Implementing DCI Layer 3 Gateway between MPLS-VPN and EVPN Data Center

This chapter module provides conceptual and configuration information for Data Center Interconnect (DCI) Layer 3 Gateway between MPLS-VPN and EVPN Data Center.

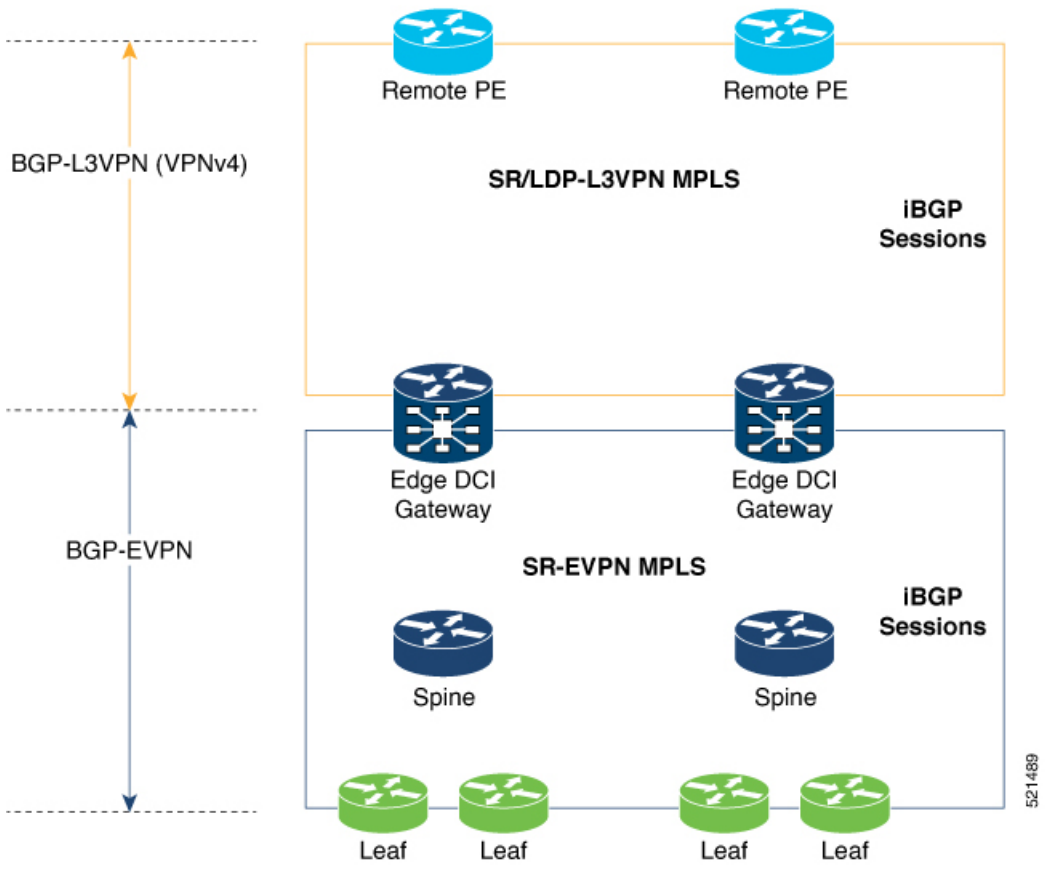
- Data Center Interconnect between MPLS-VPN and EVPN-MPLS , on page 1
- Data Center Interconnect between MPLS-VPN and EVPN-VxLAN, on page 44

Data Center Interconnect between MPLS-VPN and EVPN-MPLS

This part provides conceptual and configuration information for Data Center Interconnect (DCI) Layer 3 Gateway with EVPN-MPLS on Cisco ASR 9000 Series Router.

DCI Layer 3 Gateway with EVPN-MPLS

You can use SR-EVPN for Data Center on routers for a spine-leaf architecture with edge devices such as border leaf. DCI L3 stitching allows Data Centers that run SR-EVPN to communicate with legacy and existing MPLS VPN (VPNv4) sites.



In this topology,

Leaf (ToR) – Router acts as both access switch and distributed PE. Leaf establishes BGP EVPN neighborship with Spine route-reflector (RR). This router sends and receives prefixes from the DCI Gateway. Leaf ToR provides the following types of services:

- Regular L3 VRF configuration using subinterfaces to attach some CE devices. Traditional PE-CE scenario without EVPN configuration.
- L3 EVPN VRF using L2VPN configuration to attach multiple Data Centers services.

Leaf sends and receives prefixes from or to the DCI gateway:

- Leaf sends prefixes to DCI: Leaf re-originates local learned VRF subnet route as EVPN Route Type 5 with the EVPN RT (stitching-rt or regular RT), then sends to Spine RR. Spine RR sends prefixes to DCI gateway.
- Leaf receives prefixes from DCI: Leaf receives EVPN Route Type 5 from Spine RR that is re-originated at DCI gateway due to stitching between VPNv4 and EVPN. Leaf imports remote VPNv4 prefixes to local VRF matching VPNv4 RT (stitching-rt or regular RT).

Spine RR: Spine RR establishes BGP EVPN neighborship with Leaf (ToR) and Edge DCI Gateway serving as Route-Reflector for EVPN prefixes between the devices in the Data Center. Leaf and DCI Gateway must be configured as clients of Spine RR.

Edge (DCI gateway): Edge (DCI gateway) acts as an edge router that allows communication between services connected at Leaf and CEs in legacy MPLS network architecture. The edge DCI gateway establishes BGP EVPN neighborship with Spine RR and remote PEs, or RR depending on legacy MPLS network architecture.

The edge DCI gateway sends and receives prefixes from or to the Data Center:

- DCI gateway receives prefixes from legacy MPLS VPNv4 network and sends prefixes to Leaf: DCI gateway receives L3VPN (VPNv4) routes from remote MPLS VPN (VPNv4) PE or RR depending on legacy MPLS network architecture matching the VPNv4 RT (stitching-rt or regular RT). Then re-originate these prefixes as EVPN Route Type 5 with the EVPN RT (stitching-rt or regular RT) advertising to Spine RR due to BGP EVPN neighbor with the Spine.
- DCI gateway receives prefixes from Leaf and sends prefixes to legacy MPLS VPNv4 network: DCI gateway receives EVPN Route Type 5 originated from Leaf (ToR) by Spine RR due to BGP EVPN neighbor with the Spine. Leaf and DCI gateway does not have a direct BGP neighborship. Then import the routes to local VRF matching the EVPN RT (stitching-rt or regular RT) and re-originate this prefix as VPNv4 router with the VPNv4 RT (stitching-rt or regular RT) and advertise to remote MPLS VPN (VPNv4) PE or RR depending on legacy MPLS network architecture.

Remote PE: Remote PE receives traditional MPLS L3VPN prefixes (VPNv4) by DCI Gateway or RR depending on legacy MPLS network architecture. You must have a unique Route-Distinguisher (RD) between remote PEs and DCI gateway to allow stitching re-originate prefixes from VPNv4 to EVPN at DCI Gateway.

Stitching RTs and Regular RTs can be assigned to any side, EVPN or VPNv4, irrespective of the address-family. Consider the following supported scenarios:

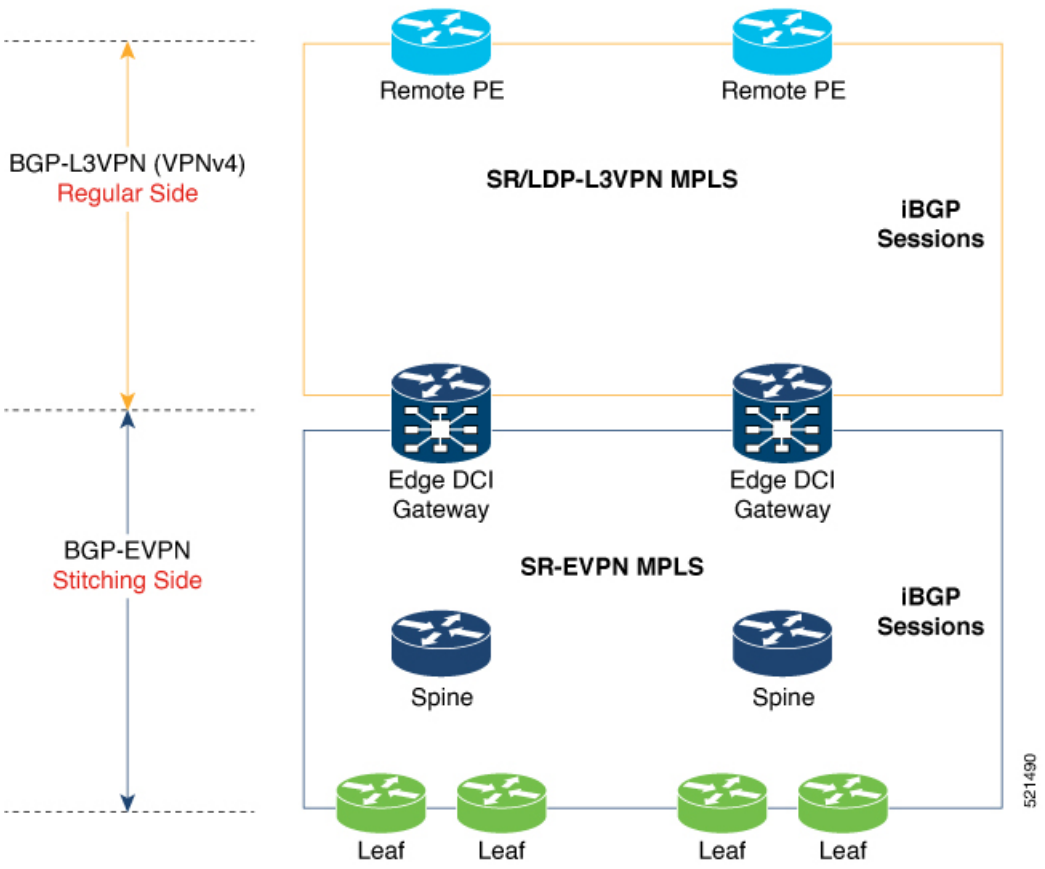
VPNv4-Regular RT and EVPN-Stitching RT

For each VRF on the DCI gateway, there are two sets of manually configured import and export route-targets for VPNv4 as a regular side and EVPN as a stitching side. Consider the following sets:

- Data Center Route-Targets for EVPN associated with EVPN BGP neighbor (Stitching RT).
- MPLS L3VPN Route-Targets for VPNv4 or VPNv6 associated with L3VPN BGP neighbor (Regular RT).

This separation of RTs enables the two sets of RTs to be independently configured. The RTs associated with the EVPN BGP neighbor require **stitching-rt** keyword under VRF configuration. The route-types associated with the L3VPN BGP neighbor do not require the keyword.

The following topology shows regular/normal and stitching side.



Route Targets

The RTs associated with the EVPN BGP neighbor are labelled as stitching RTs. The RTs associated with the L3VPN BGP neighbor are normal RTs.

Route Re-Origination

Consider control plane information propagation by the edge DCI gateway from the L3VPN (regular/normal side) to the Data Center (stitching side). Edge DCI gateway advertises to its BGP EVPN neighbor the routes that are re-originated after importing them from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the normal route-targets with the local route-target values (stitching-rt) associated with the BGP EVPN neighbors.

Route Address-Family and Encoded Address-Family

When an address-family is configured for a BGP neighbor, it means that the specified address-family routes encoded with the NLRI for that address-family are advertised to the neighbor. This does not hold for Data Center BGP neighbors because they use only EVPN address-family. Here, BGP neighbors advertise VPNv4 or VPNv6 unicast routes using the EVPN NLRI encoding. Thus, the encoded address-family and route address family can be possibly different. You can advertise the VPNv4 or VPNv6 address-family using the **advertise vpnv4 unicast** or **advertise vpnv6 unicast** command. For example, an EVPN address-family BGP neighbor configured with the **advertise vpnv4 unicast** command sends VPNv4 unicast routes in an EVPN encoded NLRI.

Local VPNv4 or VPNv6 Route Advertisement

On the edge DCI gateway, the locally sourced VPNv4 or VPNv6 routes (any CE directly connected not using L2VPN with BD/EVI/BVI, using only regular L3 VRF) can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets. You can configure this local VPNv4 or VPNv6 route advertisements to be advertised with stitching RTs to the BGP EVPN neighbors by using the **advertise vpnv4 unicast local stitching-rt** or **advertise vpnv6 unicast local stitching-rt** command as required.

VPNv4 neighbors do not require any additional configuration. By default, these routes are advertised with the normal route-targets to BGP L3VPN neighbors.

Route Distinguishers

The Router Distinguisher (RD) associated per VRF must be unique per PE in the network. There are few available options to keep unique RD per device:

- Manual configuration: You must manually assign a unique value per device in the network. For example, in this scenario:
  - Leaf (ToR) = RD 1
  - Edge DCI Gateway = RD 2
  - Remote PE = RD 3
- Use **rd auto** command under VRF. To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Note In a DCI deployment, for route re-originate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between edge DCI gateway and MPLS-VPN PE or same RD between edge DCI gateway and Leaf (ToR) is not supported.

Configure VPNv4-Regular RT and EVPN-Stitching RT

This section describes tasks to configure VPNv4-Regular RT and EVPN-Stitching RT. Perform the following tasks to complete the configuration:

- Configure Leaf (ToR)
- Configure Spine-RR (Route Reflector)
- Configure Edge DCI Gateway
- Configure EVPN BGP neighbor and route advertisements
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure Leaf (ToR)

Configure VRF in Leaf (ToR) at BGP-EVPN (Stitching Side) with Stitching-RT.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:2 stitching                              // BGP - EVPN (Stitching Side)
!
  export route-target
   1:2 stitching                             // BGP - EVPN (Stitching Side)
  !
router bgp 100
  neighbor 10.10.1.1                        // Spine Loopback IP Address
    address-family l2vpn evpn
       advertise vpnv4 unicast
       advertise vpnv6 unicast
  !
```

Note Advertise vpnv4/vpnv6 unicast enables local learned regular L3 VRF prefixes to be advertised as EVPN prefixes to BGP – EVPN neighbor. This means any local prefixes such as PE-CE without L2VPN with BD/EVI/BVI configuration. If all the services are pure EVPN with L2VPN with BD/EVI/BVI configuration these commands are not required.

Configure Spine-RR

Configure Spine RR with Leaf (ToR) and edge DCI gateway as RR client for AFI L2VPN EVPN. VRF configuration is not required.

