring up the node into service after the specified time on reload */
```

```

Router# configure
Router(config)# evpn
Router(config-evpn)# startup-cost-in 6000
Router(config-evpn) commit

```

Running Configuration

```

configure
evpn
  cost-out
!

configure
evpn
  startup-cost-in 6000
!

```

Verification

Verify the EVPN Out of Service configuration.

```

/* Verify the node cost-out configuration */

Router# show evpn summary
Fri Apr 7 07:45:22.311 IST
Global Information
-----
Number of EVIs : 2
Number of Local EAD Entries : 0
Number of Remote EAD Entries : 0
Number of Local MAC Routes : 0
Number of Local MAC Routes : 5
      MAC : 5
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes : 7
      MAC : 7
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels : 5
Number of ES Entries : 9
Number of Neighbor Entries : 1
EVPN Router ID : 192.168.0.1
BGP Router ID : ::
BGP ASN : 100
PBB BSA MAC address : 0207.1fee.be00
Global peering timer : 3 seconds
Global recovery timer : 30 seconds
EVPN cost-out : TRUE
      startup-cost-in timer : Not configured

/* Verify the no cost-out configuration */

Router# show evpn summary
Fri Apr 7 07:45:22.311 IST
Global Information

```



```

-----
Number of EVIs : 2
Number of Local EAD Entries : 0
Number of Remote EAD Entries : 0
Number of Local MAC Routes : 0
Number of Local MAC Routes : 5
      MAC : 5
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes : 7
      MAC : 7
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels : 5
Number of ES Entries : 9
Number of Neighbor Entries : 1
EVPN Router ID : 192.168.0.1
BGP Router ID : ::
BGP ASN : 100
PBB BSA MAC address : 0207.1fee.be00
Global peering timer : 3 seconds
Global recovery timer : 30 seconds
EVPN cost-out : FALSE
      startup-cost-in timer : Not configured

```

```
/* Verify the startup-cost-in timer configuration */
```

```

Router# show evpn summary
Fri Apr 7 07:45:22.311 IST
Global Information
-----
Number of EVIs : 2
Number of Local EAD Entries : 0
Number of Remote EAD Entries : 0
Number of Local MAC Routes : 0
Number of Local MAC Routes : 5
      MAC : 5
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes : 7
      MAC : 7
      MAC-IPv4 : 0
      MAC-IPv6 : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels : 5
Number of ES Entries : 9
Number of Neighbor Entries : 1
EVPN Router ID : 192.168.0.1
BGP Router ID : ::
BGP ASN : 100
PBB BSA MAC address : 0207.1fee.be00
Global peering timer : 3 seconds
Global recovery timer : 30 seconds
EVPN node cost-out : TRUE
      startup-cost-in timer : 6000

```

CFM Support for EVPN

Ethernet Connectivity Fault Management (CFM) is a service-level OAM protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services per VLAN. This includes proactive connectivity monitoring, fault verification, and fault isolation. CFM can be deployed in an EVPN network. You can monitor the connections between the nodes using CFM in an EVPN network.

Restrictions

CFM for EVPN is supported with the following restrictions:

- In an active-active multi-homing scenario, when monitoring the connectivity between a multi-homed CE device and the PE devices to which it is connected, CFM can only be used across each individual link between a CE and a PE. Attempts to use CFM on the bundle between CE and PE devices cause sequence number errors and statistical inaccuracies.
- There is a possibility of artefacts in loopback and linktrace results. Either a loopback or linktrace may report multiple results for the same instance, or consecutive instances of a loopback and linktrace between the same two endpoints may produce different results.

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM* chapter in the *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 Series Routers*.

EVPN Multiple Services per Ethernet Segment

EVPN Multiple Services per Ethernet Segment feature allows you to configure multiple services over single Ethernet Segment (ES). Instead of configuring multiple services over multiple ES, you can configure multiple services over a single ES.

You can configure the following services on a single Ethernet Bundle; you can configure one service on each sub-interface.

- Flexible cross-connect (FXC) service. It supports VLAN Unaware, VLAN Aware, and Local Switching modes.

For more information, see *Configure Point-to-Point Layer 2 Services* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS 540 Series Routers*.

- EVPN-VPWS Xconnect service

For more information, see *EVPN Virtual Private Wire Service (VPWS)* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS 540 Series Routers*.

- EVPN Integrated Routing and Bridging (IRB)

For more information, see *Configure EVPN IRB* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS 540 Series Routers*.

- Native EVPN

For more information see, *EVPN Features* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco NCS 540 Series Routers*.

All these services are supported only on all-active multihoming scenario.

Configure EVPN Multiple Services per Ethernet Segment

Consider a customer edge (CE) device connected to two provider edge (PE) devices through Ethernet Bundle interface 22001. Configure multiple services on Bundle Ethernet sub-interfaces.

Configuration Example

Consider Bundle-Ether22001 ES, and configure multiple services on sub-interface.

```

/* Configure attachment circuits */
Router# configure
Router(config)# interface Bundle-Ether22001.12 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 12
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.13 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 13
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.14 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 14
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.1 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 1
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 2
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.3 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 3
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit
Router(config)# interface Bundle-Ether22001.4 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 4
Router(config-l2vpn-subif)# exit
Router(config-l2vpn)# exit

/*Configure VLAN Unaware FXC Service */
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-unaware fxc_mh1
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.1
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.2
Router(config-l2vpn-fxs-vu)# interface Bundle-Ether22001.3
Router(config-l2vpn-fxs-vu)# neighbor evpn evi 21006 target 22016
Router(config-l2vpn-fxs-vu)# commit

/* Configure VLAN Aware FXC Service */
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 24001
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.12
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.13
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.14
Router(config-l2vpn-fxs-va)# commit

```

```

/* Configure Local Switching - Local switching is supported only on VLAN-aware FXC */
PE1
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 31400
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.1400
Router(config-l2vpn-fxs-va)# interface Bundle-Ether23001.1400
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit
PE2
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# flexible-xconnect-service vlan-aware evi 31401
Router(config-l2vpn-fxs-va)# interface Bundle-Ether22001.1401
Router(config-l2vpn-fxs-va)# interface Bundle-Ether23001.1401
Router(config-l2vpn-fxs-va)# commit
Router(config-l2vpn-fxs)# exit

/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */

Router# configure
Router(config)# interface Bundle-Ether22001.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
Router(config)# interface Bundle-Ether22001.21 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 21
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg22001
Router(config-l2vpn-xc)# p2p evpn-vpws-mclag-22001
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether22001.11
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 22101 target 220101 source 220301
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit

Router # configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group native_evpn1
Router (config-l2vpn-bg)# bridge-domain bd21
Router (config-l2vpn-bg-bd)# interface Bundle-Ether22001.21
Router (config-l2vpn-bg-bd-ac)# routed interface BVI21
Router (config-l2vpn-bg-bd-bvi)# evi 22021
Router (config-l2vpn-bg-bd-bvi)# commit
Router (config-l2vpn-bg-bd-bvi)# exit

/* Configure Native EVPN */
Router # configure
Router (config)# evpn
Router (config-evpn)# interface Bundle-Ether22001
Router (config-evpn-ac)# ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff. ee
Router (config-evpn-ac-es)# bgp route-target 2200.0001.0001
Router (config-evpn-ac-es)# exit
Router (config-evpn)# evi 24001
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:24001

```

```

Router (config-evpn-evi-bgp) # route-target export 64:24001
Router (config-evpn-evi-bgp) # exit
Router (config-evpn-evi) # exit
Router (config-evpn) # evi 21006
Router (config-evpn-evi) # bgp
Router (config-evpn-evi-bgp) # route-target route-target 64:10000
Router (config-evpn-evi-bgp) # exit
Router (config-evpn-evi) # exit
Router (config-evpn) # evi 22101
Router (config-evpn-evi) # bgp
Router (config-evpn-evi-bgp) # route-target import 64:22101
Router (config-evpn-evi-bgp) # route-target export 64:22101
Router (config-evpn-evi-bgp) # exit
Router (config-evpn-evi) # exit
Router (config-evpn) # evi 22021
Router (config-evpn-evi) # bgp
Router (config-evpn-evi-bgp) # route-target import 64: 22021
Router (config-evpn-evi-bgp) # route-target export 64: 22021
Router (config-evpn-evi-bgp) # exit
Router (config-evpn-evi) # exit
Router (config-evpn-evi) # advertise-mac
Router (config-evpn-evi) # exit
Router (config-evpn) # evi 22022
Router (config-evpn-evi) # bgp
Router (config-evpn-evi-bgp) # route-target import 64: 22022
Router (config-evpn-evi-bgp) # route-target export 64: 22022
Router (config-evpn-evi-bgp) # exit
Router (config-evpn-evi) # advertise-mac
Router (config-evpn-evi) # commit
Router (config-evpn-evi) # exit

```

Running Configuration

```

/* Configure attachment circuits */
interface Bundle-Ether22001.12 l2transport
encapsulation dot1q 1 second-dot1q 12
!
interface Bundle-Ether22001.13 l2transport
encapsulation dot1q 1 second-dot1q 13
!
interface Bundle-Ether22001.14 l2transport
encapsulation dot1q 1 second-dot1q 14
!
interface Bundle-Ether22001.1 l2transport
encapsulation dot1q 1 second-dot1q 1
!
interface Bundle-Ether22001.2 l2transport
encapsulation dot1q 1 second-dot1q 2
!
interface Bundle-Ether22001.3 l2transport
encapsulation dot1q 1 second-dot1q 3
!
interface Bundle-Ether22001.4 l2transport
encapsulation dot1q 1 second-dot1q 4

/*Configure VLAN Unaware FXC Service */
flexible-xconnect-service vlan-unaware fxc_mh1
interface Bundle-Ether22001.1
interface Bundle-Ether22001.2
interface Bundle-Ether22001.3
neighbor evpn evi 21006 target 22016

```

```

!
/*Configure VLAN Aware FXC Service */
l2vpn
  flexible-xconnect-service vlan-aware evi 24001
    interface Bundle-Ether22001.12
    interface Bundle-Ether22001.13
    interface Bundle-Ether22001.14

/* Configure Local Switching */
flexible-xconnect-service vlan-aware evi 31400
  interface Bundle-Ether22001.1400
  interface Bundle-Ether23001.1400
!
flexible-xconnect-service vlan-aware evi 31401
  interface Bundle-Ether22001.1401
  interface Bundle-Ether23001.1401
!