```
// VRF Config is not required //

router bgp 100
  neighbor 10.10.2.1                 // Leaf (ToR) Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
   !
  neighbor 10.10.3.1                // Edge DCI Gateway  Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
!
```

Configure Edge DCI Gateway

You can configure DCI with the same VRF as Leaf (ToR). Use the same RT as remote PE for L3VPN network or the same VRF if that is possible.

Configure VRF and Route Targets Import and Export rules

Perform the following steps to configure VRF and define route targets to be used for import and export of forwarding information.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:1                              // BGP – L3VPN (Regular/normal Side)
```

```
   1:2 stitching                    // BGP - EVPN (Stitching Side)
!
  export route-target
   1:1                             // BGP - L3VPN (Regular/normal Side)
   1:2 stitching                   // BGP - EVPN (Stitching Side)
!
```

Configure EVPN BGP Neighbor and Route Advertisements

Perform this task on the edge DCI gateway to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
router bgp 100
 addreess-family l2vpn evpn
!
 neighbor 10.10.1.1               // Spine Loopback IP Address
  address-family l2vpn evpn
   import stitching-rt re-originate  //Imp EVPN 1:2, reoriginate VPNv4 RT 1:1
   advertise vpnv4 unicast re-originated stitching-rt //Send routes EVPN 1:2
   advertise vpnv6 unicast re-originated stitching-rt //Send routes EVPN 1:2
!
```

Configure L3VPN BGP Neighbor Relationship and Route Advertisements

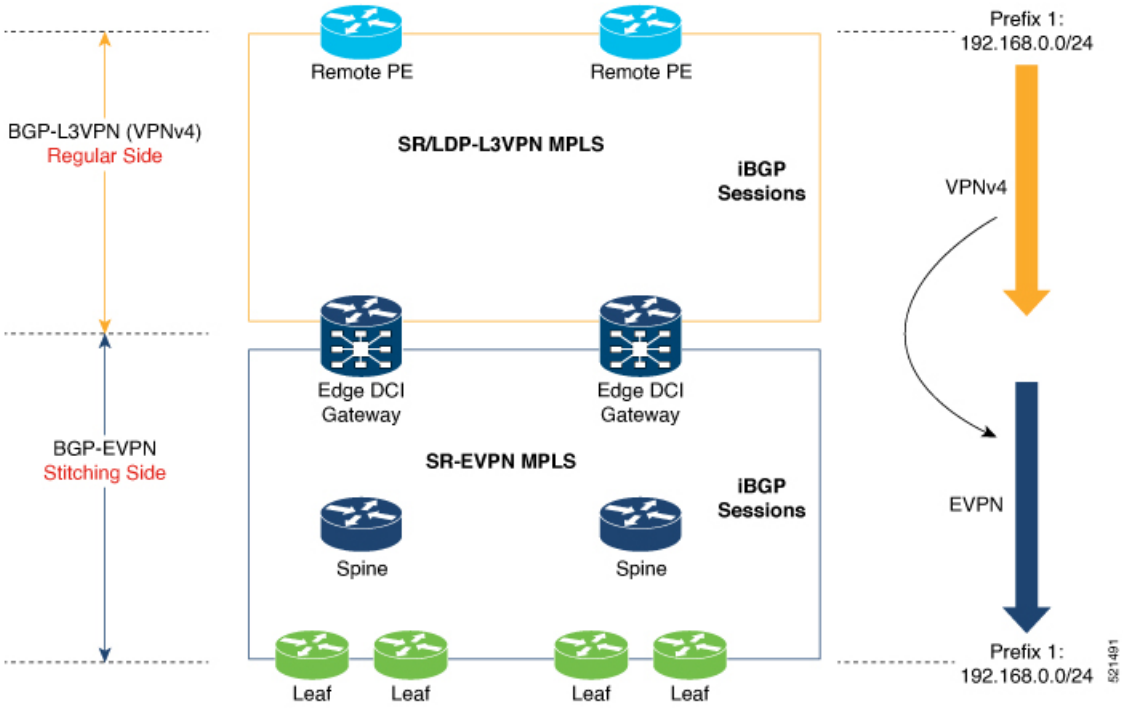
Perform the following steps to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
router bgp 100
 address-family vpnv4 unicast
!
 neighbor 10.10.1.1               // Spine Loopback IP Address
  address-family vpnv4 unicast      // Same config for VPNv6
   import re-originate stitching-rt // Imp VPNv4 1:1, re-originate EVPN 1:2
   advertise vpnv4 unicast re-originated   // Send routes VPNv4 RT 1:1
!
```

Configuration applies in two directions:

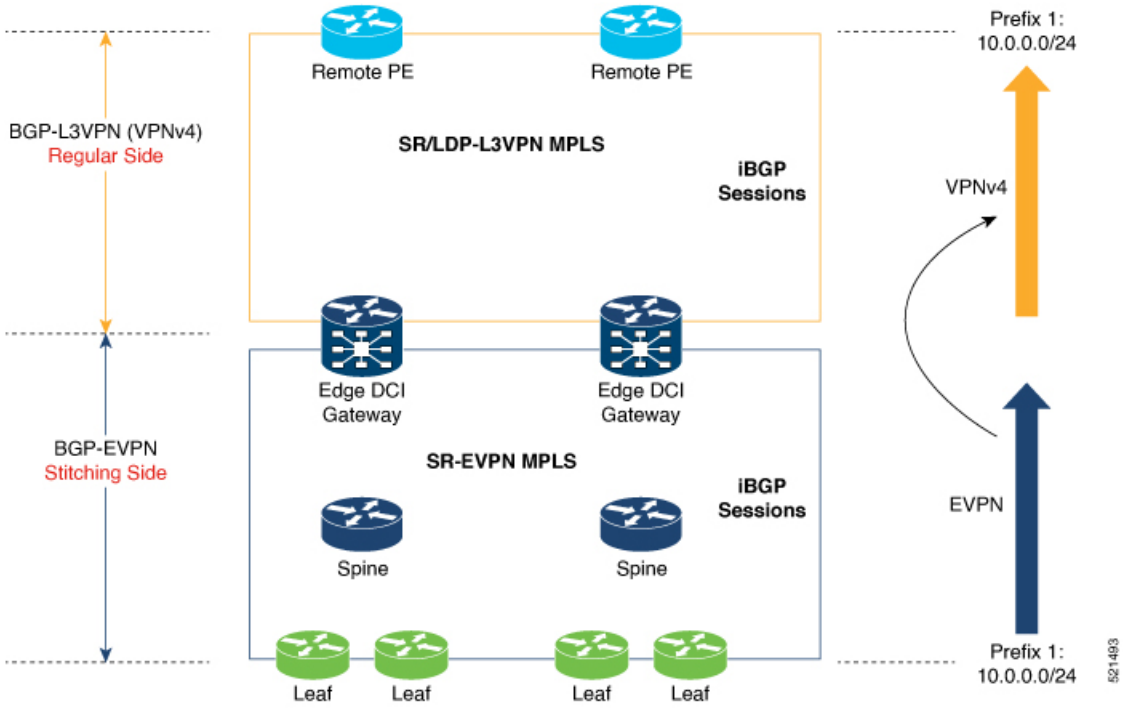
- Stitching from VPNv4 to EVPN routes. Prefixes received from MPLS L3VPN network and re-originated as EVPN prefixes towards Data Center Spine RR and Leaf (ToR).

1. Importing VPNv4 routes with import **re-originate stitching-rt** command under AFI VPNv4 UNICAST. This command imports routes using RT 1:1 and then reoriginate with BGP EVPN 1:2 **stitching-rt**.
2. Advertising re-originated EVPN routes with VPNv4 RT with advertise **vpvn4 unicast re-originated** command under AFI L2VPN EVPN. This command advertises routes from MPLS L3VPN network (VPNv4) to BGP EVPN neighbors inside Data Center (Spine RR and then Leaf (ToR)), re-originating these routes using BGP EVPN 1:2 **stitching-rt**.



- Stitching from EVPN to VPNv4 routes. Prefixes received from BGP-EVPN Data Center and re-originated as MPLS L3VPN prefixes towards VPNv4 RR or remote PE in L3VPN network.

1. Importing EVPN routes with import **stitching-rt re-originate** command under AFI L2VPN EVPN. This command imports routes using RT 1:2 **stitching-rt** and then re-originate with VPNv4 regular/normal VPNv4 RT 1:1.
2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated** command under AFI VPNv4 UNICAST. This command advertises routes from EVPN Data Center to VPNv4 RR or remote PEs, re-originating these routes using regular/normal VPNv4 RT 1:1.



Verification of Edge DCI Gateway Configuration

```
Router# show bgp l2vpn evpn

Fri Aug 21 00:24:10.773 PDT
BGP router identifier 30.30.30.30, local AS number 100
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 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
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: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
     11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
     11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
        11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
     11.0.0.1                      100      0 i
*>i[5][4231][32][100.1.1.1]/80
     11.0.0.1                      100      0 i
*>i[5][4231][32][100.1.1.2]/80
     11.0.0.1                      100      0 i
*>i[5][4231][112][fec0::1001]/176
     11.0.0.1                      100      0 i
```

```
*>i[5][4232][112][fec0::1:1001]/176
     11.0.0.1                 100      0 i

Processed 8 prefixes, 8 paths
```

```
Router# show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail

Fri Aug 21 00:34:43.747 PDT
BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
      Extended community: Flags 0x6: RT:1:1
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254
```

```
Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail
Fri Aug 21 00:25:37.383 PDT

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
    Address family VPNv6 Unicast:   Yes         No
```

```
  Address family L2VPN EVPN:      Yes         Yes
 Message stats:
  InQ depth: 0, OutQ depth: 0
                 Last_Sent               Sent  Last_Rcvd               Rcvd
  Open:          Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
  Notification:  ---                        0  ---                        0
  Update:        Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
  Keepalive:     Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
  Route_Refresh: Aug 21 00:24:01.377        3  ---                        0
  Total:                                   21                           23
 Minimum time between advertisement runs is 0 secs
 Inbound message logging enabled, 3 messages buffered
 Outbound message logging enabled, 3 messages buffered

For Address Family: VPNv4 Unicast
 BGP neighbor version 35
 Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
 Advertise Reorigination Enabled
 Advertise AFI EoR can be sent
 Route refresh request: received 0, sent 0
 0 accepted prefixes, 0 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 4, suppressed 0, withdrawn 0
 Maximum prefixes allowed 2097152
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was not received during read-only mode
 Last ack version 35, Last synced ack version 35
 Outstanding version objects: current 0, max 1
 Additional-paths operation: None
 Send Multicast Attributes

For Address Family: VPNv6 Unicast
 BGP neighbor version 29
 Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
 Advertise Reorigination Enabled
 Advertise AFI EoR can be sent
 Route refresh request: received 0, sent 0
 0 accepted prefixes, 0 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 0, suppressed 0, withdrawn 0
 Maximum prefixes allowed 1048576
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was not received during read-only mode
 Last ack version 29, Last synced ack version 29
 Outstanding version objects: current 0, max 0
 Additional-paths operation: None
 Send Multicast Attributes
 Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

For Address Family: L2VPN EVPN
 BGP neighbor version 18
 Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
 Route refresh request: received 0, sent 3
 8 accepted prefixes, 8 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 4, suppressed 0, withdrawn 6
 Maximum prefixes allowed 2097152
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was received during read-only mode
 Last ack version 18, Last synced ack version 18
 Outstanding version objects: current 0, max 2
```

```
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Stitching is enabled for this neighbor address-family
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

At the end of each one AFI VPNv4, VPNv6, or L2VPN EVPN, you can see import and advertise information based on the configuration.

```
Router# show bgp sessions

Fri Aug 21 00:25:57.216 PDT

Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router# show bgp vpnv4 unicast

Fri Aug 21 00:28:41.253 PDT
BGP router identifier 30.30.30.30, local AS number 100
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 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
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: 1:1
*> 10.0.0.1/8         32.0.0.2                               0 200 300 i
*> 10.0.0.2/8         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 10.0.0.1/8         32.0.0.2                               0 200 300 i
*> 10.0.0.2/8         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       172.16.0.1                  100        0 i
*>i100.1.1.2/32       172.16.0.1                  100        0 i
*>i200.1.1.1/32       172.16.0.1                  100        0 i
*>i200.1.1.2/32       172.16.0.1                  100        0 i
```

```
Router# show bgp vpnv4 unicast rd 192.168.0.1 10.0.0.1/8 detail

Fri Aug 21 00:28:57.824 PDT
BGP routing table entry for 10.0.0.1/8, Route Distinguisher: 192.168.0.1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
```

```
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT: 1:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1


Router# show bgp vrf foo

Fri Aug 21 00:24:36.523 PDT
BGP VRF foo, state: Active
BGP Route Distinguisher: 192.168.0.1:0
VRF ID: 0x60000002
BGP router identifier 3192.168.0.1, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

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: 30.30.30.30:0 (default for vrf foo)
*> 10.0.0.1/8         172.16.0.1                             0 200 300 i
*> 10.0.0.2/8         172.16.0.1                             0 200 300 i
*>i100.1.1.1/32       172.16.0.1                  100      0 i
*>i100.1.1.2/32       172.16.0.1                   100      0 i
*>i200.1.1.1/32       172.16.0.1                   100      0 i
*>i200.1.1.2/32       172.16.0.1                  100      0 i

Processed 6 prefixes, 6 paths


Router# show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

Mon Dec  8 23:24:50.243 PST
BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
192.168.0.1:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    172.16.0.1 (metric 2) from 20.0.0.1 (172.16.0.1)
      Received Label 1234
```

```
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: RT:1:2
      Originator: 172.16.0.1, Cluster list: 20.20.20.20
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 100:1


Router# show bgp vpnv4 unicast update-group

Fri Aug 21 00:27:57.910 PDT

Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1


Router# show bgp l2vpn evpn update-group

Fri Aug 21 00:27:42.786 PDT

Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
```

```
  First neighbor AS: 100
  Send communities
  Send GSHUT community if originated
  Send extended communities
  4-byte AS capable
  Send AIGP
  Send multicast attributes
  Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 4, replicated: 4
All neighbors are assigned to sub-group(s)
  Neighbors in sub-group: 0.1, Filter-Groups num:1
   Neighbors in filter-group: 0.1(RT num: 0)
    20.0.0.1
```

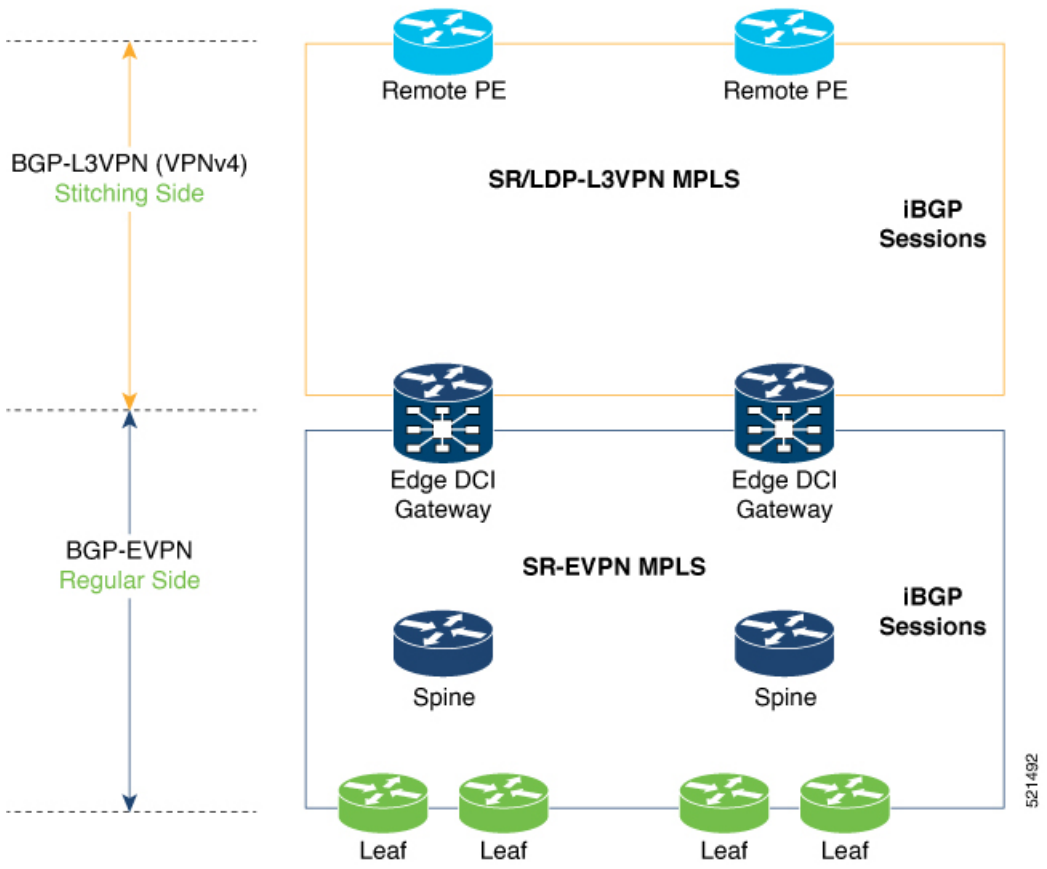
EVPN-Regular RT and VPNv4-Stitching RT

For each VRF on the DCI gateway, there are two sets of manually configured import and export route-targets for EVPN as regular side and VPNv4 as stitching side. Consider the following sets:

- Data Center Route-Targets for EVPN associated with EVPN BGP neighbor (Regular RT)
- MPLS L3VPN Route-Targets for VPNv4 or VPNv6 associated with L3VPN BGP neighbor (Stitching RT)

This separation of RTs enables the two sets of RTs to be independently configured. The RTs associated with the EVPN BGP neighbor does not require the keyword, it remains a normal configuration. The RTs associated with the L3VPN BGP neighbor require **stitching-rt** keyword under VRF configuration.

The following topology shows regular or normal and stitching side.



Route Targets

The RTs associated with the L3VPN BGP neighbor are labelled as stitching RTs. The RTs associated with the EVPN BGP neighbor are normal RTs.

Route Re-Origination

Consider control plane information propagation by the edge DCI gateway from the L3VPN (stitching side) to the Data Center (regular/normal side). Edge DCI gateway advertises to its BGP EVPN neighbor the routes that are re-originated after importing them from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the stitching route-targets with the local route-target values (regular/normal) associated with the BGP EVPN neighbors.

Local VPNv4 or VPNv6 Route Advertisement

On the edge DCI gateway, the locally sourced VPNv4 or VPNv6 routes (any CE directly connected not using L2VPN with BD/EVI/BVI, using only regular L3 VRF) can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets to the BGP EVPN Neighbors (regular/normal side)

VPNv4 neighbors require an additional configuration on the existing legacy VRF to allow these routes to be advertised to VPNv4 RR or remote PEs. Configure **stitching-rt** keyword on existing VRF under import/export RT.

Route Distinguishers

The Router Distinguisher (RD) associated per VRF must be unique per PE in the network. There are few available options to keep unique RD per device:

- Manual configuration: You must manually assign a unique value per device in the network. For example, in this scenario:
  - Leaf (ToR) = RD 1
  - Edge DCI Gateway = RD 2
  - Remote PE = RD 3
- Use **rd auto** command under VRF. To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Note In a DCI deployment, for route re-originate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between edge DCI gateway and MPLS-VPN PE or same RD between edge DCI gateway and Leaf (ToR) is not supported.

Configure EVPN-Regular RT and VPNv4-Stitching RT

This section describes tasks to configure EVPN-Regular RT and VPNv4-Stitching RT. Perform the following tasks to complete the configuration:

- Configure Leaf (ToR)
- Configure Spine-RR (Route Reflector)
- Configure Edge DCI Gateway
- Configure EVPN BGP neighbor and route advertisements
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure Leaf (ToR)

Configure VRF in Leaf (ToR) at BGP-EVPN (regular/normal side). Note that the **stitching-rt** keyword is not required.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:2                              // BGP - EVPN (Regular/Normal Side)
!
```

```
  export route-target
   1:2                            // BGP - EVPN (Regular/Normal Side)
  !
router bgp 100
  neighbor 10.10.1.1              // Spine Loopback IP Address
    address-family l2vpn evpn
       advertise vpnv4 unicast
       advertise vpnv6 unicast
!
```

Note Advertise vpnv4/vpnv6 unicast enables local learned regular L3 VRF prefixes to be advertised as EVPN prefixes to BGP-EVPN neighbor. This means any local prefixes such as PE-CE without L2VPN with BD/EVI/BVI configuration. If all the services are pure EVPN with L2VPN with BD/EVI/BVI configuration these commands are not required.

Configure Spine-RR

Configure Spine RR with Leaf (ToR) and edge DCI gateway as RR client for AFI L2VPN EVPN.

```
// VRF Config is not required //
router bgp 100
  neighbor 10.10.2.1                      // Leaf (ToR) Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
  !
  neighbor 10.10.3.1                      // Edge DCI Gateway  Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
!
```

Configure Edge DCI Gateway

You can configure DCI with the same VRF as Leaf (ToR). Use the same RT as remote PE for L3VPN network or the same VRF if that is possible.

Configure VRF and Route Targets Import and Export rules

Perform the following steps to configure VRF and define route targets to be used for import and export of forwarding information.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:1 stitching                  // BGP – L3VPN (Stitching Side)
   1:2                            // BGP - EVPN (Regular/normal Side)
!
  export route-target
   1:1 stitching                  // BGP – L3VPN (Stitching Side)
   1:2                            // BGP - EVPN (Regular/normal Side)
!
```

Configure EVPN BGP Neighbor and Route Advertisements

Perform this task on the edge DCI gateway to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
router bgp 100
 address-family l2vpn evpn
!
 neighbor 10.10.1.1          // Spine Loopback IP Address
  address-family l2vpn evpn
   import re-originate stitching-rt //Imp EVPN RT 1:2, re-originate VPNv4 1:1
   advertise vpnv4 unicast re-originated //Send routes VPNv4 RT 1:1
!
```

Configure L3VPN BGP Neighbor Relationship and Route Advertisements

Perform the following steps to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
router bgp 100
 address-family vpnv4 unicast
!
 neighbor 10.10.1.1          // Spine Loopback IP Address
  address-family vpnv4 unicast     // Same config for VPNv6
   import stitching-rt re-originate // Imp VPNv4 1:1, reoriginate EVPN 1:2
   advertise vpnv4 unicast re-originated stitching-rt //Send Routes EVPN 1:2
   advertise vpnv6 unicast re-originated stitching-rt //Send Routes EVPN 1:2
!
```

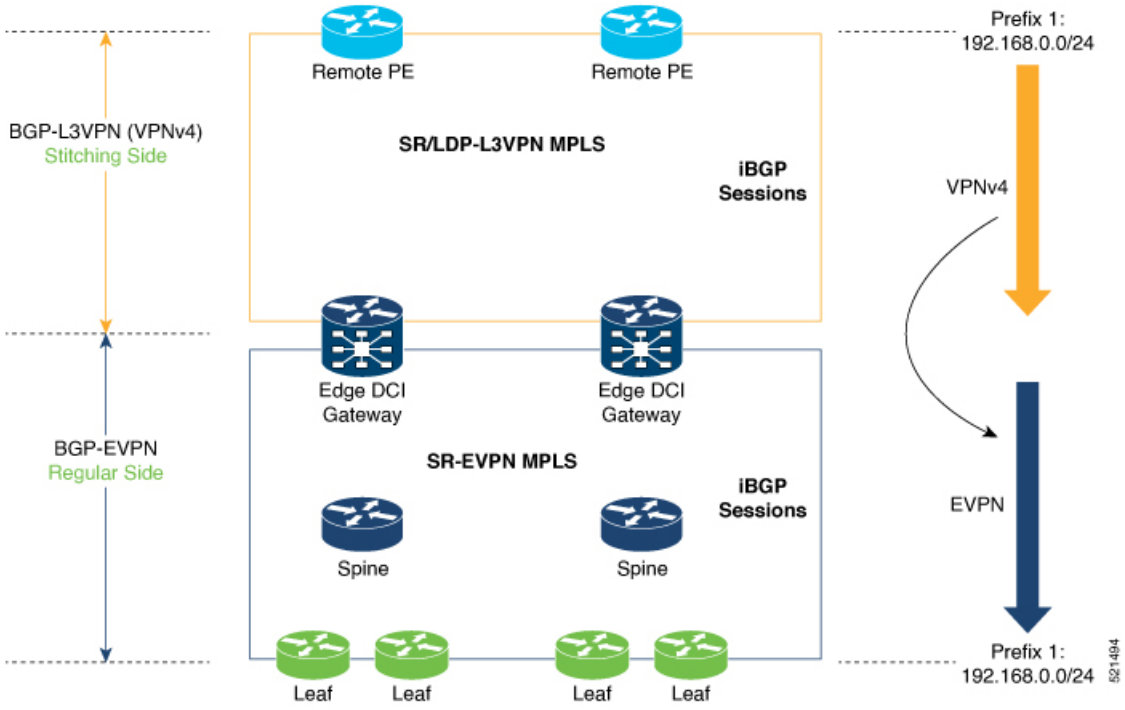
Note The **stitching-rt** applies for L3VPN RT and EVPN RT does not require the **stitching-rt** for this use case.

If there are existing regular local L3 VRF without L2VPN with BD/EVI/BVI in these devices, configure import/export Stitching-RT for existing VRFs to advertise to L3VPN RR or remote PEs.

Configuration applies in two directions:

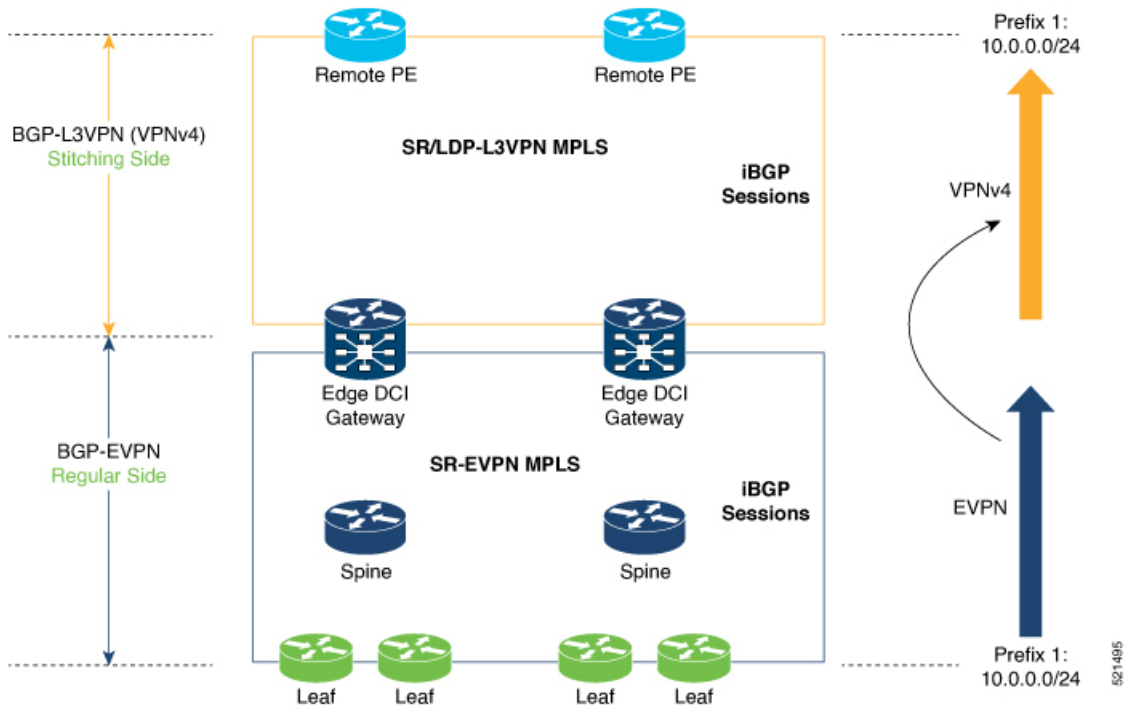
- Stitching from VPNv4 to EVPN routes. Prefixes received from MPLS L3VPN network and re-originated as EVPN prefixes towards Data Center Spine RR and Leaf (ToR)

  1. Importing VPNv4 routes with **import stitching-rt re-originate** command under AFI VPNv4 UNICAST. This command imports routes using RT 1:1 stitching-rt and then re-originate with BGP EVPN 1:2
  2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated** command under AFI L2VPN EVPN. This command advertises routes from MPLS L3VPN network (VPNv4) to BGP EVPN neighbors inside Data Center (Spine RR and then Leaf (ToR)), re-originating these routes using BGP EVPN 1:2.



- Stitching from EVPN to VPNv4 routes. Prefixes received from BGP-EVPN Data Center and re-originated as MPLS L3VPN prefixes towards VPNv4 RR or remote PE in L3VPN network.

  1. Importing EVPN routes with **import re-originate stitching-rt** command under AFI L2VPN EVPN. This command imports routes using RT 1:2 and then re-originate with VPNv4 RT 1:1 **stitching-rt**.
  2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated stitching-rt** command under AFI VPNv4 UNICAST. This command advertises routes from EVPN Data Center to VPNv4 RR or remote PEs, re-originating these routes using VPNv4 RT 1:1 **stitching-rt** .



Verification of Edge DCI Gateway Configuration

```
Router# show bgp l2vpn evpn

Fri Aug 21 00:24:10.773 PDT
BGP router identifier 30.30.30.30, local AS number 100
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 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
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: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
     11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
     11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
        11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
     11.0.0.1                       100      0 i
*>i[5][4231][32][100.1.1.1]/80
     11.0.0.1                       100      0 i
*>i[5][4231][32][100.1.1.2]/80
     11.0.0.1                       100      0 i
*>i[5][4231][112][fec0::1001]/176
     11.0.0.1                       100      0 i
*>i[5][4232][112][fec0::1:1001]/176
```

```
     11.0.0.1                       100      0 i

Processed 8 prefixes, 8 paths
```

```
Router# show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail
```

```
Fri Aug 21 00:34:43.747 PDT
BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate stitching-rt, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
      Extended community: Flags 0x6: RT:1:1
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254
```

The main difference with scenario 1 is that the prefixes have a **reoriginate stitching-rt** keyword on the output versus scenario 1 having just reoriginate.

```
Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail
```

```
Fri Aug 21 00:25:37.383 PDT

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
```

```
  Address family VPNv6 Unicast:   Yes         No
  Address family L2VPN EVPN:      Yes         Yes
Message stats:
  InQ depth: 0, OutQ depth: 0
                 Last_Sent               Sent  Last_Rcvd               Rcvd
  Open:          Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
  Notification:  ---                        0  ---                        0
  Update:        Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
  Keepalive:     Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
  Route_Refresh: Aug 21 00:24:01.377        3  ---                        0
  Total:                                   21                           23
Minimum time between advertisement runs is 0 secs
Inbound message logging enabled, 3 messages buffered
Outbound message logging enabled, 3 messages buffered

For Address Family: VPNv4 Unicast
BGP neighbor version 35
Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
Advertise Reorigination Enabled
Advertise AFI EoR can be sent
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 4, suppressed 0, withdrawn 0
Maximum prefixes allowed 2097152
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was not received during read-only mode
Last ack version 35, Last synced ack version 35
Outstanding version objects: current 0, max 1
Additional-paths operation: None
Send Multicast Attributes

For Address Family: VPNv6 Unicast
BGP neighbor version 29
Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
Advertise Reorigination Enabled
Advertise AFI EoR can be sent
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 1048576
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was not received during read-only mode
Last ack version 29, Last synced ack version 29
Outstanding version objects: current 0, max 0
Additional-paths operation: None
Send Multicast Attributes
Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

For Address Family: L2VPN EVPN
BGP neighbor version 18
Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
Route refresh request: received 0, sent 3
8 accepted prefixes, 8 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 4, suppressed 0, withdrawn 6
Maximum prefixes allowed 2097152
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was received during read-only mode
Last ack version 18, Last synced ack version 18
```

```
  Outstanding version objects: current 0, max 2
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

At the end of each one AFI VPNv4, VPNv6, or L2VPN EVPN, you can see import and advertise information based on the configuration.

Based on whether stitching-side or regular side, import stitching applies on VPNv4 AFI. In Scenario 1 you can see import stitching under L2VPN EVPN.

```
Router# show bgp sessions

Fri Aug 21 00:25:57.216 PDT

Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router# show bgp vpnv4 unicast

Fri Aug 21 00:28:41.253 PDT
BGP router identifier 30.30.30.30, local AS number 100
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 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
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: 1:1
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                    100        0 i
*>i100.1.1.2/32       11.0.0.1                    100        0 i
*>i200.1.1.1/32       11.0.0.1                    100        0 i
*>i200.1.1.2/32       11.0.0.1                    100        0 i
```

In origin IGP line, you can see that the prefix was reoriginated with regular-RT.

```
Router# show bgp vpnv4 unicast rd 30.30.30.30:0 1.1.1.0/24 detail

Fri Aug 21 00:28:57.824 PDT
BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
```

```
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT: 1:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1
```

```
Router# show bgp vrf foo

Fri Aug 21 00:24:36.523 PDT
BGP VRF foo, state: Active
BGP Route Distinguisher: 30.30.30.30:0
VRF ID: 0x60000002
BGP router identifier 30.30.30.30, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

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: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                 100           0 i
*>i100.1.1.2/32       11.0.0.1                 100           0 i
*>i200.1.1.1/32       11.0.0.1                 100           0 i
*>i200.1.1.2/32       11.0.0.1                 100           0 i

Processed 6 prefixes, 6 paths
```

```
Router# show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

Mon Dec  8 23:24:50.243 PST
BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 1234
```

```
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: RT:1:2
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 100:1v
```

```
Router# show bgp vpnv4 unicast update-group

Fri Aug 21 00:27:57.910 PDT

Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