/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */
interface Bundle-Ether22001.11 l2transport
  encapsulation dot1q 1 second-dot1q 11
  rewrite ingress tag pop 2 symmetric
!
interface Bundle-Ether22001.21 l2transport
  encapsulation dot1q 1 second-dot1q 21
  rewrite ingress tag pop 2 symmetric
!
!
l2vpn
xconnect group xg22001
p2p evpn-vpws-mclag-22001
  interface Bundle-Ether22001.11
  neighbor evpn evi 22101 target 220101 source 220301
!
bridge group native_evpn1
  bridge-domain bd21
  interface Bundle-Ether22001.21
    routed interface BVI21
    evi 22021
!
/* Configure Native EVPN */
Evpn
interface Bundle-Ether22001
  ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
  bgp route-target 2200.0001.0001
!
  evi 24001
    bgp
      route-target import 64:24001
      route-target export 64:24001
!
  evi 21006
    bgp
      route-target 64:100006
!
  evi 22101
    bgp
      route-target import 64:22101
      route-target export 64:22101
!
  evi 22021
    bgp
      route-target import 64:22021
      route-target export 64:22021

```

```

!
advertise-mac
!
evi 22022
  bgp
    route-target import 64:22022
    route-target export 64:22022
!
advertise-mac
!

```

Verification

Verify if each of the services is configured on the sub-interface.

```

Router# show l2vpn xconnect summary
Number of groups: 6
Number of xconnects: 505 Up: 505 Down: 0 Unresolved: 0 Partially-programmed: 0
AC-PW: 505 AC-AC: 0 PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
Advertised: 0 Non-Advertised: 0

```

```

Router# show l2vpn flexible-xconnect-service summary
Number of flexible xconnect services: 74
  Up: 74

```

```

Router# show l2vpn flexible-xconnect-service name fxc_mh1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
Flexible XConnect Service Segment
Name      ST  Type  Description  ST
-----
fxc_mh1  UP  AC:   BE22001.1   UP
          AC:   BE22001.2   UP
          AC:   BE22001.3   UP
-----

```

```

Router# show l2vpn flexible-xconnect-service name evi:24001
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
Flexible XConnect Service Segment
Name      ST  Type  Description  ST
-----
evi:24001 UP  AC:   BE22001.11  UP
          AC:   BE22001.12  UP
          AC:   BE22001.13  UP
          AC:   BE22001.14  UP
-----

```

```

Router# show l2vpn xconnect group xg22001 xc-name evpn-vpws-mclag-22001
Fri Sep 1 17:28:58.259 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed
XConnect
Group      Name                               ST      Segment 1      Segment 2
          Name                               Description ST      Description      Description      ST
-----

```

```
-----
xg22001  evpn-vpws-mclag-22001  UP  BE22001.101  UP  EVPN 22101, 220101,64.1.1.6  UP
-----
```

Associated Commands

- evpn
- evi
- ethernet-segment
- advertise-mac
- show evpn ethernet-segment
- show evpn evi
- show evpn summary
- show l2vpn xconnect summary
- show l2vpn flexible-xconnect-service
- show l2vpn xconnect group

EVPN MPLS Seamless Integration with VPLS

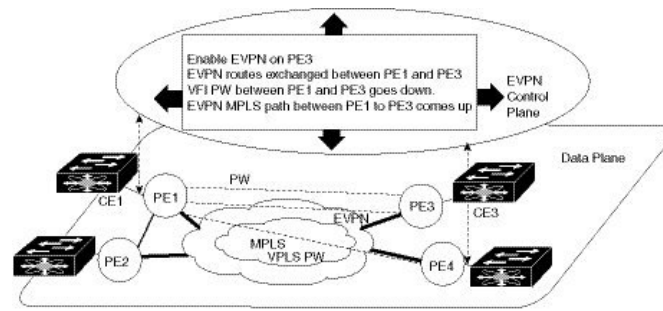
Seamless integration of EVPN MPLS with VPLS enables the co-existence of PE nodes running EVPN and VPLS for the same VPN instance. VPLS or legacy network can be upgraded to the next generation EVPN network without service disruption. You can introduce EVPN service on all the selected VPLS provider edge (PE) nodes simultaneously. However, to avoid traffic disruption, provision EVPN service on existing VPLS-enabled PEs one by one.

Migrate VPLS Network to EVPN Network through Seamless Integration

In EVPN network, VPN instances are identified by EVPN instance ID (EVI-ID). Similar to other L2VPN technologies, EVPN instances are also associated with route-targets and route-distinguisher. EVPN uses control plane for learning and propagating MAC unlike traditional VPLS, where MAC is learnt in the data plane (learns using "flood and learn technique"). In EVPN, MAC routes are carried by MP-BGP protocol. In EVPN enabled PEs, PEs import the MAC route along with the label to their respective EVPN forwarding table only if their route targets (RTs) match. An EVPN PE router is capable of performing VPLS and EVPN L2 bridging in the same VPN instance. When both EVPN and BGP-AD PW are configured in a VPN instance, the EVPN PEs advertise the BGP VPLS auto-discovery (AD) route as well as the BGP EVPN Inclusive Multicast route (type-3) for a given VPN Instance. Route type-3 referred to as ingress replication multicast route, is used to send broadcast, unknown unicast, and multicast (BUM) traffic. Other remote PEs import type-3 routes for the same VPN instance only if the sending PE RTs match with their configured RT. Thus, at the end of these route-exchanges, EVPN capable PEs discover all other PEs in the VPN instance and their associated capabilities. The type-3 routes used by PE to send its BUM traffic to other PEs ensure that PEs with the same RTs receive the BUM traffic. EVPN advertises the customer MAC address using type-2 route.

EVPN MPLS Seamless Integration with VPLS allows you to upgrade the VPLS PE routers to EVPN one by one without any network service disruption. Consider the following topology where PE1, PE2, PE3, and PE4 are interconnected in a full-meshed network using VPLS PW.

Figure 6: EVPN MPLS Seamless Integration with VPLS



The EVPN service can be introduced in the network one PE node at a time. The VPLS to EVPN migration starts on PE1 by enabling EVPN in a VPN instance of VPLS service. As soon as EVPN is enabled, PE1 starts advertising EVPN inclusive multicast route to other PE nodes. Since PE1 does not receive any inclusive multicast routes from other PE nodes, VPLS pseudo wires between PE1 and other PE nodes remain active. PE1 keeps forwarding traffic using VPLS pseudo wires. At the same time, PE1 advertises all MAC address learned from CE1 using EVPN route type-2. In the second step, EVPN is enabled in PE3. PE3 starts advertising inclusive multicast route to other PE nodes. Both PE1 and PE3 discover each other through EVPN routes. As a result, PE1 and PE3 shut down the pseudo wires between them. EVPN service replaces VPLS service between PE1 and PE3. At this stage, PE1 keeps running VPLS service with PE2 and PE4. It starts EVPN service with PE3 in the same VPN instance. This is called EVPN seamless integration with VPLS. The VPLS to EVPN migration then continues to remaining PE nodes. In the end, all four PE nodes are enabled with EVPN service. VPLS service is completely replaced with EVPN service in the network. All VPLS pseudo wires are shut down.

Configure EVPN on the Existing VPLS Network

Perform the following tasks to configure EVPN on the existing VPLS network.

- Configure L2VPN EVPN address-family
- Configure EVI and corresponding BGP route-targets under EVPN configuration mode
- Configure EVI under a bridge-domain

See [EVI Configuration Under L2VPN Bridge-Domain, on page 27](#) section for how to migrate various VPLS-based network to EVPN.

Configure L2 EVPN Address-Family

Perform this task to enable EVPN address family under both BGP and participating neighbor.

Configuration Example

```
Router# configure
Router(config)#router bgp 65530
Router(config-bgp)#nsr
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#bgp router-id 200.0.1.1
Router(config-bgp)#address-family l2vpn evpn
```

```

Router(config-bgp-af) #exit
Router(config-bgp) #neighbor 200.0.4.1
Router(config-bgp-nbr) #remote-as 65530
Router(config-bgp-nbr) #update-source Loopback0
Router(config-bgp-nbr) #address-family 12vpn evpn
Router(config-bgp-nbr-af) #commit

```

Running Configuration

```

configure
router bgp 65530
  nsr
  bgp graceful-restart
  bgp router-id 200.0.1.1
  address-family 12vpn evpn
  !
  neighbor 200.0.4.1
  remote-as 65530
  update-source Loopback0
  address-family 12vpn evpn
  !
!

```

Configure EVI and Corresponding BGP Route Target under EVPN Configuration Mode

Perform this task to configure EVI and define the corresponding BGP route targets. Also, configure advertise-mac, else the MAC routes (type-2) are not advertised.

Configuration Example

```

Router# configure
Router(config) #evpn
Router(config-evpn) #evi 1
Router(config-evpn-evi-bgp) #bgp
Router(config-evpn-evi-bgp) #table-policy spp-basic-6
Router(config-evpn-evi-bgp) #route-target import 100:6005
Router(config-evpn-evi-bgp) #route-target export 100:6005
Router(config-evpn-evi-bgp) #exit
Router(config-evpn-evi) #advertise-mac
Router(config-evpn-evi) #commit

```

Running Configuration

```

configure
evpn
  evi
    bgp
      table-policy spp-basic-6
      route-target import 100:6005
      route-target export 100:6005
    !
  advertise-mac
  !
!
!

```

Configure EVI under a Bridge Domain

Perform this task to configure EVI under the corresponding L2VPN bridge domain.

Configuration Example

```
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/0/0/0
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#evi 1
Router(config-l2vpn-bg-bd-evi)#exit
Router(config-l2vpn-bg-bd)#vfi v1
Router(config-l2vpn-bg-bd-vfi)#neighbor 10.1.1.2 pw-id 1000
Router(config-l2vpn-bg-bd-vfi-pw)#mpls static label local 20001 remote 10001
Router(config-l2vpn-bg-bd-vfi-pw)#commit
```

Running Configuration