```
Router# show bgp l2vpn evpn update-group

Fri Aug 21 00:27:42.786 PDT

Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
```

```
  Send GSHUT community if originated
  Send extended communities
  4-byte AS capable
  Send AIGP
  Send multicast attributes
  Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 4, replicated: 4
All neighbors are assigned to sub-group(s)
  Neighbors in sub-group: 0.1, Filter-Groups num:1
   Neighbors in filter-group: 0.1(RT num: 0)
    20.0.0.1
```

EVPN Default VRF Route Leaking

The EVPN Default VRF Route Leaking feature leak routes between EVPN address-family and IPv4/IPv6 unicast address-family (Default-VRF), enabling the data center hosts to access the Internet. This feature is an extension of Border Gateway Protocol (BGP) VRF Dynamic route leaking feature that provides connectivity between non-default VRF hosts and Default VRF hosts by exchanging routes between the non-default VRF and Default VRF. EVPN Default VRF Route Leaking feature extends the BGP VRF Dynamic leaking feature, by allowing EVPN/L3VPN hosts to communicate with Default VRF hosts.

The import process installs the Internet route in a VRF table or a VRF route in the Internet table, providing connectivity.

The BGP VRF Dynamic route leaking feature is enabled by:

- Importing from default-VRF to non-default-VRF using the following command in VRF address-family configuration mode.

  import from default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

  If the **advertise-as-vpn** keyword is used, the paths imported from the default-VRF to the non-default-VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.

  The EVPN Default VRF Route Leaking feature with **advertise-as-vpn** keyword, enables to advertise the paths imported from default-VRF to non-default VRFs to EVPN PE peers as well.

  A new command **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of Default-VRF leaked routes to that neighbor.

- Importing from non-default-VRF to default-VRF using the following command in VRF address-family configuration mode.

  export to default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

  The Dynamic Route Leaking feature enables leaking of local and CE routes to Default-VRF.

  A new optional keyword **allow-imported-vpn** is added to the above command, when configured, enables the leaking of EVPN and L3VPN imported/re-originated routes to the Default-VRF.

A route-policy is mandatory to filter the imported routes. This reduces the risk of unintended import of routes between the Internet table and the VRF tables and the corresponding security issues. There is no hard limit

on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note Each VRF importing global routes adds workload equivalent to a neighbor receiving the global table. This is true even if the user filters out all but a few prefixes.

Scale Limitation of Default Route Leaking

Default VRF route leaking uses Dynamic Route Leaking feature to leak prefixes between the default VRF and the DC VRF. Do not use Dynamic Route Leaking feature to leak default VRF prefixes to large number of DC VRFs, even if you filter out all prefixes except a few that are to be leaked.

The following are the key factors that affect the performance:

- The default VRF prefix scale, which is approximately 0.7 million internet prefixes.
- The number of DC VRFs the default VRF prefixes that are to be imported.

To improve the scale, either the prefix scale or the number of VRFs whose prefixes that are to be imported must be reduced.

To manage the scale limitation, Cisco recommends you to do the following:

- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI, and advertise a default route towards the DCI. On the DCI, import the default route from default VRF to DC VRFs.
- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI. On the DCI, configure a static default route in the DC VRF with the next hop of the default VRF pointing to the adjacent PE address.
- Configure the static default route 0.0.0.0/0 on DC VRF with nexthop as “vrf default”.

Note If the static routes are re-distributed to BGP, make sure it is not unintentionally advertised out.

EVPN Default VRF Route Leaking on the DCI for Internet Connectivity

The EVPN Default VRF Route Leaking feature leak routes between the Default-VRF and Data Center-VRF on the DCI to provide Internet access to data center hosts.

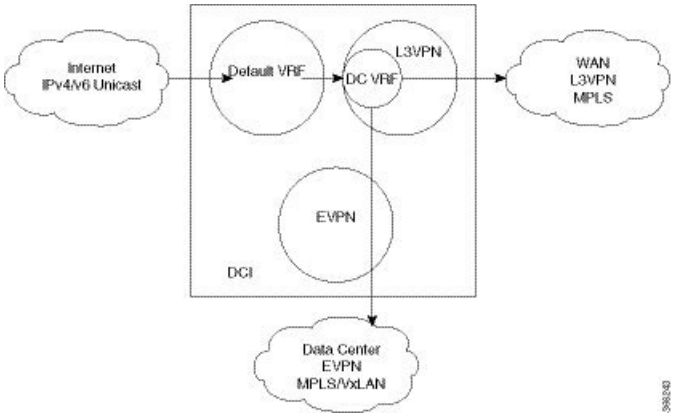
This feature is enabled by:

- Leaking routes from Default-VRF to Data Center-VRF
- Leaking routes to Default-VRF from Data Center-VRF

Leaking Routes from Default-VRF to Data Center-VRF

This section explains the process of leaking Default-VRF routes to Data Center-VRF.

Figure 1: Leaking Routes from Default-VRF to Data Center-VRF



Step 1 The Internet routes are present in the Default-VRF on the DCI.

Note A static default-route (0/0) can be configured under Default-VRF router static address-family configuration and redistributed to BGP.

Step 2 A route-policy is configured to select the routes to be leaked from Default-VRF to Data Center-VRF.

Example:

```
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
```

Note Instead of leaking the internet routes, you can leak the default-route 0/0 from Default-VRF to Data Center-VRF using the following policy.

```
route-policy import-from-default-policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (0::0/0) then
    pass
  endif
end-policy
!
```

Step 3 Leak Default-VRF routes specified in the route-policy to Data Center-VRF by configuring **import from default-vrf route-policy import-from-default-policy(-v6)** under Data Center VRF address-family configuration mode.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6
!
```

Step 4 Advertise the leaked (Default-VRF) routes in the Data Center-VRF as EVPN routes towards Data Center routers by configuring **advertise-as-vpn** option.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy advertise-as-vpn
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
!
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Default-VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When default-originate is configured under the neighbor address-family for EVPN/L3VPN, there is no need to advertise the Default-VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Default-VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** command under respective neighbor address-family.

Example:

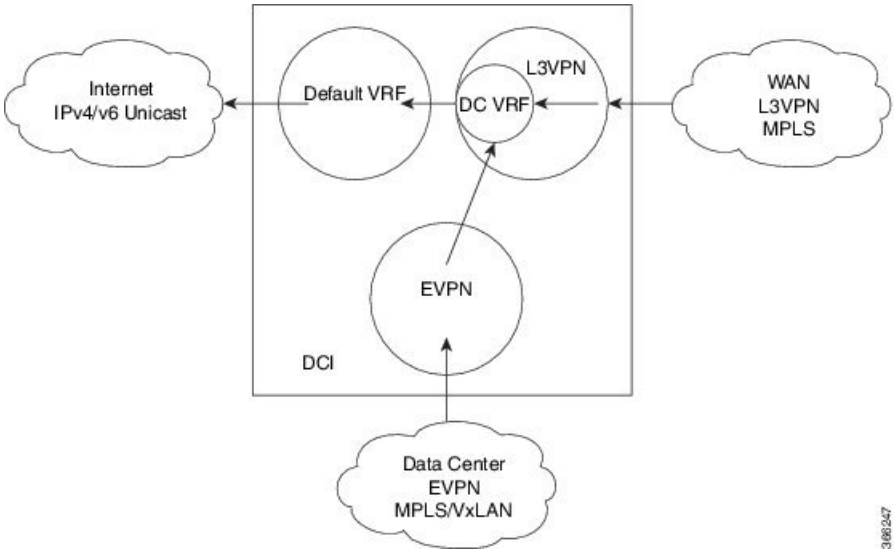
```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
```

```
      advertise vpnv4 unicast imported-from-default-vrf disable
      advertise vpnv6 unicast imported-from-default-vrf disable
!
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      advertise vpnv4 unicast imported-from-default-vrf disable
    address-family vpnv6 unicast
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Leaking Routes to Default-VRF from Data Center-VRF

This section explains the process of leaking Data Center-VRF routes to Default-VRF.

Figure 2: Leaking Routes to Default-VRF from Data Center-VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center-VRF to Default-VRF.

Example:

```
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
```

Step 3 Leak Data Center-VRF routes specified in the above policy to Default-VRF by configuring **export to default-vrf route-policy export-to-default-policy(-v6) [allow-imported-vpn]** under Data Center-VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Default-VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Default-VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    export to default-vrf route-policy export-to-default-policy [allow-imported-vpn]
  !
  address-family ipv6 unicast
    export to default-vrf route-policy export-to-default-policy-v6 [allow-imported-vpn]
  !
```

Step 4 The Leaked routes in the Default VRF are advertised to the Internet.

Note Instead of advertising the leaked routes to the Internet, an aggregate can be configured and advertised to the Internet.

Sample Router Configuration

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured on a DCI router to provide Internet access to the data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
  address-family ipv6 unicast
    import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !

route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
```

```
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured along with default-originate on a DCI router to provide Internet access to data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
  address-family ipv6 unicast
    import from default-vrf route-policy import-from-default-policy-v6 <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!
route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!
```

```
route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
      default-originate <= Added=>

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate <= Added=>
  !
  address-family ipv6 unicast
    allow vpn default-originate <= Added=>
```

EVPN Service VRF Route Leaking

The EVPN Service VRF Route Leaking feature enables connectivity to the services in the Service VRF to customers in EVPN Data Center VRF. The Service VRF and Data Center VRF routes can be IPv4 and/or IPv6 addresses. The Services VRF is any L3 VRF providing services reachable through connected, static, re-distributed IGP or BGP routes.

This feature leaks routes between Data Center VRF and Service VRF, enabling the EVPN/L3VPN hosts to access the Services in the Service VRF. This feature rely on Border Gateway Protocol (BGP) VRF extranet feature that imports routes between two VRFs.

The import process installs the Data Center VRF routes in a Service VRF table or a Service VRF routes in the Data Center VRF table, providing connectivity.

The BGP Service VRF route leaking feature is enabled by:

- Importing routes from Service VRF to Data Center VRF and advertising it as EVPN/L3VPN route from Data Center VRF.
    - Importing Service VRF routes to Data Center VRF by attaching Data Center VRF import RTs to Service VRF routes.

      This can be achieved by configuring one or more Data Center VRF import RTs as export RT of Service VRF, or configuring a Service VRF export route-policy to attach import RT EXTCOMM to Service VRF routes matching the import RTs of Data Center VRF using the following command in Service VRF address-family configuration mode.

      export route-policy service-vrf-export-route-policy-name

Where the route-policy "service-vrf-export-route-policy-name" attaches the RT EXTCOMM matching the one or more import RTs of Data Center VRF to Service VRF routes.

- Advertising Data Center VRF imported routes that are exported from Service VRFs as EVPN/L3VPN NLRI from Data Center VRF using the following command in Data Center VRF address-family configuration mode.

  import from vrf advertise-as-vpn

  If the **advertise-as-vpn** keyword is used, the paths imported from the Service VRF to the Data Center VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the Service VRF to the Data Center VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.

- Block advertising Data Center VRF leaked routes from being advertised to a neighbor using the following command in neighbor address-family configuration mode.

  advertise vpnv4/vpnv6 unicast imported-from-vrf disable

  A new command **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of VRF to VRF leaked routes to that neighbor.

- Importing EVPN/L3VPN routes from Data Center VRF to Service VRF
  - Importing EVPN/L3VPN routes from Data Center VRF to Service VRF by attaching Service VRF import RTs.

    This can be achieved by configuring one or more Service VRF import RTs as export RT of Data Center VRF, or configuring a Data Center VRF export route-policy to attach import RT EXTCOMM to Data Center VRF routes matching the import RTs of Service VRF using the following command in Data Center VRF address-family configuration mode.

    export route-policy data-center-vrf-export-route-policy-name

    The route-policy "data-center-vrf-export-route-policy-name" attaches the RT EXTCOMM matching one or more import RTs of Service VRF.

  - Allow leaking of Data Center VRF routes to Service VRF by using the following command in Data Center VRF address-family configuration mode.

    export to vrf allow-imported-vpn

Note In order to prevent un-intended import of routes to VRFs, select unique RT's to import routes between Service VRF and Data Center VRF, which are not used for normal import of VPN/EVPN routes to Data Center VRFs.

The Extranet Route Leaking feature enables leaking of local and CE routes from one VRF to another VRF. A new command **export to vrf allow-imported-vpn** is added to enable the leaking of EVPN and L3VPN imported/re-originated Data Center VRF routes to the Service VRF.

Note A route-policy is preferred to filter the imported routes. This reduces the risk of unintended import of routes between the Data Center VRF and the Service VRF, and the corresponding security issues. There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note This feature does not advertise EVPN/L3VPN PE routes imported to Data Center VRF and leaked to Service VRF as EVPN/L3VPN PE route.

EVPN Service VRF Route Leaking on the DCI for Service Connectivity

The EVPN Service VRF Route Leaking feature leaks routes between the Service VRF and Data Center VRF on the DCI to provide access to Services to data center hosts.

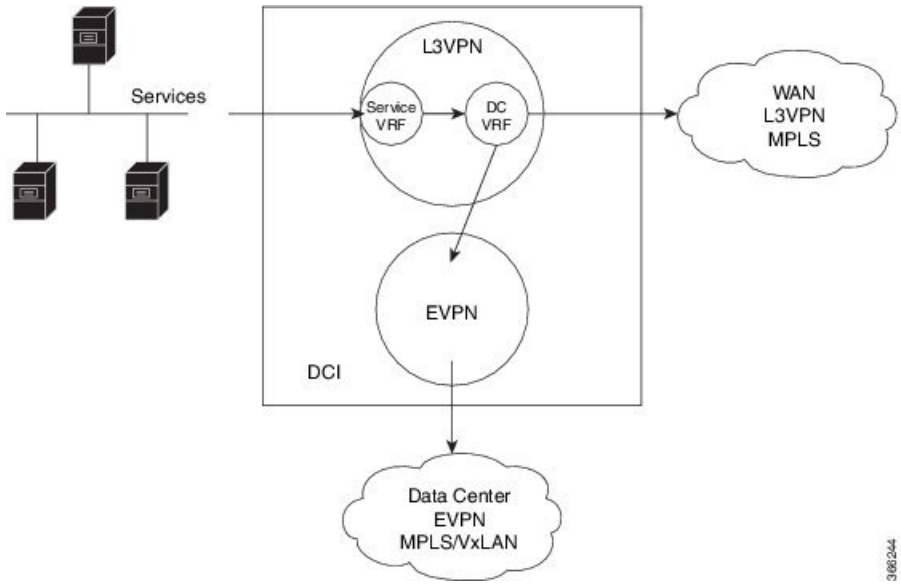
This feature is enabled by:

- Leaking routes from Service VRF to Data Center VRF
- Leaking routes to Service VRF from Data Center VRF

Leaking Routes from Service VRF to Data Center VRF

This section explains the process of leaking Service VRF routes to Data Center VRF.

Figure 3: Leaking Routes from Service VRF to Data Center VRF



Step 1 The Service routes are present in the Service VRF on the DCI.

Step 2 A route-policy is configured to select the routes to be leaked from Service VRF to Data Center VRF.

Example:

```
route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
```

Step 3 Leak Service VRF routes specified in the route-policy to Data Center VRF by configuring **export route-policy service-vrf-export-policy(-v6)** under Service VRF address-family configuration mode.

Example:

```
vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !
```

Step 4 Advertise the leaked (Service VRF) routes in the Data Center VRF as EVPN/L3VPN routes towards Data Center routers by configuring **import from vrf advertise-as-vpn** under Data Center VRF address-family configuration mode..

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
```

```
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Service VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When **allow vpn default-originate** is configured under the Data Center VRF, there is no need to advertise the Service VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Service VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** command under respective neighbor address-family.

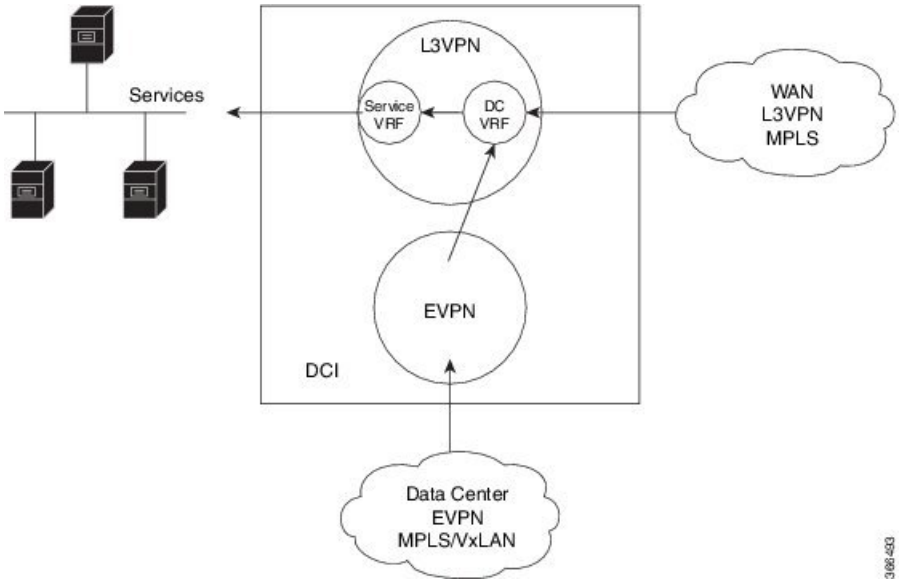
Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
    !
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Leaking Routes to Service VRF from Data Center VRF

This section explains the process of leaking Data Center VRF routes to Service VRF.

Figure 4: Leaking Routes to Service VRF from Data Center VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center VRF to Service VRF.

The policy attaches RT EXTCOMM to Data Center VRF routes matching one or more import RT of the Service VRF.

Example:

```
route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  if destination in (200.168.0.0/16) then <--- VPNv4 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  elseif destination in (200:168::0/64) then <--- VPNv6 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
```

Note An EVPN/L3VPN route received from a neighbor configured locally with "import stitching-rt re-originate" is imported to Data Center VRF if the route's RT EXTCOMM matches with one or more Data Center VRF import stitching RTs, and is leaked to Service VRF if the Data Center VRF route's RT EXTCOMM matches with one or more Service VRF import stitching RTs.

Step 3 Leak Data Center VRF routes specified in the above policy to Service VRF by configuring **export route-policy data-center-vrf-export-policy(-v6)** under Data Center VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Service VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Service VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
```

Step 4 The Data Center VRF leaked routes in the Service VRF are advertised to Service VRF CE peers.

Sample Router Configuration

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured on a DCI router providing access to data center hosts to Services in the Service VRF.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
```

```
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16)
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64)
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!
```

```
router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured along with default-originate on a DCI router to provide data center hosts access to Services in the Service VRF..