```
configure
l2vpn
  bridge group bg1
  bridge-domain bd1
  interface GigabitEthernet0/0/0/0
  !
  evi 1
  !
  vfi v1
  neighbor 10.1.1.2 pw-id 1000
  mpls static label local 20001 remote 10001
  !
  !
  evi 1
!
```

EVI Configuration Under L2VPN Bridge-Domain

The following examples show EVI configuration under L2VPN bridge-domain for various VPLS-based networks:

MPLS Static Labels Based VPLS

```
l2vpn
  bridge group bg1
  bridge-domain bd-1-1
  interface GigabitEthernet0/0/0/0
  !
  vfi vfi-1-1
  neighbor 200.0.2.1 pw-id 1200001
  mpls static label local 20001 remote 10001
  !
  neighbor 200.0.3.1 pw-id 1300001
```

```

    mpls static label local 30001 remote 10001
    !
    neighbor 200.0.4.1 pw-id 1400001
    mpls static label local 40001 remote 10001
    !
    !
    evi 1
    !

```

AutoDiscovery BGP and BGP Signalling Based VPLS

```

l2vpn
bridge group bg1
bridge-domain bd-1-2
    interface GigabitEthernet0/0/0/2
    !
    vfi vfi-1-2
    vpn-id 2
    autodiscovery bgp
    rd 101:2
    route-target 65530:200
    signaling-protocol bgp
    ve-id 11
    ve-range 16
    !
    !
    evi 2
    !

```

Targeted LDP-Based VPLS

```

bridge-domain bd-1-4
    interface GigabitEthernet0/0/0/4
    !
    vfi vfi-1-4
    neighbor 200.0.2.1 pw-id 1200004
    !
    neighbor 200.0.3.1 pw-id 1300004
    !
    neighbor 200.0.4.1 pw-id 1400004
    !
    evi 3
    !

```

Verify EVPN Configuration

Use the following commands to verify EVPN configuration and MAC advertisement. Verify EVPN status, AC status, and VFI status.

- show l2vpn bridge-domain
- show evpn summary
- show bgp rt l2vpn evpn
- show evpn evi
- show l2route evpn mac all

```

Router#show l2vpn bridge-domain bd-name bd-1-1
Mon Feb 20 21:03:40.244 EST
Legend: pp = Partially Programmed.
Bridge group: bgl, bridge-domain: bd-1-1, id: 0, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 1, PWs: 3 (2 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
    Gi0/2/0/0.1, state: up, Static MAC addresses: 0, MSTi: 2
  List of Access PWs:
  List of VFIs:
    VFI vfi-1-1 (up)
      Neighbor 200.0.2.1 pw-id 1200001, state: up, Static MAC addresses: 0
      Neighbor 200.0.3.1 pw-id 1300001, state: down, Static MAC addresses: 0
      Neighbor 200.0.4.1 pw-id 1400001, state: up, Static MAC addresses: 0
  List of Access VFIs:
  When PEs are evpn enabled, pseudowires that are associated with that BD will be brought
  down. The VPLS BD pseudowires are always up.

```

Verify the number of EVI's configured, local and remote MAC-routes that are advertised.

```

Router#show evpn summary
Mon Feb 20 21:05:16.755 EST
-----
Global Information
-----
Number of EVIs                : 6
Number of Local EAD Entries    : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 4
    MAC                        : 4
    MAC-IPv4                   : 0
    MAC-IPv6                   : 0
Number of Local ES:Global MAC : 1
Number of Remote MAC Routes   : 0
    MAC                        : 0
    MAC-IPv4                   : 0
    MAC-IPv6                   : 0
Number of Remote SOO MAC Routes : 0
Number of Local IMCAST Routes : 4
Number of Remote IMCAST Routes : 4
Number of Internal Labels     : 0
Number of ES Entries          : 1
Number of Neighbor Entries    : 4
EVPN Router ID                : 200.0.1.1
BGP ASN                       : 65530
PBB BSA MAC address           : 0026.982b.c1e5
Global peering timer          : 3 seconds
Global recovery timer         : 30 seconds

```

Verify EVPN route-targets.

```

Router#show bgp rt l2vpn evpn
Mon Feb 20 21:06:18.882 EST
EXTCOMM      IMP/EXP
RT:65530:1   1 / 1
RT:65530:2   1 / 1
RT:65530:3   1 / 1
RT:65530:4   1 / 1
Processed 4 entries

```

```

Locally learnt MAC routes can be viewed by forwarding table
show l2vpn forwarding bridge-domain mac-address location 0/0/cpu0
To Resynchronize MAC table from the Network Processors, use the command...
l2vpn resynchronize forwarding mac-address-table location <r/s/i>

```

Mac Address	Type	Learned from/Filtered on	LC learned	Resync Age/Last Change	Mapped to
0033.0000.0001	dynamic	Gi0/2/0/0.1	N/A	20 Feb 21:06:59	N/A
0033.0000.0002	dynamic	Gi0/2/0/0.2	N/A	20 Feb 21:06:59	N/A
0033.0000.0003	dynamic	Gi0/2/0/0.3	N/A	20 Feb 21:04:29	N/A
0033.0000.0004	dynamic	Gi0/2/0/0.4	N/A	20 Feb 21:06:59	N/A

```

The remote routes learned via evpn enabled BD
show l2vpn forwarding bridge-domain mac-address location 0/0$
To Resynchronize MAC table from the Network Processors, use the command...
l2vpn resynchronize forwarding mac-address-table location <r/s/i>

```

Mac Address	Type	Learned from/Filtered on	LC learned	Resync Age/Last Change	Mapped to
0033.0000.0001	EVPN	BD id: 0	N/A	N/A	N/A
0033.0000.0002	EVPN	BD id: 1	N/A	N/A	N/A
0033.0000.0003	EVPN	BD id: 2	N/A	N/A	N/A
0033.0000.0004	EVPN	BD id: 3	N/A	N/A	N/A

Verify EVPN MAC routes pertaining to specific VPN instance.

```

Router#show evpn evi vpn-id 1 mac
Mon Feb 20 21:36:23.574 EST

```

EVI Label	MAC address	IP address	Next hop
1	0033.0000.0001	::	200.0.1.1 45106

Verify L2 routing.

```

Router#show l2route evpn mac all
Mon Feb 20 21:39:43.953 EST

```

Topo ID	Mac Address	Prod	Next Hop(s)
0	0033.0000.0001	L2VPN	200.0.1.1/45106/ME
1	0033.0000.0002	L2VPN	200.0.1.1/45108/ME
2	0033.0000.0003	L2VPN	200.0.1.1/45110/ME
3	0033.0000.0004	L2VPN	200.0.1.1/45112/ME

Verify EVPN route-type 2 routes.

```

Router#show bgp l2vpn evpn route-type 2
Mon Feb 20 21:43:23.616 EST
BGP router identifier 200.0.3.1, local AS number 65530

```

```

BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
                i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[2][0][48][0033.0000.0001][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[2][0][48][0033.0000.0002][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.1.1:3
*>i[2][0][48][0033.0000.0003][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.1.1:4
*>i[2][0][48][0033.0000.0004][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[2][0][48][0033.0000.0001][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[2][0][48][0033.0000.0002][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
*>i[2][0][48][0033.0000.0003][0]/104
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
*>i[2][0][48][0033.0000.0004][0]/104
                200.0.1.1                      100      0 i

Processed 8 prefixes, 8 paths

```

Verify inclusive multicast routes and route-type 3 routes.

```

Router#show bgp l2vpn evpn route-type 3
Mon Feb 20 21:43:33.970 EST
BGP router identifier 200.0.3.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
                i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[3][0][32][200.0.1.1]/80
                200.0.1.1                      100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[3][0][32][200.0.1.1]/80
                200.0.1.1                      100      0 i

```

```

Route Distinguisher: 200.0.1.1:3
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
Route Distinguisher: 200.0.1.1:4
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
*> [3][0][32][200.0.3.1]/80
      0.0.0.0                    0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
*> [3][0][32][200.0.3.1]/80
      0.0.0.0                    0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
*> [3][0][32][200.0.3.1]/80
      0.0.0.0                    0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
*>i[3][0][32][200.0.1.1]/80
      200.0.1.1                100      0 i
*> [3][0][32][200.0.3.1]/80
      0.0.0.0                    0 i

```

Network Convergence using Core Isolation Protection

The Network Convergence using Core Isolation Protection feature allows the router to converge fast when remote links and local interfaces fail. This feature reduces the duration of traffic drop by rapidly rerouting traffic to alternate paths. This feature uses Object Tracking (OT) to detect remote link failure and failure of connected interfaces.

Tracking interfaces can only detect failure of connected interfaces and not failure of a remote router interfaces that provides connectivity to the core. Tracking one or more BGP neighbor sessions along with one or more of the neighbor's address-families enables you to detect remote link failure.

Object Tracking

Object tracking (OT) is a mechanism for tracking an object to take any client action on another object as configured by the client. The object on which the client action is performed may not have any relationship to the tracked objects. The client actions are performed based on changes to the properties of the object being tracked.

You can identify each tracked object by a unique name that is specified by the track command in the configuration mode.

The tracking process receives the notification when the tracked object changes its state. The state of the tracked objects can be up or down.

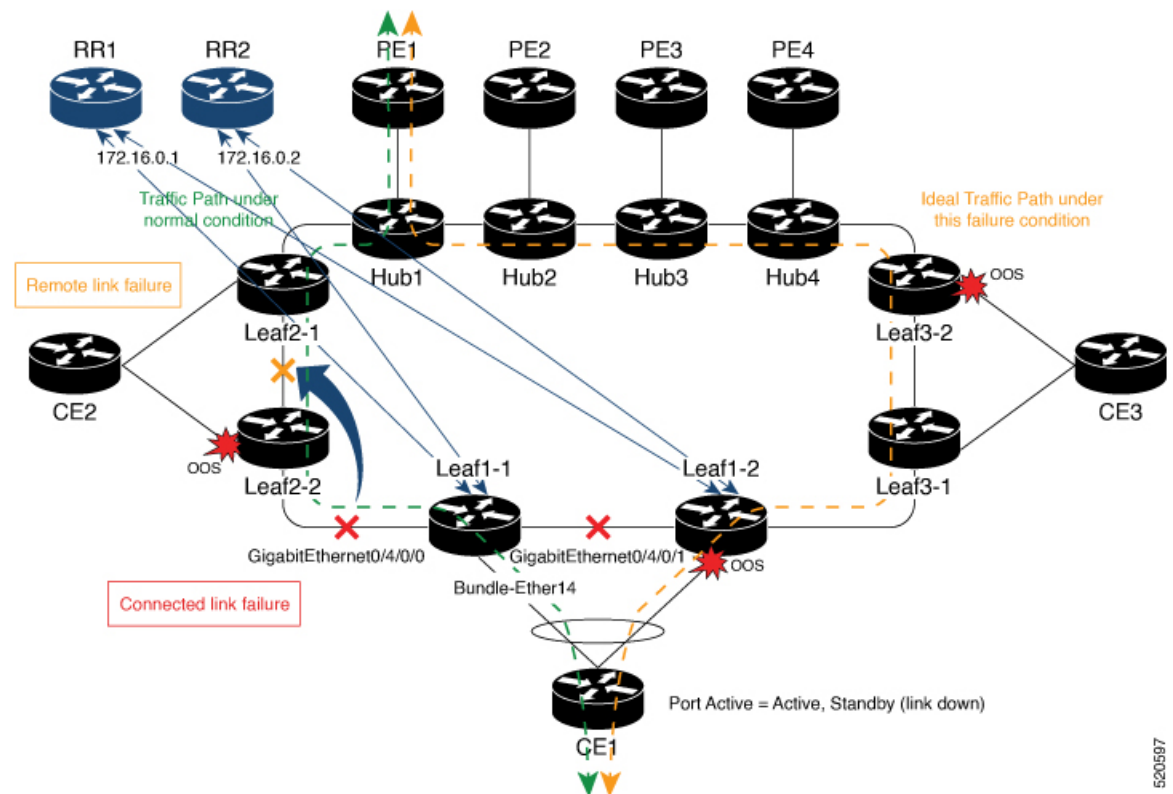
You can also track multiple objects by a list. You can use a flexible method for combining objects with Boolean logic. This functionality includes:

- Boolean AND function—When a tracked list has been assigned a Boolean AND function, each object defined within a subset must be in an up state, so that the tracked object can also be in the up state.

- Boolean OR function—When the tracked list has been assigned a Boolean OR function, it means that at least one object defined within a subset must also be in an up state, so that the tracked object can also be in the up state.

For more information on OT, see the *Configuring Object Tracking* chapter in the *System Management Configuration Guide for Cisco NCS 5500 Series Routers*.

Figure 7: EVPN Convergence Using Core Isolation Protection



Consider a traffic flow from CE1 to PE1. The CE1 can send the traffic either from Leaf1-1 or Leaf1-2. When Leaf1-1 loses the connectivity to both the local links and remote link, BGP sessions to both route reflectors (RRs) are down; the Leaf1-1 brings down the Bundle-Ether14 connected to CE1. The CE1 redirects the traffic from Leaf1-2 to PE1.

You can track the connected interfaces to identify the connected link failures. However, if there is a remote link failure, tracking connected interfaces does not identify the remote link failures. You must track BGP sessions to identify the remote link failure.



Note When you configure the **bgp graceful-restart** command, unconfiguring a neighbor is considered as a non-gr event. This generates a BGP notification to the neighbor before the neighbor is unconfigured.

On the remote router, if the track is configured for this neighbor, the track state is brought down immediately.

However, certain configurations are treated as graceful reset reason and when unconfigured they suppress the BGP notification to the neighbor. The route-reflector-client configuration under the neighbor or neighbor address-family is one of the examples.

On the remote router, if the track is configured for this neighbor, the track state is not brought down immediately because a notification is not received.

To overcome this situation, shutdown the neighbor before unconfiguring the neighbor. This generates a BGP notification to the neighbor, and any track configured for the neighbor is brought down immediately.

Configure EVPN Convergence using Core Isolation Protection

A tracked list contains one or more objects. The Boolean expression enables tracking objects using either AND or OR operators. For example, when tracking two interfaces, using the AND operator, up means that *both* interfaces are up, and down means that *either* interface is down.



Note An object must exist before it can be added to a tracked list.

The NOT operator is specified for one or more objects and negates the state of the object.

After configuring the tracked object, you must associate the neighbor or interface whose state must be tracked.

Perform the following tasks to configure EVPN convergence using core isolation protection:

- Configure BGP
- Track the Line Protocol State of an Interface
- Track neighbor address-family state
- Track objects for both interfaces and neighbors

Configuration Example

In this example, Leaf1-1 brings the down the AC connected to CE1 when:

Both local interfaces GigabitEthernet0/4/0/0 and GigabitEthernet0/4/0/1 of Leaf1-1 are down.

OR

Leaf1-1 BGP sessions to both RRs are down.

CE1 re-directs the traffic it was sending to Leaf1-1 to Leaf1-2.

Perform the following tasks on Leaf1-1:

```
/* Configure BGP */
Router# configure
```

```

Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af) # exit
Router(config-bgp) # neighbor 172.16.0.1
Router(config-bgp-nbr) # remote-as 100
Router(config-bgp-nbr) # address-family l2vpn evpn
Router(config-bgp-nbr-af) # neighbor 172.16.0.1
Router(config-bgp-nbr) # remote-as 100
Router(config-bgp-nbr) # address-family l2vpn evpn
Router(config-bgp-nbr-af) # commit

/* Track the Line Protocol State of an Interface */
Router# configure
Router(config)# track interface-1
Router(config-track) # type line-protocol state
Router(config-track-line-prot) # interface GigabitEthernet0/4/0/0
Router(config-track-line-prot) # exit
Router(config-track) # exit
Router(config)# track interface-2
Router(config-track) # type line-protocol state
Router(config-track-line-prot) # interface GigabitEthernet0/4/0/1
Router(config-track-line-prot) # exit
Router(config-track) # exit
Router(config)# track interface-group-1
Router(config-track) # type list boolean or
Router(config-track-list-boolean) # object interface-1
Router(config-track-list-boolean) # object interface-2
Router(config-track-list-boolean) # commit

/* Track neighbor address-family state */
Router# configure
Router(config)# track neighbor-A
Router(config-track) # type bgp neighbor address-family state
Router(config-track-bgp-nbr-af) # address-family l2vpn evpn
Router(config-track-bgp-neighbor) # neighbor 172.16.0.1
Router(config-track-bgp-neighbor) # exit
Router(config-track-bgp-nbr-af) # exit
Router(config-track) # exit
Router(config)# track neighbor-B
Router(config-track) # type bgp neighbor address-family state
Router(config-track-bgp-nbr-af) # address-family l2vpn evpn
Router(config-track-bgp-neighbor) # neighbor 172.16.0.2
Router(config-track-bgp-neighbor) # exit
Router(config-track-bgp-nbr-af) # exit
Router(config-track) # exit
Router(config)# track neighbor-group-1
Router(config-track) # type list boolean or
Router(config-track-list-boolean) # object neighbor-A
Router(config-track-list-boolean) # object neighbor-B
Router(config-track-list-boolean) # commit

/* Track objects for both interfaces and neighbors */
Router# configure
Router(config)# track core-group-1
Router(config-track) # type list boolean and
Router(config-track-list-boolean) # object neighbor-group-1
Router(config-track-list-boolean) # object interface-group-1
Router(config-track-list-boolean) # action
Router(config-track-action) # track-down error-disable interface Bundle-Ether14 auto-recover
Router(config-track-action) # commit

```

Running Configuration

This section shows EVPN convergence using core isolation protection running configuration.

```

router bgp 100
  address-family l2vpn evpn
  !
  neighbor 172.16.0.1
    remote-as 100
    address-family l2vpn evpn
  !
  !
  neighbor 172.16.0.2
    remote-as 100
    address-family l2vpn evpn
  !
  !
!

track interface-1
  type line-protocol state
  interface GigabitEthernet0/4/0/0
  !
!
track interface-2
  type line-protocol state
  interface GigabitEthernet0/4/0/1
  !
!
track interface-group-1
  type list boolean or
  object interface-1
  object interface-2
  !
!

track neighbor-A
  type bgp neighbor address-family state
  address-family l2vpn evpn
  neighbor 172.16.0.1
  !
!
!
track neighbor-B
  type bgp neighbor address-family state
  address-family l2vpn evpn
  neighbor 172.16.0.1
  !
!
!

track neighbor-group-1
  type list boolean or
  object neighbor-A
  object neighbor-B
  !
!
!
track core-group-1
  type list boolean and
  object neighbor-group-1
  object interface-group-1
  !
action

```

```

track-down error-disable interface Bundle-Ether14 auto-recover
!
!

```

Verification

Verify that you have configured the EVPN convergence using core isolation protection feature successfully.

```

Router# show track
Wed May 27 04:42:11.995 UTC

```

```

Track neighbor-A
  BGP Neighbor AF L2VPN EVPN NBR 172.16.0.1 vrf default
  Reachability is UP
    Neighbor Address Reachability is Up
    BGP Neighbor Address-family state is Up
  4 changes, last change UTC Tue May 26 2020 20:14:33.171

```

```

Track neighbor-B
  BGP Neighbor AF L2VPN EVPN NBR 172.16.0.2 vrf default
  Reachability is UP
    Neighbor Address Reachability is Up
    BGP Neighbor Address-family state is Up
  4 changes, last change UTC Tue May 26 2020 20:14:27.527

```

```

Track core-group-1
  List boolean and is UP
  2 changes, last change 20:14:27 UTC Tue May 26 2020
  object interface-group-1 UP
  object neighbor-group-1 UP

```

```

Track interface-1
  Interface GigabitEthernet0/4/0/0 line-protocol
  Line protocol is UP
  2 changes, last change 20:13:32 UTC Tue May 26 2020

```

```

Track interface-2
  Interface GigabitEthernet0/4/0/1 line-protocol
  Line protocol is UP
  2 changes, last change 20:13:28 UTC Tue May 26 2020

```

```

Track interface-group-1
  List boolean or is UP
  2 changes, last change 20:13:28 UTC Tue May 26 2020
  object interface-2 UP
  object interface-1 UP

```

```

Track neighbor-group-1
  List boolean or is UP
  2 changes, last change 20:14:27 UTC Tue May 26 2020
  object neighbor-A UP
  object neighbor-B UP

```

```

Router# show track brief
Wed May 27 04:39:19.740 UTC

```

Track	Object	Parameter
Value		
neighbor-A	bgp nbr L2VPN EVPN 172.16.0.1 vrf defau	reachability
Up		
neighbor-B	bgp nbr L2VPN EVPN 172.16.0.1 vrf defau	reachability
Up		