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
```

```
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
      default-originate <= Added=>
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
```

```
    route-policy pass-all in
    route-policy pass-all out
    advertise vpnv4 unicast re-originated
    advertise vpnv4 unicast imported-from-vrf disable
    default-originate <= Added=>
  address-family vpnv6 unicast
    import re-originate stitching-rt
    route-policy pass-all in
    route-policy pass-all out
    advertise vpnv6 unicast re-originated
    advertise vpnv6 unicast imported-from-vrf disable
    default-originate <= Added=>

vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate <= Added=>
  !
  address-family ipv6 unicast
    allow vpn default-originate <= Added=>
```

Data Center Interconnect between MPLS-VPN and EVPN-VxLAN

This part provides conceptual and configuration information for Data Center Interconnect (DCI) VXLAN Layer 3 Gateway on the router.

Release	Modification
Release 5.3.2	This feature was introduced.
Release 6.1.x	OpFlex
Release 6.6.x	EVPN VxLAN VRF leaking

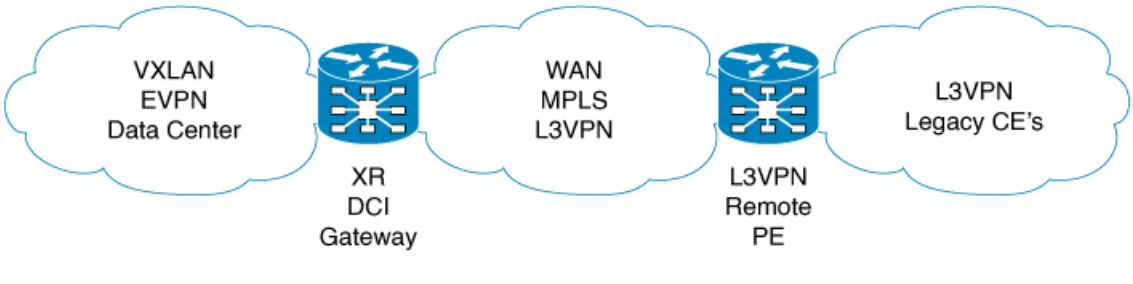
Data Center Interconnect VXLAN Layer 3 Gateway

Router can serve as a Data Center Interconnect (DCI) L3 Gateway using stitching technology between VPNv4/v6 and EVPN-VXLAN. The DCI provides a solution for a new EVPN-VXLAN Data Center that needs to communicate with legacy and existing traditional MPLS VPN networks (VPNv4) having PE-CE architecture.

The DCI L3 gateway provides the following functions:

- **IP connectivity between multi-tenant remote Data Center sites**: Consider the following network topology that has two Data Center sites connected through the intermediate service provider network. The multi-tenant Data Centers use VXLAN encapsulation to carry separate tenant IP traffic. The VXLAN-enabled Data Center sites use the MP-BGP EVPN control plane for distributing both Layer-2 and Layer-3 forwarding information within the site. The router uses MPLS L3VPN application service over the service provider network to provide L3 connectivity between the two Data Center sites. Making this translation between EVPN-VXLAN to VPNv4 overlay.

- **IP Connectivity between Data Center and remote PEs in a legacy network**: Consider the following network topology that has one new Data Center site connected through the intermediate service provider network. The multi-tenant Data Center uses VXLAN encapsulation to carry separate tenant IP traffic. The VXLAN-enabled Data Center site uses the MP-BGP EVPN control plane for distributing both Layer-2 and Layer-3 forwarding information within the site. The router uses MPLS L3VPN application service over the service provider network to provide L3 connectivity between the Data Center services and the legacy CEs using VPNv4 to communicate with services placed inside the Data Center. Making this translation between EVPN-VXLAN to VPNv4 overlay.



Note

- DCI gateway does not provide layer 2 inter-connectivity across Data Centers.
- In a DCI deployment, for route reoriginate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between DCI and MPLS-VPN PE or same RD between DCI and VxLAN Top of Rack (ToR) is not supported.

Route Targets

For each VRF on the DCI router, there are two sets of manually configured import and export route-targets. One set of import and export route-targets is associated with the Data Center BGP neighbor that uses EVPN address-family to exchange L3 information; the other set of import and export route-targets is associated with the L3VPN BGP neighbor that use VPNv4 or VPNv6 unicast address-family to exchange L3 information. This separation of route targets (RTs) enables the two sets of RTs to be independently configured. The DCI router effectively stitches the two set of RTs. The RTs associated with the EVPN BGP neighbor are labelled as stitching RTs. The RTs associated with the L3VPN BGP neighbor are normal RTs.

Route Re-origination

Consider the case of control plane information propagation by the DCI from the L3VPN side to the Data Center side. Here, instead of advertising the remote Data Center's original BGP EVPN routes, you can configure the DCI router to advertise to its BGP EVPN neighbor the routes that are re-originated after importing them

from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the normal route-targets with the local route-target values associated with the BGP EVPN neighbors . The converse holds true for the routing information traffic propagation from the BGP EVPN control plane to BGP L3VPN control plane. You can configure this re-origination by using the re-originate keyword in the import re-originate command.. Configuring this command, by default, also enables advertisement of L2VPN EVPN prefixes to the EVPN BGP neighbors. You can suppress native L2VPN EVPN address-family NLRI advertisements towards the EVPN Neighbor using the advertise l2vpn evpn disable command under the EVPN BGP address-family configuration mode.

Route Address-Family and Encoded Address-Family

When an address-family is configured for a BGP neighbor, it means that the specified address-family routes encoded with the NLRI for that address-family is advertised to the neighbor. This does not hold for data center BGP neighbors because they use only EVPN address-family. Here, BGP neighbors advertise VPNv4 or VPNv6 unicast routes using the EVPN NLRI encoding. Thus, here the encoded address-family and route address family can be possibly different. You can advertise the VPNv4 or VPNv6 address-family using the advertise vpnv4 unicast or advertise vpnv6 unicast command. For example, a EVPN address-family BGP neighbor configured with the advertise vpnv4 unicast command sends VPNv4 unicast routes in an EVPN encoded NLRI.

Local VPNv4 or VPNv6 Routes Advertisement

On the DCI router, the locally sourced VPNv4 or VPNv6 routes can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets. You can configure these local VPNv4 or VPNv6 route advertisements to be advertised with stitching RTs to the BGP EVPN neighbors by using the advertise vpnv4 unicast local stitching-rt or advertise vpnv6 unicast local stitching-rt command as required.

Data Center VXLAN with Support for MP-BGP

The Data Center VXLAN uses MP-BGP for control-plane learning of end-host Layer 2 and Layer 3 reachability information. The DCI router is configured with a VXLAN Tunnel EndPoint (VTEP). For VTEP configuration details, see the chapter Implementing Layer 3 VXLAN Gateway. You also need to run the host-reachabilty protocol bgp command to specify that control-plane learning within Data center site is through BGP routing protocol.

The DCI Gateway router and the EVPN BGP neighbor (Data Center BGP neighbor) exchange BGP EVPN NLRIs of route type 5 that carry L3 routing information and associated VXLAN encapsulation information. Some of the VXLAN information is carried in the EVPN NLRI and the rest is carried in RFC 5512 Tunnel Type Encapsulation EXTCOMM and Router MAC EXTCOMM defined in draft-ietf-bess-evpn-inter-subnet-forwarding-00. BGP downloads VXLAN encapsulation as RIB remote next hop opaque attribute to L3RIB.

Default-Originate Forwarding to BGP EVPN Neighbor

Instead of advertising the specific networks available in the remote Data Center, you can configure the DCI gateway to advertise a default route to the directly connected Data Center neighbor. To send the default route for a VRF instance to the Data Center BGP EVPN neighbor, the VPN default-originate information that is typically forwarded to the L3VPN BGP neighbor, is also configured to be forwarded to the BGP EVPN neighbor in the Data Center. To do so, you need to configure allow vpn default-originate command in the BGP VRF configuration mode and also configure default-originate command under EVPN BGP neighbor in L2VPN EVPN address-family configuration mode. This configures BGP to forward only one default route

information for a VRF instance from the DCI Gateway to the BGP neighbor that has L2VPN EVPN address-family. This default route information is encoded in the EVPN "IP Prefix Route" NLRI.

With the advertisement of a default route to the connected Data Center, the DCI Gateway should not advertise specific prefixes of the remote Data Center to the BGP EVPN neighbor. To prevent forwarding of VRF prefixes, you need to configure the DCI gateway with a EVPN BGP neighbor policy that drops forwarding of all prefixes.

Configure Data Center Interconnect Router

Perform the following steps tto configure the Data Center Interconnect (DCI) router:

- Configure VRF and route targets import/export rules
- Configure Bridge Domain for DCI Gateway
- Configure VTEP.
- Configure EVPN BGP neighbor and route advertisement
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure VRF and route targets import/export rules

Perform the following to configure VRF and define route targets to be used for import and export of forwarding information.

```
Router# configure
Router(config)# vrf data-center-10
Router(config-vrf-af)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target 1:1
Router(config-vrf-af)# export route-target 1:2
Router(config-vrf-af)# import route-target 10:1 stitching
Router(config-vrf-af)# export route-target 10:2 stitching
Router(config-vrf-af)# commit
```

Configure Bridge Domain for DCI Gateway

Perform the following to configure the bridge domain on the DCI Gateway.

Note For DCI VxLAN L3 Gateway, only routed interface BVI and member vni can be configured in the bridge-domain. All other L2 services such as EVI, PW, or AC are not supported in the bridge-domain.

```
Router# configure
Router(config)# interface bvi 1
Router(config-if)# vrf cust1
Router(config-if)# ipv4 address 40.1.1.1 255.255.255.254
Router(config)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd)# member vni 5001
Router(config-l2vpn-bg-bd)# commit
```

Configure VTEP (VxLAN Terminal EndPoint) on the DCI Gateway.

Perform the following to configure VTEP (VxLAN Terminal EndPoint) on the DCI Gateway.

```
Router# configure
Router(config)# interface loopback 0
Router(config-if)# ipv4 address 40.1.1.1 255.255.255.255
Router(config)# exit
Router(config)# interface nve 1
Router(config-if)# source interface loopback 0
Router(config-if)# member vni 5001
Router(config-nve-vni)# vrf cust1
Router(config-nve-vni)# host reachability protocol bgp
Router(config-nve-vni)# commit
```

Configure EVPN BGP neighbor and route advertisements

Perform the following on the DCI router to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr)# default-originate /*optional configuration*/
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
Router(config-bgp-nbr-af)# advertise l2vpn evpn disable/*optional configuration*/
Router(config-bgp-nbr-af)# commit
```

Configure L3VPN BGP neighbor relationship and route advertisements

Perform the following to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp-nbr)# address-family vpnv4
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family vpnv4
Router(config-bgp-nbr-af)# import reoriginate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated


Router# configure
Router(config)# router bgp 100
Router(config-bgp-nbr)# address-family vpnv6
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family vpnv6
Router(config-bgp-nbr-af)# import reoriginate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
```

Verification

You can use the folllowing show commands to verify the DCI Gateway configurations:

```
Router# show bgp l2vpn evpn

BGP router identifier 30.30.30.30, local AS number 100
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 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
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: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
                      11.0.0.1                 100      0 i
*>i[5][4231][32][100.1.1.1]/80
                      11.0.0.1                 100      0 i
*>i[5][4231][32][100.1.1.2]/80
                      11.0.0.1                 100      0 i
*>i[5][4231][112][fec0::1001]/176
                      11.0.0.1                 100      0 i
*>i[5][4232][112][fec0::1:1001]/176
                      11.0.0.1                 100      0 i

Processed 8 prefixes, 8 paths


Router#  show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail

BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
     Extended community: Flags 0x2: Encapsulation Type:8 Router MAC:aabb.ccdd.eeff RT:65540:1
 RT:40.40.40.40:1 RT:100:1
```

```
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254

Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
    Address family VPNv6 Unicast:   Yes         No
    Address family L2VPN EVPN:      Yes         Yes
  Message stats:
    InQ depth: 0, OutQ depth: 0
                    Last_Sent               Sent  Last_Rcvd               Rcvd
    Open:           Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
    Notification:   ---                        0  ---                        0
    Update:         Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
    Keepalive:      Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
    Route_Refresh:  Aug 21 00:24:01.377        3  ---                        0
    Total:                                    21                           23
  Minimum time between advertisement runs is 0 secs
  Inbound message logging enabled, 3 messages buffered
  Outbound message logging enabled, 3 messages buffered

 For Address Family: VPNv4 Unicast
  BGP neighbor version 35
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 0
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 35, Last synced ack version 35
```

```
  Outstanding version objects: current 0, max 1
  Additional-paths operation: None
  Send Multicast Attributes

 For Address Family: VPNv6 Unicast
  BGP neighbor version 29
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 0, suppressed 0, withdrawn 0
  Maximum prefixes allowed 1048576
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 29, Last synced ack version 29
  Outstanding version objects: current 0, max 0
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

 For Address Family: L2VPN EVPN
  BGP neighbor version 18
  Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
  Route refresh request: received 0, sent 3
  8 accepted prefixes, 8 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 6
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was received during read-only mode
  Last ack version 18, Last synced ack version 18
  Outstanding version objects: current 0, max 2
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Stitching is enabled for this neighbor address-family
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

```
Router# show bgp sessions


Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router#  show bgp vpnv4 unicast
BGP router identifier 30.30.30.30, local AS number 100
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 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
```

```
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: 1:1
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                 100           0 i
*>i100.1.1.2/32       11.0.0.1                 100           0 i
*>i200.1.1.1/32       11.0.0.1                 100           0 i
*>i200.1.1.2/32       11.0.0.1                 100           0 i


Router# show bgp vpnv4 unicast rd 30.30.30.30:0 1.1.1.0/24 detail

BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT:100:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1


Router# show bgp vrf foo


BGP VRF foo, state: Active
BGP Route Distinguisher: 30.30.30.30:0
VRF ID: 0x60000002
BGP router identifier 30.30.30.30, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

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: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                 100           0 i
```

```
*>i100.1.1.2/32       11.0.0.1                 100      0 i
*>i200.1.1.1/32       11.0.0.1                 100      0 i
*>i200.1.1.2/32       11.0.0.1                 100      0 i

Processed 6 prefixes, 6 paths


Router#  show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 1234
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: Encapsulation Type:8 Router MAC:aabb.ccdd.eeff RT:1:2
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      RIB RNH: table_id 0xe0000011, Encap 8, VNI 1234, MAC Address: aabb.ccdd.eeff, IP
Address: 11.0.0.1, IP table_id 0xe0000000
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route
Distinguisher: 100:1

Router# show bgp vpnv4 unicast update-group



Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
```

```
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

```
Router# show bgp l2vpn evpn update-group
```

```
Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 4, replicated: 4
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

Example for configuring Data Center Interconnection Layer 3 Gateway

The following configurations provide an example Data Center Interconnection (DCI) Layer 3 Gateway configuration.

VTEP-related configuration:

```
interface Loopback1
 ipv4 address 40.1.1.1 255.255.255.255
!

interface nve1
 source-interface Loopback1
 member vni 1
  vrf cust1
  host-reachabilty protocol bgp
 !
```

```
interface BVI1
 vrf cust1
 ipv4 address 10.99.1.30 255.255.255.0
 ipv6 address 10:99:1::30/64
!

l2vpn
 bridge group bg1
  bridge-domain bd1
   routed interface BVI1
   member vni 1
  !
 !
```

VRF-related configuration

```
vrf data-center-10
  import route-target 1:1
  export route-target 1:2
  import route-target 10:10 stitching
  export route-target 10:20 stitching
```

Data Center EVPN BGP neighbor-related configuration

```
router bgp 1
 neighbor 1.1.1.1
   address-family l2vpn evpn
     import stitching-rt reoriginate
     advertise vpnv4 unicast reoriginated
     advertise vpnv6 unicast reoriginated
     advertise vpnv4 unicast local stitching-rt
     advertise vpnv6 unicast local stitching-rt
     advertise l2vpn evpn disable
```

L3VPN BGP neighbor-related configuration

```
router bgp 2
neighbor 10.10.10.10
  address-family vpnv4
    import reoriginate stitching-rt
    advertise vpnv4 unicast reoriginated
```

The following example configuration shows how to configure the DCI router to forward default route to its Data Center neighbor.

```
router bgp 1
 address-family vpnv4 unicast
 address-family vpnv6 unicast
 address-family l2vpn evpn
 exit
 neighbor 1.1.1.1
  address-family l2vpn evpn
  default-originate
  exit
 vrf foo
  rd 2:1
  address-family ipv4 unicast
  allow vpn default-originate
  exit
 address-family ipv6 unicast
 allow vpn default-originate
 exit
exit
!
```

EVPN VxLAN VRF Route Leaking

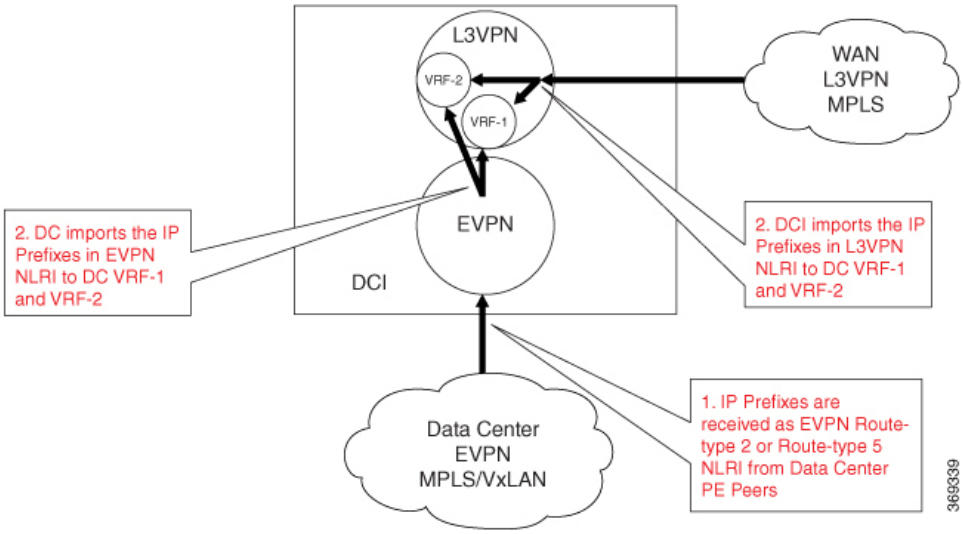
The EVPN VxLAN VRF Route Leaking feature enables you to import IP prefixes to more than one VRF and leaks IP prefixes from source VRF to destination VRF. This feature supports the following functionalities:

- Import L3VPN and EVPN IP Prefixes to more than one VRF, on page 56
- Extranet Route Leaking, on page 56
- Dynamic Route Leaking, on page 56
- Leak L3VPN and EVPN Imported IP Prefixes to another VRF, on page 57
- Advertise Leaked Prefix, on page 60
- Advertise Leaked Prefix Back to Originator, on page 65
- Lookup in Source VRF, on page 68

Import L3VPN and EVPN IP Prefixes to more than one VRF

This functionality allows you to import L3VPN and EVPN IP prefixes to more than one VRF when the RT EXTCOMM in the prefix matches import RT of more than one VRF. Starting from Cisco IOS XR Software Release 6.6.2, this functionality is supported for prefixes received with VxLAN tunnel attributes.