```

core-group-1          list          boolean and
  Up
interface-1          interface GigabitEthernet0/4/0/0  line protocol
  Up
interface-2          interface GigabitEthernet0/4/0/1  line protocol
  Up
interface-group-1    list          boolean or
  Up
neighbor-group-1     list          boolean or
  Up

```

```
Router# show bgp track
```

```
Wed May 27 05:05:51.285 UTC
```

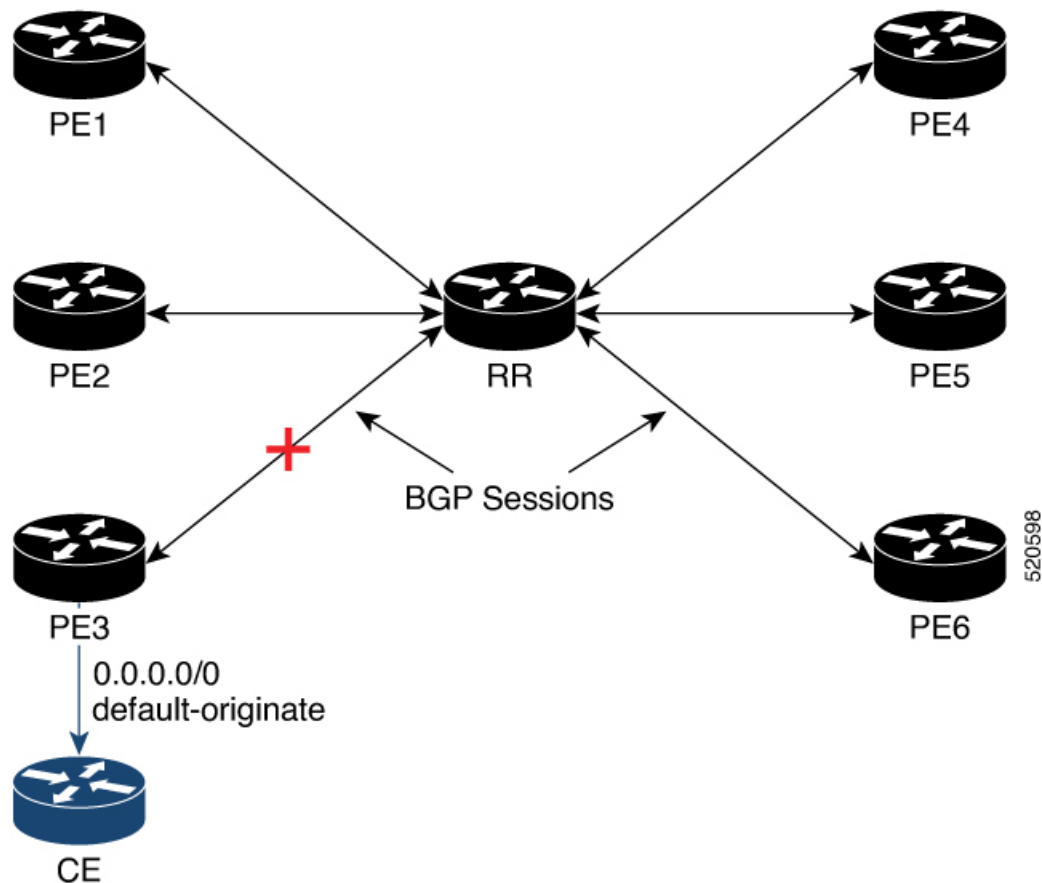
VRF	Address-family	Neighbor	Status	Flags
default	L2VPN EVPN	172.16.0.1	UP	0x01
default	L2VPN EVPN	172.16.0.2	UP	0x01

```
Processed 2 entries
```

Conditional Advertisement of Default-Originate

The router advertises the default-originate (0.0.0.0/0) towards the network fabric only upon receiving all the core routes. The router withdraws the advertisement of default-originate when the core is isolated. To avoid traffic drop, install the routes in the hardware. To accommodate an additional delay for the routes to be installed in the hardware, you can configure a timeout for the installed routes.

Figure 8: Advertisement of default-originate



In this topology, PE3 advertises the default-originate to CE only when the PE3 session to RR is established and all the routes are received from the RR.

Configure Conditional Advertisement of Default-Originate

Perform the following tasks to configure conditional advertisement of default-originate.

- Configure BGP
- Configure RPL
- Track BGP neighbor address-family state

Configuration Example

Perform the following task on PE3:

```
/* Configure BGP */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 192.0.2.1
Router(config-bgp)# address-family vpnv4 unicast
```

```

Router(config-bgp-af) # exit
Router(config-bgp) # neighbor 172.16.0.5
Router(config-bgp-nbr) # remote-as 200
Router(config-bgp-nbr) # address-family vpnv4 unicast
Router(config-bgp-nbr-af) # exit
Router(config-bgp-nbr) # exit
Router(config-bgp) # vrf cust1
Router(config-bgp-vrf) # rd auto
Router(config-bgp-vrf) # address-family ipv4 unicast
Router(config-bgp-vrf-af) # redistribute connected
Router(config-bgp-vrf-af) # redistribute static
Router(config-bgp-vrf-af) # exit
Router(config-bgp-vrf) # neighbor 172.16.0.5
Router(config-bgp-vrf-nbr) # remote-as 200
Router(config-bgp-vrf-nbr) # address-family ipv4 unicast
Router(config-bgp-vrf-nbr-af) # default-originate route-policy track-bgp-core-policy
Router(config-bgp-vrf-nbr-af) # route-policy pass in
Router(config-bgp-vrf-nbr-af) # route-policy pass out
Router(config-bgp-vrf-nbr-af) # commit

/* Configure RPL */
Router# configure
Router(config)# route-policy track-bgp-core-policy
Router(config-rpl)# if track core-group-1 is up then
Router(config-rpl-if) # pass
Router(config-rpl-if) # endif
Router(config-rpl) # end-policy
Router(config) # commit

/* Track BGP neighbor address-family state */
Router# configure
Router(config)# track core-group-1
Router(config-track)# type bgp neighbor address-family state
Router(config-track-bgp-nbr-af) # address-family vpnv4 unicast
Router(config-track-bgp-neighbor) # neighbor 172.16.0.5
Router(config-track-bgp-neighbor) # commit

```

Running Configuration

This section shows conditional advertisement of default-originate running configuration.

```

configure
router bgp 100
  bgp router-id 192.0.2.1
  address-family vpnv4 unicast
!
neighbor 172.16.0.5
  remote-as 200
  address-family vpnv4 unicast
!

vrf cust1
  rd auto
  address-family ipv4 unicast
    redistribute connected
    redistribute static
!
neighbor 172.16.0.5
  remote-as 200
  address-family ipv4 unicast
    default-originate route-policy track-bgp-core-policy

```



```

    route-policy pass in
    route-policy pass out
!

route-policy track-bgp-core-policy
  if track core-group-1 is up then
    pass
  endif
end-policy
!
track network-core
  type bgp neighbor address-family state
  address-family vpv4 unicast
  neighbor 172.16.0.5
!

```

Verification

Verify conditional advertisement of default-originate.

```

Router# show rpl active route-policy
Wed May 27 06:54:31.902 UTC

ACTIVE -- Referenced by at least one policy which is attached
INACTIVE -- Only referenced by policies which are not attached
UNUSED -- Not attached (directly or indirectly) and not referenced

```

The following policies are (ACTIVE)

```

-----
track-bgp-core
-----

```

```

Router# show rpl route-policy track-bgp-core-policy
Wed May 27 06:54:38.090 UTC
route-policy track-bgp-core-policy
  if track core-group-1 is up then
    pass
  endif
end-policy
!

```

```

Router# show bgp policy route-policy track-bgp-core-policy summary
Wed May 27 06:54:42.823 UTC
Network          Next Hop          From              Advertised to
0.0.0.0/0        0.0.0.0           Local             172.16.0.5

```

```

Router# show bgp neighbor 172.16.0.5
Wed May 27 06:55:39.535 UTC

```

```

BGP neighbor is 172.16.0.5
  Remote AS 9730, local AS 9730, internal link
  Remote router ID 172.16.0.5
  BGP state = Established, up for 10:41:12
[snip]
For Address Family: IPv4 Unicast
  BGP neighbor version 2
  Update group: 0.4 Filter-group: 0.1 No Refresh request being processed
  Default information originate: default route-policy track-bgp-core-policy, default sent
  AF-dependent capabilities:
[snip]
Track Enabled, Status UP, Nbr GR state Not Enabled, EOR tmr Not Running
Advertise routes with local-label via Unicast SAFI

```

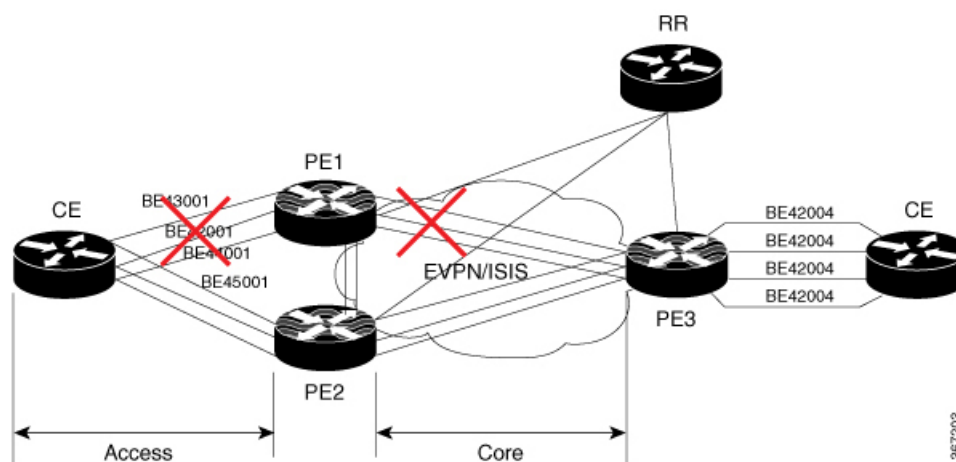
EVPN Core Isolation Protection

The EVPN Core Isolation Protection feature enables you to monitor and detect the link failure in the core. When a core link failure is detected in the provider edge (PE) device, EVPN brings down the PE's Ethernet Segment (ES), which is associated with access interface attached to the customer edge (CE) device.

EVPN replaces ICCP in detecting the core isolation. This new feature eliminates the use of ICCP in the EVPN environment.

Consider a topology where CE is connected to PE1 and PE2. PE1, PE2, and PE3 are running EVPN over the MPLS core network. The core interfaces can be Gigabit Ethernet or bundle interface.

Figure 9: EVPN Core Isolation Protection



When the core links of PE1 go down, the EVPN detects the link failure and isolates PE1 node from the core network by bringing down the access network. This prevents CE from sending any traffic to PE1. Since BGP session also goes down, the BGP invalidates all the routes that were advertised by the failed PE. This causes the remote PE2 and PE3 to update their next-hop path-list and the MAC routes in the L2FIB. PE2 becomes the forwarder for all the traffic, thus isolating PE1 from the core network.

When all the core interfaces and BGP sessions come up, PE1 advertises Ethernet A-D Ethernet Segment (ES-EAD) routes again, triggers the service carving and becomes part of the core network.

Configure EVPN Core Isolation Protection

Configure core interfaces under EVPN group and associate that group to the Ethernet Segment which is an attachment circuit (AC) attached to the CE. When all the core interfaces go down, EVPN brings down the associated access interfaces which prevents the CE device from using those links within their bundles. All interfaces that are part of a group go down, EVPN brings down the bundle and withdraws the ES-EAD route.

Restrictions

- A maximum of 24 groups can be created under the EVPN.
- A maximum of 12 core interfaces can be added under the group.
- The core interfaces can be reused among the groups. The core interface can be a bundle interface.

- EVPN group must only contain core interfaces, do not add access interfaces under the EVPN group.
- The access interface can only be a bundle interface.
- EVPN core facing interfaces must be physical or bundle main interfaces only. Sub-interfaces are not supported.