Figure 5: Import IP Prefixes to DC VRF-1 and DC VRF-2



Extranet Route Leaking

This functionality allows you to attach the RT EXTCOMM to the prefix that matches the import RT of another VRF. This enables you to leak redistributed or CE VRF IP prefix to another VRF. It is an existing L3VPN functionality to leak redistributed and CE prefixes between VRFs.

Dynamic Route Leaking

This functionality enables you to leak routes between default-VRF and L3VPN VRFs.

For more information about extranet route leaking, see the *Implementing BGP* chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Leak L3VPN and EVPN Imported IP Prefixes to another VRF

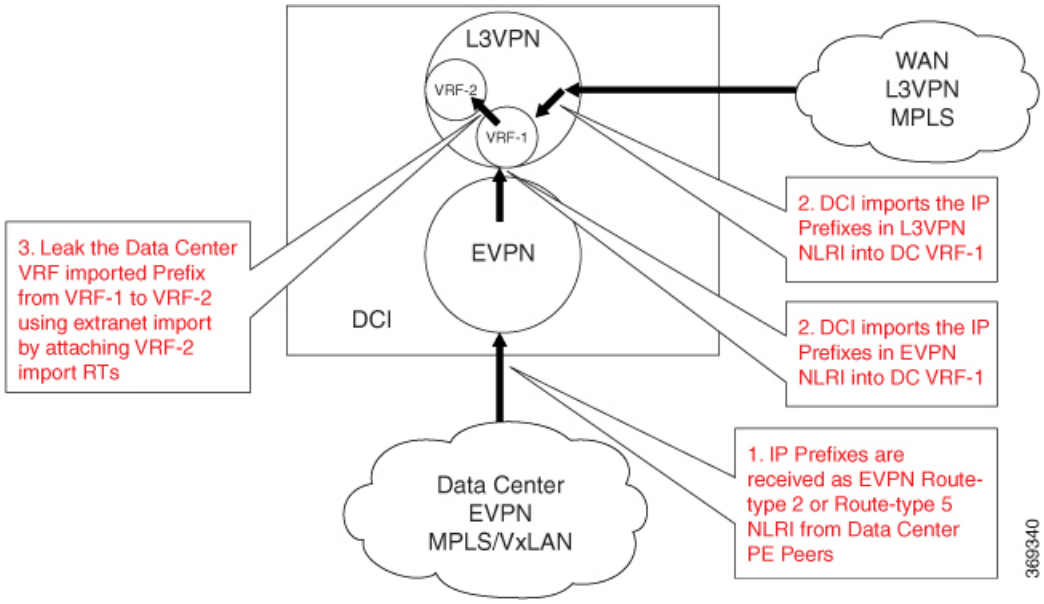
This functionality allows you to attach the RT EXTCOMM to the imported prefix that matches the import RT of another VRF. This enables you to leak L3VPN and EVPN IP prefixes to another VRF.

Configure the following under global VRF address-family mode to enable this functionality:

- Configure the VRF export route-policy in source VRF to attach the import RT of the destination VRF.
- Use the **export to vrf allow-imported-vpn** command under source VRF to enable the leaking of imported prefix from source VRF to destination VRF.

Note Starting from Cisco IOS XR Software Release 6.6.2, this functionality allows you to leak imported L3VPN and EVPN IP prefixes with VxLAN tunnel attributes.

Figure 6: Leak from DC VRF-1 to DC VRF-2



Configuration Example

In this example, the IP address (203.0.113.1/32, 2001:DB8::1/128) in EVPN route-type 2 and prefix (203.0.113.0/24, 2001:DB8::/32) in route-type 5 are imported to VRF-1 and leaked to VRF-2. The RT EXTCOMM (100:1) in NLRI matches one or more import stitching-RTs of VRF-1, and VRF-1 export policy which matches the NLRI IP address and prefix attaches the import stitching-RT (300:1) of VRF-2.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (203.0.113.1/32, 203.0.113.0/24) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
```

```
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (2001:DB8::1/128, 2001:DB8::/32) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-2
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  200:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 209.165.200.22
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (203.0.113.1/32, 203.0.113.0/24) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!
route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  if destination in (2001:DB8::1/128, 2001:DB8::/32) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!


vrf VRF-1
 address-family ipv4 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching


vrf VRF-2
 address-family ipv4 unicast
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
```

```
 !
 address-family ipv6 unicast
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !

router bgp 200
 bgp router-id 209.165.200.22
!
 address-family vpnv4 unicast
!
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
!
 neighbor 10.40.0.1
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
 !
```

Advertise Leaked Prefix

This functionality enables you to advertise L3VPN and EVPN prefixes that are leaked from source VRF to destination VRF as L3VPN and EVPN NLRI.

Use the **import from vrf advertise-as-vpn** command under destination VRF address-family, either IPv4 or IPv6 unicast to advertise leaked (CE, redistributed and imported L3VPN and EVPN) prefixes to L3VPN and EVPN PE peers.

Use the **advertise vpnv4 unicast imported-from-vrf disable** or the **advertise vpnv6 unicast imported-from-vrf disable** command under neighbor address-family, either EVPN or L3VPN, to disable advertisement of leaked prefixes from destination VRF to EVPN or L3VPN default VRF neighbors.

Note When the router advertises from VRF-2, it advertises the leaked prefix with VRF-2 route distinguisher, and VRF-2 export route targets (replacing the paths of the existing RTs).

Note The originator loop check prevents the originator from accepting the prefix.

Figure 7: Advertise EVPN Leaked Prefix

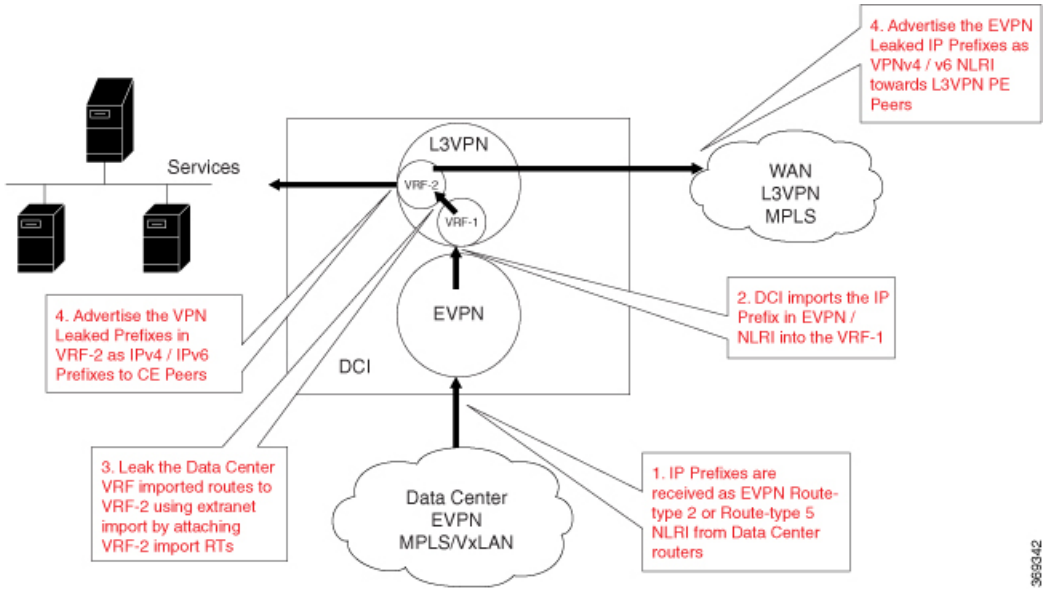
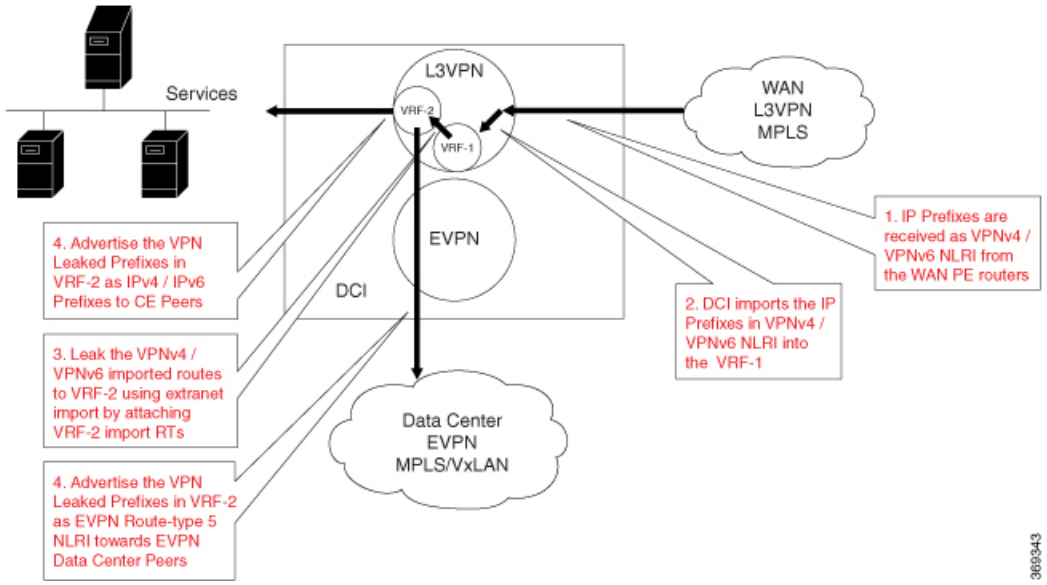


Figure 8: Advertise L3VPN Leaked Prefix



Configuration Example

In this example, the IP address (203.0.113.1/32, 2001:DB8::1/128) in EVPN route-type 2 and prefix (203.0.113.0/24, 2001:DB8::/32) in route-type 5 are imported to VRF-1 and leaked to VRF-2. The RT EXTCOMM (100:1) in NLRI matches one or more import stitching-RTs of VRF-1, and VRF-1 export policy which matches the NLRI IP address and the prefix attaches the import stitching-RT (300:1) of VRF-2.

The router imports prefixes (203.0.113.1/32, 2001:DB8::1/128, 203.0.113.0/24, 2001:DB8::/32) to VRF-1, leaks them from VRF-1 to VRF-2, and advertises them from VRF-2 to L3VPN and EVPN peers.

Use the **advertise vpnv4 unicast imported-from-vrf disable** command under neighbor 10.70.0.1 address-family VPNv4 unicast to block the advertisement of the leaked routes to neighbor 10.70.0.1.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (203.0.113.1/32, 203.0.113.0/24) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (2001:DB8::1/128, 2001:DB8::/32) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-2
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf)# import from vrf advertise-as-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  200:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 2:1
```

```
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
!

RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.60.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
!

RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
!

RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.70.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast imported-from-vrf disable
!

RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast imported-from-vrf disable
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (203.0.113.1/32, 203.0.113.0/24) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!
route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  if destination in (2001:DB8::1/128, 2001:DB8::/32) then
    set extcommunity rt (300:1) additive
```

```
  endif
end-policy
!

vrf VRF-1
 address-family ipv4 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching


vrf VRF-2
 address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
 !
 address-family ipv6 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !

router bgp 200
 bgp router-id 172.16.0.1

!
 address-family vpnv4 unicast
!
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
!
```

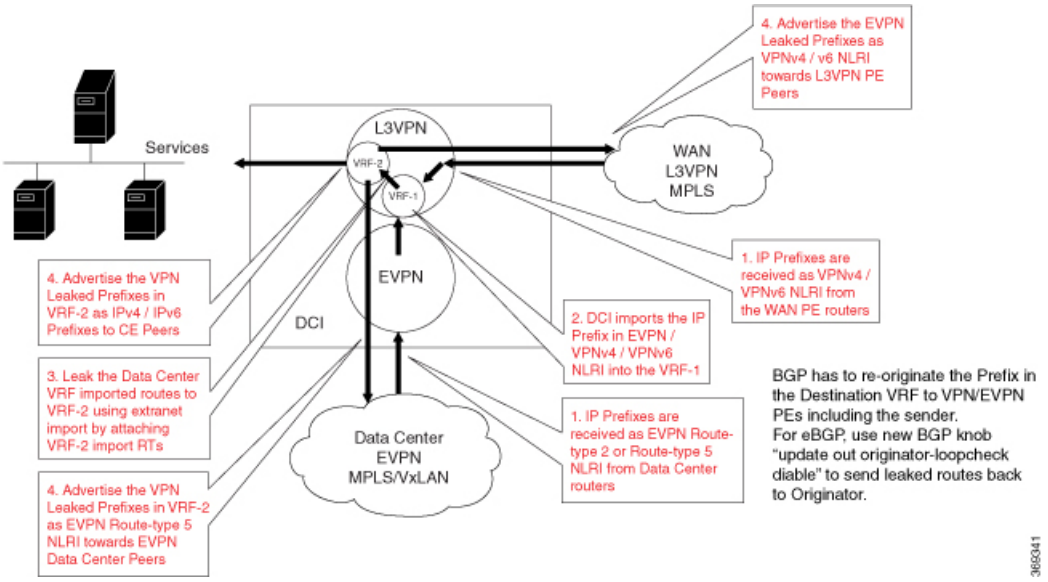
```
 neighbor 10.40.0.1

  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
 !
!
neighbor 10.60.0.1
  remote-as 200
  update-source Loopback1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   advertise vpnv4 unicast re-originated
  !
  address-family vpnv6 unicast
   import re-originate stitching-rt
   advertise vpnv6 unicast re-originated
  !
neighbor 10.70.0.1
  remote-as 200
  update-source Loopback1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   advertise vpnv4 unicast re-originated
   advertise vpnv4 unicast imported-from-vrf disable
  !
  address-family vpnv6 unicast
   import re-originate stitching-rt
   advertise vpnv6 unicast re-originated
   advertise vpnv4 unicast imported-from-vrf disable
!
!
```

Advertise Leaked Prefix Back to Originator

The router advertises leaked L3VPN and EVPN prefixes back to the originator from source VRF to destination VRF as L3VPN or EVPN NLRI.

To provide connectivity between hosts in multiple tenant VRFs, this functionality enables you to leak prefixes between tenant VRFs and advertise the leaked prefixes back to the originator. When source VRF and destination VRF are on the same data center and possibly on the same ToR and Leaf, the leaked prefixes are advertised back to the originator from the DCI.

Figure 9: Advertise Back to Originator



BGP has built-in mechanisms to prevent routing loops, and hence blocks advertisement of prefixes back to the originator. To enable advertisement back to the originator, the prefix update must override sender-side split-horizon check and receiver-side loop checks. BGP has knobs to override sender-side and receiver-side loop checks for iBGP and eBGP.

Service providers prefer eBGP peering in the data center. When the DCI has eBGP peering with the SPINE and the SPINE acting as a route reflector. The SPINE has iBGP peering with TORs in the data center. The following existing configurations impact advertisement of routes from DCI back to originating TOR:

- **as-override** configuration on the DCI router overrides the autonomous system number (ASN) of a data center SPINE with the ASN of the DCI. This configuration disables split-horizon check when the SPINE neighbor is in its own unique update group.
- **as-path-loopcheck out disable** configuration on the DCI router disables AS path loop checking for outbound updates when the SPINE neighbor is in its own unique update group.
- **allowas-in** <*x*> configuration on the SPINE to allow an AS path of data center SPINE autonomous system number (ASN) for a specified number of times. This is needed when the DCI advertises routes back to originator through the SPINE and the **as-override** command is not configured on the DCI.

The above mentioned existing knobs are not adequate to advertise the data center routes from the DCI back to the originating TOR.

Use the **update out originator-loopcheck disable** command under the neighbor configuration to advertise routes back to the originator.

To disable the originator loop checking for outbound updates, use the **update out originator-loopcheck disable** command in the BGP neighbor configuration mode. To re-enable the default originator loop checking, use the **no form** of this command.

Use one of the following configurations to advertise routes received from TOR back to the originating TOR:

- eBGP between DCI and SPINE; and iBGP between the SPINE and TOR1 and TOR2

  Configure the following commands on the DCI:

- **update out originator-loopcheck disable** under SPINE neighbor configuration mode.
- **as-override** under SPINE neighbor address-family configuration mode.

- eBGP between DCI and SPINE; and iBGP between SPINE and TOR1 and TOR2

  Configure the following commands on the DCI:

  - **update out originator-loopcheck disable** under SPINE neighbor configuration mode.
  - **as-path-loopcheck out disable** under global address-family configuration mode.

  Configure the following command on the SPINE:

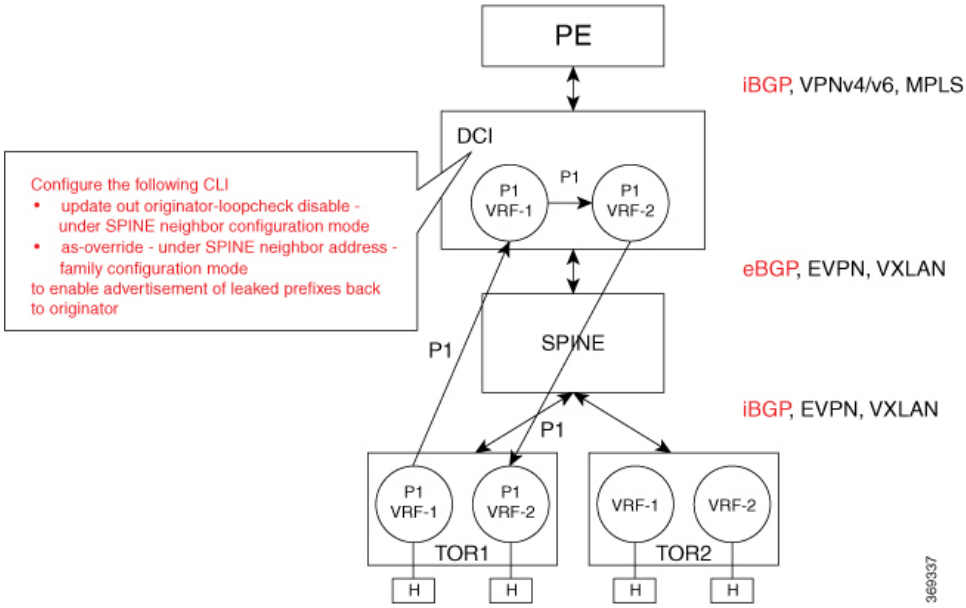
  allowas-in <*x*> command under neighbor address-family configuration mode.

- iBGP between DCI and SPINE; and iBGP between the SPINE and TOR1 and TOR2

  Configure **update out originator-loopcheck disable** command under SPINE neighbor configuration mode.

Configuration Example

Figure 10: Advertise Back to Originator using eBGP



In this example, TOR prefixes are received from the SPINE neighbor 10.40.0.1 on the DCI. The prefixes are imported to VRF-1 and then leaked to VRF-2. The leaked prefixes are advertised from VRF-2 back to the originating neighbor 10.40.0.1.

The example in the previous section *Advertise Leaked Prefix* explains how L3VPN and EVPN prefixes are imported to VRF-1 and how they are leaked to VRF-2 and advertised to L3VPN or EVPN peers, except back to the originator. To advertise the prefix back to the originating neighbor 10.40.0.1, use the **update out originator-loopcheck disable** command under eBGP neighbor configuration mode and **as-override** command under the eBGP neighbor address-family L2VPN EVPN.

```
RP/0/0/CPU0:router#(config)#router bgp 200
RP/0/0/CPU0:router#(config-bgp)#bgp router-id 172.16.0.1
RP/0/0/CPU0:router#(config-bgp)#address-family vpnv4 unicast
!
RP/0/0/CPU0:router#(config-bgp)#address-family vpnv6 unicast
!
RP/0/0/CPU0:router#(config-bgp)#address-family l2vpn evpn
RP/0/0/CPU0:router#(config-bgp-af)#exit
RP/0/0/CPU0:router#(config-bgp)#neighbor 10.40.0.1
RP/0/0/CPU0:router#(config-bgp-nbr)#remote-as 100
RP/0/0/CPU0:router#(config-bgp-nbr)#ebgp-multihop 4
RP/0/0/CPU0:router#(config-bgp-nbr)#update-source Loopback1
RP/0/0/CPU0:router#(config-bgp-nbr)#address-family l2vpn evpn
RP/0/0/CPU0:router#(config-bgp-nbr-af)#update out originator-loopcheck disable
RP/0/0/CPU0:router#(config-bgp-nbr-af)#as-override
RP/0/0/CPU0:router#(config-bgp-nbr-af)#import stitching-rt re-originate
RP/0/0/CPU0:router#(config-bgp-nbr-af)#route-policy pass in
RP/0/0/CPU0:router#(config-bgp-nbr-af)#encapsulation-type vxlan
RP/0/0/CPU0:router#(config-bgp-nbr-af)#route-policy pass out
RP/0/0/CPU0:router#(config-bgp-nbr-af)#advertise vpnv4 unicast re-originated stitching-rt
RP/0/0/CPU0:router#(config-bgp-nbr-af)#advertise vpnv6 unicast re-originated stitching-rt
RP/0/0/CPU0:router#(config-bgp-nbr-af)#commit
```

Running Configuration

This section shows the running configuration.

```
router bgp 200
bgp router-id 172.16.0.1
!
address-family vpnv4 unicast
!
address-family vpnv6 unicast
!
address-family l2vpn evpn
exit
neighbor 10.40.0.1
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
   address-family l2vpn evpn
    update out originator-loopcheck disable
   as-override
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
!
```

Lookup in Source VRF

Lookup in Source VRF functionality is enabled for an IP prefix when:

- IP prefix is leaked from the source VRF to destination VRF.
- a forwarding lookup for the prefix in destination VRF requires a second look up in the source VRF.

Lookup in source VRF functionality is used when:

- only summary prefix or default-route (0.0.0.0, ::) is leaked from source VRF to destination VRF.
- the longest match prefixes or /32 or /128 host routes are not leaked and are present in the source VRF.
- forwarding requires a match against a longest prefix or /32 or /128 host routes.

This functionality is used in the following scenarios:

- Internet access to the tenant VRF hosts – Where only the default-route (0.0.0.0, ::) is leaked from default VRF to tenant VRF. Internet prefixes are kept in the default VRF. The summary prefixes advertised to the Internet are leaked from tenant VRF to default VRF, the host routes are kept in the tenant VRF.
- Merging two VRFs – Where only the summary prefixes in the two VRFs are leaked to the other VRF.

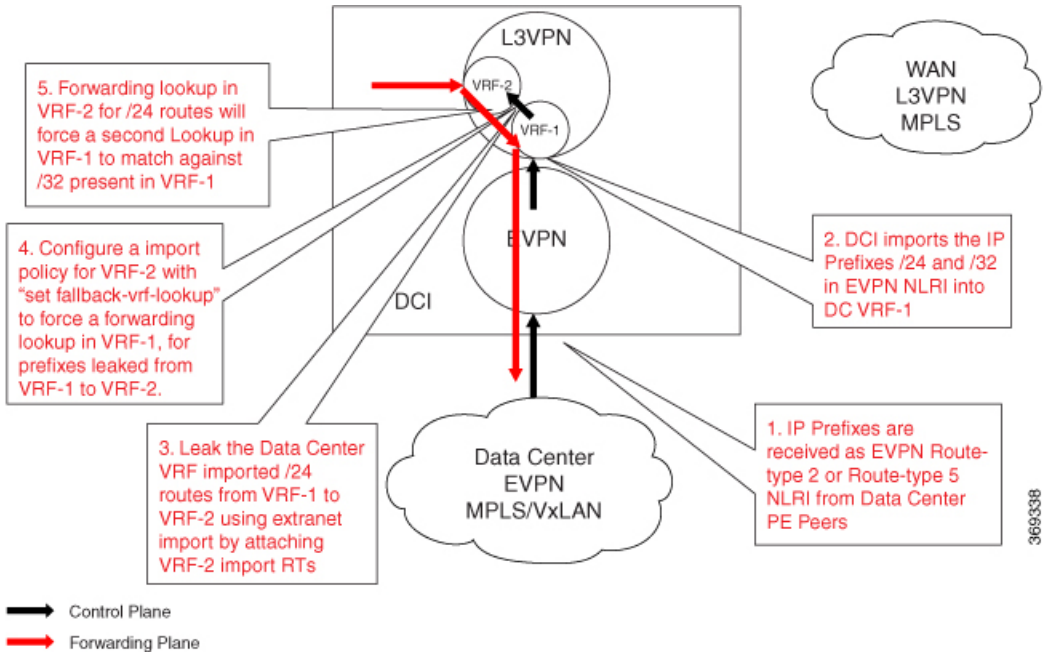
Use the **set fallback-vrf-lookup** command to enable this functionality in the following route-policy attach points:

- VRF import
- VRF import from default-vrf
- VRF export to default-vrf

The route-policy is executed when the prefix is imported to the destination VRF. Use the **set fallback-vrf-lookup** command to select the leaked prefixes that are programmed to force a second look up in the source VRF.

Configuration Example

Figure 11: Force Lookup in Source VRF



The example in the earlier section *Advertise Leaked Prefix* explains how L3VPN and EVPN prefixes are imported to VRF-1, leaked to VRF-2, and advertised to L3VPN and EVPN peers.

Without configuring the Lookup in Source VRF functionality, when traffic is received for the advertised prefix in VRF-2, a lookup is done for the prefix in VRF-2 and traffic is forwarded towards the prefix next hop.

When Lookup in Source VRF functionality is configured for L3VPN and EVPN, prefixes are imported to VRF-1 and leaked to VRF-2. And, when the **set fallback-vrf-lookup** command is used, the prefixes are advertised to L3VPN and EVPN peers. When traffic is received from the opposite direction in VRF-2, first a lookup is performed in VRF-2, which forces a second lookup in VRF-1. To force this second lookup in VRF-1, configure the following import route-policy in VRF-2.

```
RP/0/0/CPU0:router#(config)#route-policy vrf-2-import-policy
RP/0/0/CPU0:router#(config-rpl)#if destination in (203.0.113.0/24) then
RP/0/0/CPU0:router#(config-rpl-if)#set fallback-vrf-lookup
RP/0/0/CPU0:router#(config-rpl-if)#endif
RP/0/0/CPU0:router#(config-rpl)#end-policy
RP/0/0/CPU0:router#(config)#route-policy vrf-2-import-policy-v6
RP/0/0/CPU0:router#(config-rpl)#if destination in (2001:DB8::/32) then
RP/0/0/CPU0:router#(config-rpl-if)#set fallback-vrf-lookup
RP/0/0/CPU0:router#(config-rpl-if)#endif
RP/0/0/CPU0:router#(config-rpl)#end-policy
RP/0/0/CPU0:router#(config)#vrf VRF-2
RP/0/0/CPU0:router#(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router#(config-vrf-af)#import route-policy vrf-2-import-policy
RP/0/0/CPU0:router#(config-vrf-af)#import from vrf advertise-as-vpn
RP/0/0/CPU0:router#(config-vrf-af)#import route-target
RP/0/0/CPU0:router#(config-vrf-import-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-import-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-import-rt)#300:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-import-rt)#export route-target
RP/0/0/CPU0:router#(config-vrf-export-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-export-rt)#200:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-export-rt)#address-family ipv6 unicast
RP/0/0/CPU0:router#(config-vrf-af)#import route-policy vrf-2-import-policy-v6
RP/0/0/CPU0:router#(config-vrf-af)#import from vrf advertise-as-vpn
RP/0/0/CPU0:router#(config-vrf-af)#import route-target
RP/0/0/CPU0:router#(config-vrf-import-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-import-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-import-rt)#300:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-import-rt)#export route-target
RP/0/0/CPU0:router#(config-vrf-export-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-export-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-export-rt)#commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-2-import-policy
  if destination in (203.0.113.0/24) then
    set fallback-vrf-lookup
  endif
end-policy

route-policy vrf-2-import-policy-v6
  if destination in (2001:DB8::/32) then
    set fallback-vrf-lookup
  endif
```

```
end-policy


vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-2-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
!
address-family ipv6 unicast
  import route-policy vrf-2-import-policy-v6
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
```

Enable Services using EVPN VxLAN VRF Route Leaking

Service providers can use the Cisco VxLAN solution to provide the following additional services to its customers:

- Internet access
- Service VRF
- Internet full feed to customer edge devices from DCI
- Inter-VRF routing

Internet Access

The Internet access provides Internet access to tenant hosts in the data center.

For more information, see the *EVPN Default VRF Route Leaking on the DCI for Internet Connectivity* section.

Service VRF

The Service VRF enables connectivity to the services in the Service VRF to customers in the EVPN data center VRF.

For more information, see the *EVPN Service VRF Route Leaking* section.

Internet Full Feed to Customer Edge Devices from DCI

The Internet Full Feed to Customer Edge Devices from DCI service downloads the Internet prefixes on the DCI default VRF to customer edge (CE) device and provides Internet access to CE network hosts through the data center fabric.

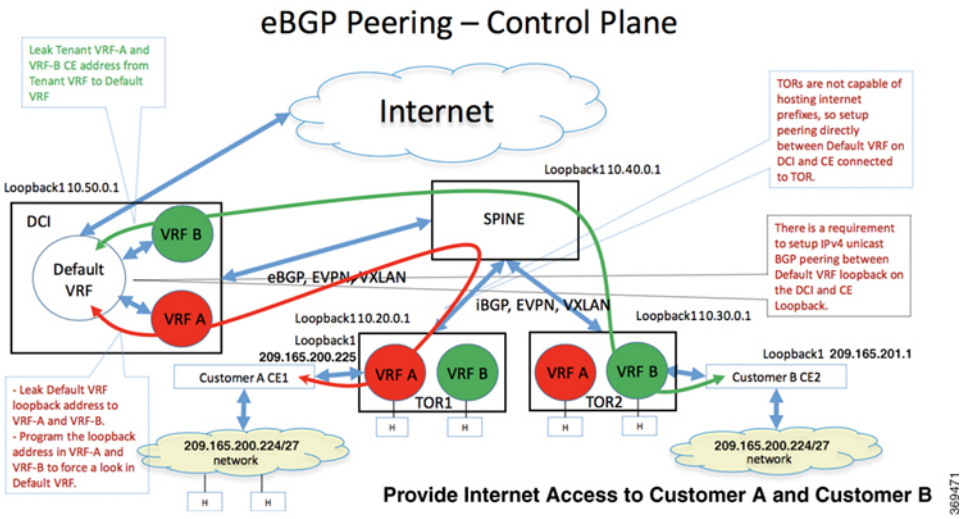
To download Internet prefixes to CE router and to provide Internet access to CE network hosts through the data center fabric, perform the followin tasks:

- Set up reachability between DCI default VRF loopback address and CE loopback address.
- Configure eBGP session between default VRF loopback address on the DCI and CE loopback address.
- Enable exchange of routes between DCI default VRF and CE.
- Provide Internet connectivity to CE network hosts through the data center fabric.

Setup the Reachability between DCI Default VRF Loopback Address and CE Loopback Address

This example shows the configuration required to set up the reachability between default VRF loopback1 address 10.50.0.1 on the DCI, and CE1 loopback1 address 209.165.200.225 connected to VRF-A on the TOR1.

Figure 12: Setup the Reachability between DCI Default VRF Loopback Address and CE Loopback Address



Propagate Reachability of DCI Default VRF Loopback Address to CE1

This section explains the configuration required to propagate the reachability of DCI default VRF loopback address to CE1 router.