```

Router# configure
Router(config)# evpn
Router(config-evpn)# group 42001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/1
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/3
Router(config-evpn-group)#exit
!
Router(config-evpn)# group 43001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/2
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/4
Router(config-evpn-group)#exit
!
Router# configure
Router(config)# evpn
Router(config-evpn)# interface bundle-Ether 42001
Router(config-evpn-ac)# core-isolation-group 42001
Router(config-evpn-ac)# exit
!
Router(config-evpn)# interface bundle-Ether 43001
Router(config-evpn-ac)# core-isolation-group 43001
Router(config-evpn-ac)# commit

```

Running Configuration

```

configure
evpn
  group 42001
    core interface GigabitEthernet0/2/0/1
    core interface GigabitEthernet0/2/0/3
    !
  group 43001
    core interface GigabitEthernet0/2/0/2
    core interface GigabitEthernet0/2/0/4
    !
!
configure
evpn
  interface bundle-Ether 42001
    core-isolation-group 42001
    !
  interface bundle-Ether 43001
    core-isolation-group 43001
    !
!

```

Verification

The **show evpn group** command displays the complete list of evpn groups, their associated core interfaces and access interfaces. The status, up or down, of each interface is displayed. For the access interface to be up, at least one of the core interfaces must be up.

```
Router# show evpn group /* Lists specific group with core-interfaces and access interface
status */
EVPN Group: 42001
  State: Ready
  Core Interfaces:
    Bundle-Ethernet110: down
    Bundle-Ethernet111: down
    GigabethEthernet0/2/0/1: up
    GigabethEthernet0/2/0/3: up
    GigabethEthernet0/4/0/8: up
    GigabethEthernet0/4/0/9: up
    GigabethEthernet0/4/0/10: up
  Access Interfaces:
    Bundle-Ether42001: up

EVPN Group: 43001
  State: Ready
  Core Interfaces:
    Bundle-Ethernet110: down
    GigabethEthernet0/2/0/2: up
    GigabethEthernet0/2/0/4: up
    GigabethEthernet0/4/0/9: up

  Access Interfaces:
    Bundle-Ether43001: up
```

EVPN Routing Policy

The EVPN Routing Policy feature provides the route policy support for address-family L2VPN EVPN. This feature adds EVPN route filtering capabilities to the routing policy language (RPL). The filtering is based on various EVPN attributes.

A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

This feature enables you to configure route-policies using EVPN network layer reachability information (NLRI) attributes of EVPN route type 1 to 5 in the route-policy match criteria, which provides more granular definition of route-policy. For example, you can specify a route-policy to be applied to only certain EVPN route-types or any combination of EVPN NLRI attributes. This feature provides flexibility in configuring and deploying solutions by enabling route-policy to filter on EVPN NLRI attributes.

To implement this feature, you need to understand the following concepts:

- Routing Policy Language
- Routing Policy Language Structure
- Routing Policy Language Components
- Routing Policy Language Usage
- Policy Definitions
- Parameterization
- Semantics of Policy Application
- Policy Statements

- Attach Points

For information on these concepts, see [Implementing Routing Policy](#).

Currently, this feature is supported only on BGP neighbor "in" and "out" attach points. The route policy can be applied only on inbound or outbound on a BGP neighbor.

EVPN Route Types

The EVPN NLRI has the following different route types:

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet (AD) routes are advertised on per EVI and per Ethernet Segment Identifier (ESI) basis. These routes are sent per Ethernet segment (ES). They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed.

An Ethernet A-D route type specific EVPN NLRI consists of the following fields:

```

+-----+
|Route Type (1 octet)                |*
+-----+
|Length (1 octet)                    |
+-----+
|Route Distinguisher (RD) (8 octets) |*
+-----+
|Ethernet Segment Identifier (10 octets)|*
+-----+
|Ethernet Tag ID (4 octets)          |*
+-----+
|MPLS Label (3 octets)               |
+-----+

```

NLRI Format: Route-type 1:

[Type] [Len] [RD] [ESI] [ETag] [MPLS Label]

Net attributes: [Type] [RD] [ESI] [ETag]

Path attributes: [MPLS Label]

Example

```

route-policy evpn-policy
  if rd in (1.1.1.1:0) [and/or evpn-route-type is 1] [and/or esi in (0a1.a2a3.a4a5.a6a7.a8a9)]
    [and/or etag is 4294967295] then
    set ..
  endif
end-policy
!
route-policy evpn-policy
  if rd in (1.1.1.2:0) [and/or evpn-route-type is 1] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy

```

Route Type 2: MAC/IP Advertisement Route

The host's IP and MAC addresses are advertised to the peers within NLRI. The control plane learning of MAC addresses reduces unknown unicast flooding.

A MAC/IP Advertisement Route type specific EVPN NLRI consists of the following fields:

```

+-----+
|Route Type (1 octet)          |*
+-----+
|Length (1 octet)             |
+-----+
|RD (8 octets)                 |*
+-----+
|Ethernet Segment Identifier (10 octets)|
+-----+
|Ethernet Tag ID (4 octets)    |*
+-----+
|MAC Address Length (1 octet)  |*
+-----+
|MAC Address (6 octets)        |*
+-----+
|IP Address Length (1 octet)   |*
+-----+
|IP Address (0, 4, or 16 octets)|*
+-----+
|MPLS Label1 (3 octets)       |
+-----+
|MPLS Label2 (0 or 3 octets)  |
+-----+

```

3063156

NLRI Format: Route-type 2:

[Type][Len][RD][ESI][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr][MPLS Label1][MPLS Label2]

Net attributes: [Type][RD][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr]

Path attributes: [ESI], [MPLS Label1], [MPLS Label2]

Example

```

route-policy evpn-policy
  if rd in (1.1.1.2:0) [and/or evpn-route-type is 2] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or macaddress in (0013.aabb.cccd)]
[and/or destination in (1.2.3.4/32)] then
    set ..
  endif
end-policy

```

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

An Inclusive Multicast Ethernet Tag route type specific EVPN NLRI consists of the following fields:

Route Type (1 octet)	*
Length (1 octet)	
RD (8 octets)	*
Ethernet Tag ID (4 octets)	*
IP Address Length (1 octet)	*
Originating Router's IP Address (4 or 16 octets)	*

NLRI Format: Route-type 3:

[Type][Len][RD][ETag][IP Addr Len][Originating Router's IP Addr]

Net attributes: [Type][RD][ETag][IP Addr Len][Originating Router's IP Addr]

Example

```
route-policy evpn-policy
  if rd in (1.1.1.1:300) [and/or evpn-route-type is 3] [and/or etag is 0] [and/or
  evpn-originator in (1.1.1.1)] then
    set ..
  endif
end-policy
```

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

An Ethernet Segment route type specific EVPN NLRI consists of the following fields:

```

+-----+
|Route Type (1 octet)          |*
+-----+
|Length (1 octet)             |
+-----+
|RD (8 octets)                 |*
+-----+
|Ethernet Segment Identifier (10 octets)|*
+-----+
|IP Address Length (1 octet)   |*
+-----+
|Originating Router's IP Address |*
|(4 or 16 octets)             |
+-----+

```

3/6/2016

NLRI Format: Route-type 4:

[Type][Len][RD][ESI][IP Addr Len][Originating Router's IP Addr]

Net attributes: [Type][RD][ESI][IP Addr Len][Originating Router's IP Addr]

Example

```

route-policy evpn-policy
  if rd in (1.1.1.1:0) [and/or evpn-route-type is 4] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or evpn-originator in (1.1.1.1)] then
    set ..
  endif
end-policy

```

Route Type 5: IP Prefix Route

An IP Prefix Route type specific EVPN NLRI consists of the following fields:

Route Type (1 octet)	*
Length (1 octet)	
RD (8 octets)	*
Ethernet Segment Identifier (10 octets)	
Ethernet Tag ID (4 octets)	*
IP Address Length (1 octet)	*
IP Address (4 or 16 octets)	*
GW IP Address (4 or 16 octets)	
MPLS Label (3 octets)	

NLRI Format: Route-type 5:

[Type][Len][RD][ESI][ETag][IP Addr Len][IP Addr][GW IP Addr][Label]

Net attributes: [Type][RD][ETag][IP Addr Len][IP Addr]

Path attributes: [ESI], [GW IP Addr], [Label]

Example

```
route-policy evpn-policy
  if rd in (30.30.30.30:1) [and/or evpn-route-type is 5] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or destination in (12.2.0.0/16)] [and/or
evpn-gateway in (0.0.0.0)] then
    set ..
  endif
end-policy
```

EVPN RPL Attribute

Route Distinguisher

A Route Distinguisher (rd) attribute consists of eight octets. An rd can be specified for each of the EVPN route types. This attribute is not mandatory in route-policy.

Example

```
rd in (1.2.3.4:0)
```

EVPN Route Type

EVPN route type attribute consists of one octet. This specifies the EVPN route type. The EVPN route type attribute is used to identify a specific EVPN NLRI prefix format. It is a net attribute in all EVPN route types.

Example

```
evpn-route-type is 3
```

The following are the various EVPN route types that can be used:

```
1 - ethernet-ad
2 - mac-advertisement
3 - inclusive-multicast
4 - ethernet-segment
5 - ip-advertisement
```

IP Prefix

An IP prefix attribute holds IPv4 or IPv6 prefix match specification, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. When IP prefix is specified in EVPN route type 2, it represents either a IPv4 or IPv6 host IP Address (/32 or /128). When IP prefix is specified in EVPN route type 5, it represents either IPv4 or IPv6 subnet. It is a net attribute in EVPN route type 2 and 5.

Example

```
destination in (128.47.10.2/32)
destination in (128.47.0.0/16)
destination in (128:47::1/128)
destination in (128:47::0/112)
```

esi

An Ethernet Segment Identifier (ESI) attribute consists of 10 octets. It is a net attribute in EVPN route type 1 and 4, and a path attribute in EVPN route type 2 and 5.

Example

```
esi in (ffff.ffff.ffff.ffff.fff0)
```

etag

An Ethernet tag attribute consists of four octets. An Ethernet tag identifies a particular broadcast domain, for example, a VLAN. An EVPN instance consists of one or more broadcast domains. It is a net attribute in EVPN route type 1, 2, 3 and 5.

Example

```
etag in (10000)
```

mac

The mac attribute consists of six octets. This attribute is a net attribute in EVPN route type 2.

Example

```
mac in (0206.acb1.e806)
```

evpn-originator

The evpn-originator attribute specifies the originating router's IP address (4 or 16 octets). This is a net attribute in EVPN route type 3 and 4.

Example

```
evpn-originator in (1.2.3.4)
```

evpn-gateway

The evpn-gateway attribute specifies the gateway IP address. The gateway IP address is a 32-bit or 128-bit field (IPv4 or IPv6), and encodes an overlay next-hop for the IP prefixes. The gateway IP address field can be zero if it is not used as an overlay next-hop. This is a path attribute in EVPN route type 5.

Example

```
evpn-gateway in (1.2.3.4)
```

EVPN RPL Attribute Set

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The prefix-set specifies one or more IP prefixes.

Example

```
prefix-set ip_prefix_set
14.2.0.0/16,
54.0.0.0/16,
12.12.12.0/24,
50:50::1:0/112
end-set
```

mac-set

The mac-set specifies one or more MAC addresses.

Example

```
mac-set mac_address_set
1234.2345.6789,
2345.3456.7890
end-set
```

esi-set

The esi-set specifies one or more ESI's.

Example

```
esi-set evpn_esi_set
1234.2345.3456.4567.5678,
1234.2345.3456.4567.5670
end-set
```

etag-set

The etag-set specifies one or more Ethernet tags.

Example

```
etag-set evpn_etag_set
10000,
20000
end-set
```

Configure EVPN RPL Feature

The following section describe how to configure mac-set, esi-set, evpn-gateway, and evpn-originator.

```
/* Configuring a mac-set and referring it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# mac-set demo_mac_set
Router(config-mac)# 1234.ffff.aaa3,
Router(config-mac)# 2323.4444.ffff
Router(config-mac)# end-set
Router(config)# !
Router(config)# route-policy policy_use_pass_mac_set
Router(config-rpl)# if mac in demo_mac_set then
Router(config-rpl-if)# set med 200
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 1000
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```

```

Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af) # !
Router(config-bgp-af) # neighbor 10.0.0.10
Router(config-bgp-nbr) # remote-as 8
Router(config-bgp-nbr) # address-family l2vpn evpn
Router(config-bgp-nbr-af) # route-policy policy_use_pass_mac_set in
Router(config-bgp-nbr-af) # commit

/* Configuring a esi-set and referring it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# esi-set demo_esi
Router(config-esi) # ad34.1233.1222.ffff.44ff,
Router(config-esi) # ad34.1233.1222.ffff.6666
Router(config-esi) # end-set
Router(config) # !
Router(config) # route-policy use_esi
Router(config-rpl) # if esi in demo_esi then
Router(config-rpl-if) # set local-preference 100
Router(config-rpl-if) # else
Router(config-rpl-else) # set local-preference 300
Router(config-rpl-else) # endif
Router(config-rpl) # end-policy
Router(config) # commit

/* Configuring evpn-gateway/evpn-originator in a route-policy (Attach point - neighbor-in
and out) */
Router# configure
Router(config) # route-policy gateway_demo
Router(config-rpl) # if evpn-gateway in (10.0.0.0/32) then
Router(config-rpl-if) # pass
Router(config-rpl-if) # endif
Router(config-rpl) # end-policy
Router(config) # commit
Router(config) # route-policy originator_demo
Router(config-rpl) # if evpn-originator in (10.0.0.1/32) then
Router(config-rpl-if) # set local-preference 100
Router(config-rpl-if) # else
Router(config-rpl-else) # set med 200
Router(config-rpl-else) # endif
Router(config-rpl) # end-policy
Router(config) # commit
Router(config) # router bgp 100
Router(config-bgp) # address-family ipv4 unicast
Router(config-bgp-af) # !
Router(config-bgp-af) # neighbor 10.0.0.10
Router(config-bgp-nbr) # remote-as 8
Router(config-bgp-nbr) # address-family ipv4 unicast
Router(config-bgp-nbr-af) # route-policy gateway_demo in
Router(config-bgp-nbr-af) # route-policy originator_demo out
Router(config-bgp-nbr-af) # commit

```

Running Configuration

```

/* Configuring a mac-set and referring it in a route-policy (Attach point - neighbor-in) */
mac-set demo_mac_set
  1234.ffff.aaa3,
  2323.4444.ffff
end-set
!
route-policy policy_use_pass_mac_set

```

```

    if mac in demo_mac_set then
        set med 200
    else
        set med 1000
    endif
end-policy
!
router bgp 100
  address-family l2vpn evpn
  !
  neighbor 10.0.0.10
    remote-as 8
    address-family l2vpn evpn
    route-policy policy_use_pass_mac_set in
  !
  !
end

/* Configuring a esi-set and referring it in a route-policy (Attach point - neighbor-in) */
Wed Oct 26 11:52:23.720 IST
esi-set demo_esi
  ad34.1233.1222.ffff.44ff,
  ad34.1233.1222.ffff.6666
end-set
!
route-policy use_esi
  if esi in demo_esi then
    set local-preference 100
  else
    set local-preference 300
  endif
end-policy

```

EVPN Route Policy Examples

```

route-policy ex_2
  if rd in (2.2.18.2:1004) and evpn-route-type is 1 then
    drop
  elseif rd in (2.2.18.2:1009) and evpn-route-type is 1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy ex_3
  if evpn-route-type is 5 then
    set extcommunity bandwidth (100:9999)
  else
    pass
  endif
end-policy
!
route-policy samp
end-policy
!
route-policy samp1
  if rd in (30.0.101.2:0) then
    pass
  endif
end-policy

```

```
!
route-policy samp2
  if rd in (30.0.101.2:0, 1:1) then
    pass
  endif
end-policy
!
route-policy samp3
  if rd in (*:*) then
    pass
  endif
end-policy
!
route-policy samp4
  if rd in (30.0.101.2:*) then
    pass
  endif
end-policy
!
route-policy samp5
  if evpn-route-type is 1 then
    pass
  endif
end-policy
!
route-policy samp6
  if evpn-route-type is 2 or evpn-route-type is 5 then
    pass
  endif
end-policy
!
route-policy samp7
  if evpn-route-type is 4 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp8
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp9
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
  is 4 then
    pass
  endif
end-policy
!
route-policy test1
  if evpn-route-type is 2 then
    set next-hop 10.2.3.4
  else
    pass
  endif
end-policy
!
route-policy test2
  if evpn-route-type is 2 then
    set next-hop 10.10.10.10
  else
    drop
  endif
end-policy
```

```
end-policy
!
route-policy test3
  if evpn-route-type is 1 then
    set tag 9988
  else
    pass
  endif
end-policy
!
route-policy samp21
  if mac in (6000.6000.6000) then
    pass
  endif
end-policy
!
route-policy samp22
  if extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp23
  if evpn-route-type is 1 and esi in (aaaa.bbbb.cccc.dddd.eeee) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp24
  if evpn-route-type is 5 and extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp25
  if evpn-route-type is 2 and esi in (1234.1234.1234.1234.1236) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp26
  if etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp27
  if destination in (99.99.99.1) and etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
```



```
route-policy samp31
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
  is 4 or evpn-route-type is 5 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp33
  if esi in evpn_esi_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp34
  if destination in (90:1:1::9/128) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp35
  if destination in evpn_prefix_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp36
  if evpn-route-type is 3 and evpn-originator in (80:1:1::3) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp37
  if evpn-gateway in (10:10::10) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp38
  if mac in evpn_mac_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp39
  if mac in (6000.6000.6002) then
    pass
  else
    drop
  endif
end-policy
```

```
!  
route-policy samp41  
  if evpn-gateway in (10.10.10.10, 10:10::10) then  
    pass  
  else  
    drop  
  endif  
end-policy  
!  
route-policy samp42  
  if evpn-originator in (24.162.160.1/32, 70:1:1::1/128) then  
    pass  
  else  
    drop  
  endif  
end-policy  
!  
route-policy example  
  if rd in (62300:1903) and evpn-route-type is 1 then  
    drop  
  elseif rd in (62300:19032) and evpn-route-type is 1 then  
    drop  
  else  
    pass  
  endif  
end-policy  
!  
route-policy samp100  
  if evpn-route-type is 4 or evpn-route-type is 5 then  
    drop  
  else  
    pass  
  endif  
end-policy  
!  
route-policy samp101  
  if evpn-route-type is 4 then  
    drop  
  else  
    pass  
  endif  
end-policy  
!  
route-policy samp102  
  if evpn-route-type is 4 then  
    drop  
  elseif evpn-route-type is 5 then  
    drop  
  else  
    pass  
  endif  
end-policy  
!  
route-policy samp103  
  if evpn-route-type is 2 and destination in evpn_prefix_set1 then  
    drop  
  else  
    pass  
  endif  
end-policy  
!  
route-policy samp104  
  if evpn-route-type is 1 and etag in evpn_etag_set1 then  
    drop
```

```

elseif evpn-route-type is 2 and mac in evpn_mac_set1 then
    drop
elseif evpn-route-type is 5 and esi in evpn_esi_set1 then
    drop
else
    pass
endif
end-policy
!

```

Support for DHCPv4 and DHCPv6 Client over BVI

The Support for DHCPv4 and DHCPv6 Client over the BVI feature allows you to configure DHCPv4 and DHCPv6 client on the Bridged Virtual Interface (BVI). You can configure a BVI, and request DHCP IPv4 or IPv6 address on the BVI. This allows your customer's device to have initial connectivity to your network without user intervention in the field. After the device is connected to your network, the customer devices can push a node-specific configuration with static IP addresses on a different BVI for customer deployment.

Configure DHCPv4 and DHCPv6 Client over BVI

Perform the following tasks to configure DHCPv4 and DHCPv6 client over BVI:

- Configure AC interface
- Configure L2VPN
- Configure BVI

Configuration Example

```

/* Configure AC interface */
Router# configure
Router(config)# interface tenGigE 0/5/0/1/1
Router(config-if)# bundle id 1 mode on
Router(config-if)# exit
Router(config)# interface Bundle-Ether1
Router(config-if)# no shut
Router(config-if)# exit
Router(config)# interface bundle-ether 1.100 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 100
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure L2VPN */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BVI
Router(config-l2vpn-bg)# bridge-domain bvi
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.100
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd-bvi)# commit

/* Configure BVI */
Router# configure
Router(config)# interface BVI1

```

```
Router(config-if)# ipv4 address dhcp
Router(config-if)# ipv6 address dhcp
Router(config-if)# commit
```

Running Configuration

This section shows the DHCPv4 and DHCPv6 client over BVI running configuration.

```
interface TenGigE0/5/0/1/1
bundle id 1 mode on
!
interface Bundle-Ether1
!
interface Bundle-Ether1.100 l2transport
encapsulation dot1q 100
rewrite ingress tag pop 1 symmetric
!
l2vpn
bridge group BVI
  bridge-domain bvi
    interface Bundle-Ether1.100
      !
      routed interface BVI1
      !
    !
  !
interface BVI1
ipv4 address dhcp
ipv6 address dhcp
!
```

Verification

The show output given in the following section display the details of DHCPv4 and DHCPv6 client over BVI configuration.

```
Router# show l2vpn bridge-domain
Legend: pp = Partially Programmed.
Bridge group: BVI, bridge-domain: bvi, id: 0, state: up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 64000, Action: none, Notification: syslog
Filter MAC addresses: 0
ACs: 2 (2 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
List of ACs:
  BVI, state: up, BVI MAC addresses: 1
  BE1.100, state: up, Static MAC addresses: 0
List of Access PWs:
List of VFIs:
List of Access VFIs:
```

```
Router# show dhcp ipv4 client
```

Interface name	IP Address	Binding State	Lease Time Rem
BVI1	172.16.0.2	BOUND	3598 secs (00:59:58)

```
Router# show dhcp ipv6 client
```

Interface name	IPv6 Address	State	Lease Time Rem
BVI1	2000::1	BOUND	2591982

Router# **show dhcp ipv4 client bvi1 detail**

```

-----
Client Interface name       : BVI1
Client Interface handle    : 0x8804054
Client ChAddr              : 008a.9628.ac8a
Client ID                  : BVI1.00:8a:96:28:ac:8a
Client State               : BOUND
Client IPv4 Address (Dhcp) : 172.16.0.2
Client IPv4 Address Mask   : 255.240.0.0
Client Lease Time Allocated : 3600 secs (01:00:00)
Client Lease Time Remaining : 3571 secs (00:59:31)
Client Selected Server Address: 172.16.0.1
Client Next Hop Address    : 0.0.0.0
-----

```

Router# **show dhcp ipv4 client BVI1 statistics**

```

Client Interface name       : BVI1
-----
CLIENT COUNTER(s)         | VALUE
-----
Num discovers sent         :      44
Num requests sent         :       1
Num offers received       :       1
Num acks received         :       1
-----

```

Router# **show dhcp ipv6 client**

```

-----
Interface name             | IPv6 Address | State | Lease Time Rem
-----
BVI1                       | 2000::1     | BOUND | 2591685
-----

```

Router# **show dhcp ipv6 client statistics-all**

```

Interface name       : BVI1
Interface handle    : 0x8804054
VRF                 : 0x60000000

```

```

-----
TYPE | TRANSMIT | RECEIVE | DROP |
-----
SOLICIT | 17 | 0 | 0 |
ADVERTISE | 0 | 1 | 0 |
REQUEST | 1 | 0 | 0 |
REPLY | 0 | 2 | 0 |
CONFIRM | 0 | 0 | 0 |
RENEW | 1 | 0 | 0 |
REBIND | 0 | 0 | 0 |
RELEASE | 0 | 0 | 0 |
RECONFIG | 0 | 0 | 0 |
INFORM | 0 | 0 | 0 |
-----

```

```

-----
TIMER | STARTED | STOPPED | EXPIRED |
-----
INIT | 1 | 0 | 1 |
VBIND | 0 | 0 | 0 |
RENEW | 2 | 1 | 0 |
REBIND | 2 | 1 | 0 |
RETRANS | 19 | 3 | 16 |
-----

```

VALID | 2 | 1 | 0 |

Configure DHCPv6 Client Options

You can configure different DHCPv6 client options to differentiate between clients as required. Configure different DHCPv6 client options to differentiate how a DHCPv6 client communicates with a DHCPv6 server. The different DHCPv6 client options that you can configure are:

- **DUID:** If the DUID DHCPv6 client option is configured on an interface, DHCPv6 client communicates with the DHCPv6 server through the link layer address.
- **Rapid Commit:** If the Rapid Commit DHCPv6 client option is configured on an interface, DHCPv6 client can obtain configuration parameters from the DHCPv6 server through a rapid two-step exchange (solicit and reply) instead of the default four-step exchange (solicit, advertise, request, and reply).
- **DHCP Options:** The various other DHCPv6 options that can be configured on a DHCPv6 client are:
 - **Option 15:** Option 15 is also known as the User Class option and it is used by a DHCPv6 client to identify the type or category of users or applications it represents.
 - **Option 16:** Option 16 is also known as the Vendor ID option and it is used by a DHCPv6 a client to identify the vendor that manufactured the hardware on which the client is running.
 - **Option 23:** Option 23 is also known as the Domain name Server (DNS) option provides a list of one or more IPv6 addresses of DNS recursive name servers to which a client's DNS resolver can send DNS queries.
 - **Option 24:** Option 24 is also known as the Domain List option and it specifies the domain search list that the client uses to resolve hostnames with the DNS.
- **DHCP Timers:** This option is used to set different timer value for DHCP client configurations. The various DHCP timer options are:
 - **Release-timeout:** It is used to set retransmission timeout value for the initial release message.
 - **Req-max-rt:** It is used to set the maximum retransmission timeout value for the request message.
 - **Req-timeout:** It is used to set the initial request timeout value of the request message.
 - **Sol-max-delay:** It is used to set the maximum delay time of the first solicit message.
 - **Sol-max-rt:** It is used to set the maximum solicit retransmission time.
 - **Sol-time-out:** It is used to set the initial timeout value of the solicit message.

Configuration Example

Perform this task to configure DHCPv6 client options on a BVI interface.

```
Router# configure
Router(config)# interface BVI 10
Router(config-if)# ipv6 address dhcp-client-options
Router(config-dhcpv6-client)# duid linked-layer-address
Router(config-dhcpv6-client)# rapid-commit
Router(config-dhcpv6-client)# timers release-timeout 3
Router(config-dhcpv6-client)# timers sol-max-delay 1
Router(config-dhcpv6-client)# timers sol-time-out 1
```

```
Router(config-dhcpv6-client)# timers sol-max-rt 120
Router(config-dhcpv6-client)# timers req-max-rt 30
Router(config-dhcpv6-client)# timers req-timeout 1
Router(config-dhcpv6-client)# commit
```

Verification

To verify the DHCPv6 client options, use the **show dhcp ipv6 client BVI10 detail** command.

```
Router# show dhcp ipv6 client BVI10 detail
Wed Jun 10 16:19:21.272 IST
```

```
-----
Client Interface name : MgmtEth0/0/CPU0/1
Client Interface handle : 0x4040
Client MACAddr : 02f0.2b39.44be
Client State : BOUND
Client Link Local Address : fe80::f0:2bff:fe39:44be
Client IPv6 Address (Dhcp) : 600:1::12
Lease Remaining (in secs) : 74
DUID : 0003000102f02b3944be

Client Configuration
Timers
SOL_MAX_DELAY : 1 secs (00:00:01)
SOL_TIMEOUT : 1 secs (00:00:01)
SOL_MAX_RT : 120 secs (00:02:00)
REQ_TIMEOUT : 1 secs (00:00:01)
REQ_MAX_RT : 30 secs (00:00:30)
REL_TIMEOUT : 3 secs (00:00:01)

Options
RAPID-COMMIT : True
USER-CLASS : ciscoupnp
VENDOR-CLASS : vendor
DNS-SERVERS : True
DOMAIN-LIST : True

DUID Type : DUID_LL

Server Information
Server Address : fe80::d2:alff:feb2:3b9f
Preference : 0
DUID : 000300010206826e2e00
Status : SUCCESS
IA-NA
Status : SUCCESS
IAID : 0x40400001
T1 : 60 secs (00:01:00)
T2 : 96 secs (00:01:36)
IA-ADDR
IA NA Address : 600:1::12
Preferred Time : 120 secs (00:02:00)
Valid Time : 120 secs (00:02:00)
Flags : 0x0
```

Related Topics

- [Support for DHCPv4 and DHCPv6 Client over BVI, on page 59](#)

Associated Commands

- show l2vpn bridge-domain
- show dhcp ipv4 client
- show dhcp ipv6 client
- show dhcp ipv4 client bvi