DCI Configuration

- Redistribute the default VRF Looback1 address 10.50.0.1 into BGP default VRF.
- Leak the default VRF loopback address 10.50.0.1 to VRF-A, and configure the loopback address in VRF-A to force a fallback VRF lookup in the default VRF.
- Advertise DCI default VRF loopback address 10.50.0.1 in VRF-A towards TOR1 through the SPINE.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)#ipv4 address 10.50.0.1 255.255.255.255
RP/0/0/CPU0:router(config-if)# exit
RP/0/0/CPU0:router(config)# vrf VRF-A
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from default-vrf route-policy
vrf-a-default-vrf-import-policy advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export to default-vrf route-policy
vrf-a-default-vrf-export-policy allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config)# route-policy vrf-a-default-vrf-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (10.50.0.1/32, 0.0.0.0/0) then ← DCI
Default VRF loopback is leaked to VRF-A
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup ← Look up in VRF-A forces a
second look up in Default VRF
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
RP/0/0/CPU0:router(config)# router bgp 300
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
!
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1 ← SPINE
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
!
RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf VRF-A
RP/0/0/CPU0:router(config-bgp-vrf)# rd auto
RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-af)# !
```

Note Configure the default VRF loopback address 10.50.0.1 using **set fallback-vrf-lookup** command while importing to VRF-A. With this configuration, a forwarding lookup for default VRF loopback address 10.50.0.1 in VRF-A forces a second forwarding lookup in default VRF. This results in the TCP and BGP packets to hit the default VRF LPTS entries to be steered towards the TCP and BGP process in the default VRF context.

Running Configuration

This section shows the DCI running configuration.

```
interface Loopback1
ipv4 address 10.50.0.1 255.255.255.255
exit
vrf VRF-A
address-family ipv4 unicast
  import from default-vrf route-policy vrf-a-default-vrf-import-policy advertise-as-vpn
  import route-target
   1:1
   100:1 stitching
  !
  export to default-vrf route-policy vrf-a-default-vrf-export-policy allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
!
route-policy vrf-a-default-vrf-import-policy
  if destination in (10.50.0.1/32, 0.0.0.0/0) then ← DCI Default VRF loopback is leaked to
 VRF-A
    set fallback-vrf-lookup ← Look up in VRF-A forces a second look up in default VRF
    pass
  endif
end-policy

router bgp 300
bgp router-id 172.16.0.1
!
address-family ipv4 unicast
   redistribute connected
!
neighbor 10.40.0.1 ← SPINE
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
!
vrf VRF-A
  rd auto
  address-family ipv4 unicast
  !
!
```

SPINE Configuration

The SPINE acts as a route reflector and reflects the EVPN routes between DCI and TORs.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)# ipv4 address 10.40.0.1 255.255.255.255
!
RP/0/0/CPU0:router(config)# router bgp 100
RP/0/0/CPU0:router(config-bgp)# bgp router-id 192.168.0.2
!
RP/0/0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.50.0.1 ← DCI
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 300
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
```

```
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.20.0.1 ← TOR1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.30.0.1 ← TOR2
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
```

Running Configuration

This section shows the SPINE running configuration.

```
interface Loopback1
ipv4 address 10.40.0.1 255.255.255.255
!
router bgp 100
bgp router-id 192.168.0.2
!
address-family l2vpn evpn
!
neighbor 10.50.0.1 ← DCI
  remote-as 300
  ebgp-multihop 4
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
   route-policy pass in
   route-policy pass out
   next-hop-unchanged
  !
!
neighbor 10.20.0.1 ← TOR1
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
  !
!
neighbor 10.30.0.1 ← TOR2
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
  !
```

```
!
!
```

TOR1 Configuration

In this example, advertise DCI default VRF loopback address 10.50.0.1 in VRF-A to CE1 through the eBGP session between TOR1 and CE1, with TOR1 VRF-A loopback address as 10.20.0.2 as the next hop.

```
RP/0/0/CPU0:router(config)# interface Loopback2
RP/0/0/CPU0:router(config-if)# vrf foo
RP/0/0/CPU0:router(config-if)# ipv4 address 10.20.0.2 255.255.255.255
RP/0/0/CPU0:router(config-if)# router bgp 100
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.2
!
RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1 ← SPINE
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
!
RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf VRF-A
RP/0/0/CPU0:router(config-bgp-vrf)# rd auto
RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp-vrf)# neighbor 209.165.200.225 ← CE1
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# remote-as 500
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# update-source Loopback2
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# commit
```

Running Configuration

This section shows the TOR1 running configuration.

```
interface Loopback2
vrf foo
ipv4 address 10.20.0.2 255.255.255.255

router bgp 100
bgp router-id 172.16.0.2
!
address-family vpnv4 unicast
!
address-family vpnv6 unicast
!
address-family l2vpn evpn
!
neighbor 10.40.0.1 ← SPINE
  remote-as 100
  update-source Loopback1
```

```
  address-family l2vpn evpn
   encapsulation-type vxlan
   advertise vpnv4 unicast re-originated
   advertise vpnv6 unicast re-originated
  !
!
vrf VRF-A
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 209.165.200.225 ← CE1
   remote-as 500
   ebgp-multihop 4
   update-source Loopback2
   address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
   !
  !
```

CE1 Configuration

In this example, the DCI default VRF loopback address 10.50.0.1 is received by CE1 through the eBGP session between CE1 and TOR1.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)# ipv4 address 209.165.200.225 255.255.255.255
!
RP/0/0/CPU0:router(config)# router bgp 500
RP/0/0/CPU0:router(config-bgp)# bgp router-id 209.165.200.225
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp)# neighbor 10.20.0.2 ← TOR1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the CE1 running configuration.

```
interface Loopback1
ipv4 address 209.165.200.225 255.255.255.255
!
router bgp 500
bgp router-id 209.165.200.225
address-family ipv4 unicast
  redistribute connected
!
!
neighbor 10.20.0.2 ← TOR1
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
```

```
   route-policy pass in
   route-policy pass out
  !
!
!
```

Propagate CE1 Loopback Address Reachability to DCI Default VRF

This section explains the configuration required to propagate the reachability of CE1 loopback address to DCI default VRF.

CE1 Configuration

Redistribute the CE1 loopback address 209.165.200.225 into BGP. Advertise the CE1 loopback address 209.165.200.225 to TOR1 VRF-A through the eBGP session between CE1 and TOR1.

TOR1 Configuration

Advertise the CE1 loopback address 209.165.200.225 in VRF-A towards DCI.

DCI Configuration

```
RP/0/0/CPU0:router(config)# vrf VRF-A
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from default-vrf route-policy
vrf-a-default-vrf-import-policy advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-import)# export to default-vrf route-policy
vrf-a-default-vrf-export-policy allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-export-rt)# route-policy vrf-a-default-vrf-export-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.225/32) then  ← CE1 Loopback1
 is leaked from VRF-A to Default VRF
RP/0/0/CPU0:router(config-rpl-if)  pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
```

Note Advertise only the CE1 subnet address 209.165.200.224/27 to the Internet. Do not advertise the CE1 Loopback1 address 209.165.200.225/32 to the Internet.

Running Configuration

This section shows the DCI running configuration.

```
vrf VRF-A
address-family ipv4 unicast
  import from default-vrf route-policy vrf-a-default-vrf-import-policy advertise-as-vpn
  import route-target
   1:1
   100:1 stitching
```

```
  !
  export to default-vrf route-policy vrf-a-default-vrf-export-policy allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

route-policy vrf-a-default-vrf-export-policy
  if destination in (209.165.200.225/32) then  ← CE1 Loopback1 is leaked from VRF-A to
Default VRF
    pass
  endif
end-policy
```

Configure eBGP Session between DCI Default VRF and CE

This configuration brings up eBGP IPv4 unicast session between DCI default VRF loopback address and CE1 loopback address.

DCI Configuration

Configure the eBGP neighbor configuration with CE1 loopback address 209.165.200.225.

```
RP/0/0/CPU0:router(config)# router bgp 300
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
!
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
!
RP/0/0/CPU0:router(config-bgp)# neighbor 209.165.200.225 ←CE1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 500
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the DCI running configuration.

```
router bgp 300
bgp router-id 172.16.0.1
!
address-family ipv4 unicast
!
neighbor 209.165.200.225  ←CE1
  remote-as 500
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !
!
!
```

CE1 Configuration

Configure the eBGP neighbor configuration with DCI default VRF loopback address 10.50.0.1.

```
RP/0/0/CPU0:router(config)# router bgp 500
RP/0/0/CPU0:router(config-bgp)#bgp router-id 209.165.200.225
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.50.0.1 ←DCI
RP/0/0/CPU0:router(config-bgp-nbr)#remote-as 300
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)!!
```

Running Configuration

This section shows the CE1 running configuration.

```
router bgp 500
bgp router-id 209.165.200.225
address-family ipv4 unicast
  redistribute connected
!
!
neighbor 10.50.0.1 ←DCI
  remote-as 300
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !
!
!
```

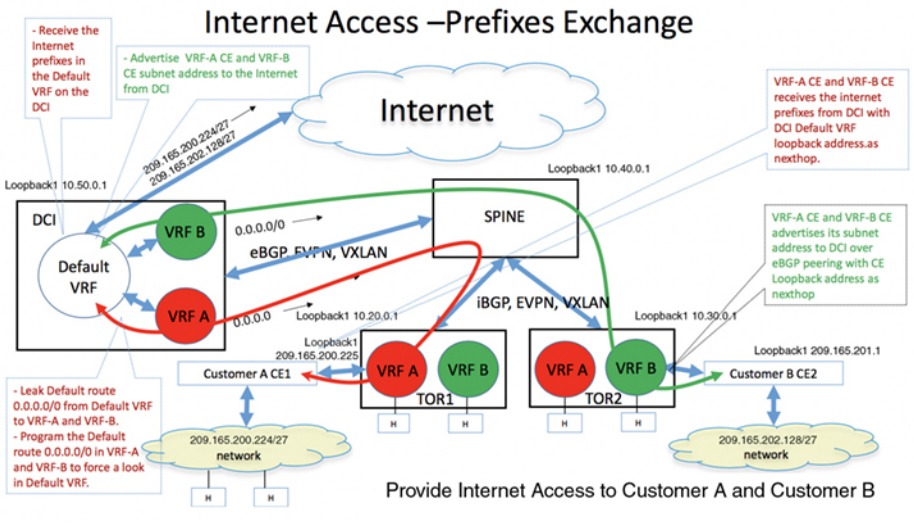
Exchange Prefixes between DCI Default VRF and CE1

You must establish an eBGP session between DCI default VRF loopback address and CE1 loopback address to enable the exchange of IPv4 unicast routes between DCI default VRF and CE1 to advertise:

- Internet prefixes on DCI default VRF to CE1 with DCI default VRF loopback address 10.50.0.1 as the next hop.
- CE1 prefix 209.165.200.224/27 to DCI default VRF with CE1 loopback address 209.165.200.225 as the next hop.

Note Advertise the same prefix 209.165.200.224/27 from CE1 into the data center VRF on TOR to enable forwarding of Internet traffic destined to CE in the data center fabric.

Figure 13: Internet Access - Prefixes Exchange



The following sections explain how to exchange prefixes between DCI default VRF and CE1:

Advertise Internet Prefixes from DCI to CE1

This section explains how to advertise Internet prefixes to CE1.

DCI

1. DCI receives the Internet prefixes in the default VRF.
2. Advertise these prefixes to CE1 with DCI default VRF loopback address 1050.0.1 as the next hop.
3. Configure the default route 0.0.0.0/0 in VRF-A to force a second look up in default VRF.
4. Advertise the default route (or configure default-originate) in VRF-A towards TOR1.

The default route in VRF-A is required to forward traffic destined to the Internet from CE1 network hosts through the data center fabric.

TOR1

TOR1 VRF-A has default route 0.0.0.0/0 with DCI default VRF loopback address 10.50.0.1 as the next hop.

Note Configure an outbound route-policy session with CE1 to block the advertisement of default route 0.0.0.0/0 to CE1.

CE1

CE1 receives Internet prefixes with DCI default VRF loopback address 10.50.0.1 as the next hop.

Note Do not advertise the Internet prefixes received by CE1 from DCI eBGP session to TOR1 eBGP session. Similarly, do not advertise the VRF-A routes received by CE1 from TOR1 eBGP session to DCI eBGP session.

Advertise CE1 Prefixes to DCI Default VRF

This section explains how to advertise CE1 subnet prefixes to the Internet.

CE1

From CE1, advertise CE1 subnet address 209.165.200.224/27 to DCI over eBGP peering with CE1 loopback address 209.165.200.225 as the next hop. Also, advertise the same CE1 subnet address 209.165.200.224/27 to VRF-A TOR1 over eBGP peering between CE1 and TOR1. This is required for forwarding traffic coming from the Internet to CE network hosts through the data center fabric.

TOR1

From TOR1, advertise the CE1 subnet address 209.165.200.224/27 to DCI VRF-A with TOR1 loopback address as the next hop.

DCI

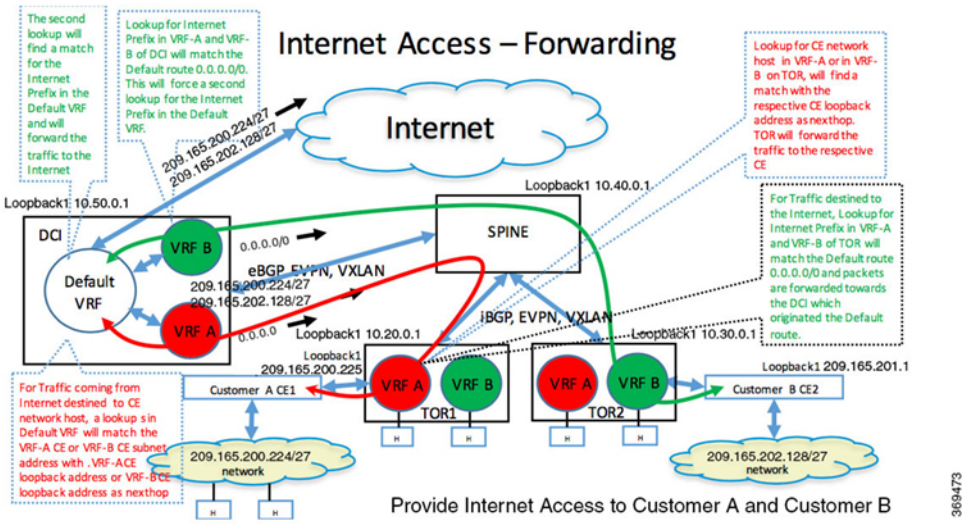
From DCI, advertise the CE1 subnet address 209.165.200.224/27 to the Internet with itself (DCI) as the next hop.

Note Do not advertise the /32 prefixes (that includes the CE loopback address 192.168.0.1/32 and DCI loopback address 10.50.0.1/32) in the default VRF to the Internet. Also, do not advertise these prefixes to CE1 through the eBGP session to CE1.

Forwarding Traffic to and from Internet in the Data Center Fabric

This section explains how forwarding works when CE1 network hosts access the Internet.

Figure 14: Forwarding Traffic to and from Internet in the Data Center Fabric



Forwarding CE1 traffic destined to Internet

This section explains how forwarding works for traffic destined to Internet from CE network hosts.

CE1

For traffic destined to Internet, a lookup for Internet prefix on CE1 finds a match with default VRF loopback address 10.50.0.1 as next hop. The next hop 10.50.0.1 is reachable through TOR1 VRF-A loopback address 10.20.0.2 forwards the traffic to TOR1 VRF-A.

TOR1

For traffic destined to the Internet from CE1, a lookup for Internet prefix in VRF-A on TOR1 matches the default route 0.0.0.0/0 with next hop pointing to the DCI, and forwards the traffic towards the DCI in VRF-A.

DCI

Lookup for Internet prefix in VRF-A matches the default route 0.0.0.0/0. The default route 0.0.0.0/0 is programmed to force a second look up in the default VRF. A second lookup in default VRF for Internet Prefix finds a match, and forwards the traffic to the Internet.

Forwarding Internet traffic destined to CE1

This section explains how forwarding works for traffic coming from Internet towards CE network hosts.

DCI

For traffic coming from Internet and destined to a CE1 network host, a lookup for CE1 network host in default VRF matches the CE1's subnet address with the CE1's loopback address as the next hop. The traffic is forwarded towards TOR1 in VRF-A.

TOR1

Lookup for CE1 network host in VRF-A matches the CE1 subnet address with CE1 loopback address as the next hop. The traffic is forwarded to CE1.

CE1

Lookup for CE1 network host on CE1 finds a match for the host route.

Inter-VRF Routing

The Inter-VRF Routing service provides connectivity between hosts in multiple data center VRFs.

You must import the routes of each VRF into the other VRF to enable Inter-VRF routing between the two VRFs. Configure the export RT in each VRF with the import RT of both VRFs or configure the import RT in each VRF with the export RTs of both VRFs. This configuration is required on all routers that have these VRFs configured. By default, it imports all the VRFs routes to the other VRF.

Alternatively, leak the routes of each VRF into the other VRF and advertise the leaked routes with the new VRFs context, such as RD and export RT to establish Inter-VRF routing between the two VRFs. This minimizes the configuration required for Inter-VRF routing to only one central router. In the data center, the DCI provides a gateway functionality through a central router. You can configure Inter-VRF routing on this central router using route leaking.

The following section explains how to enable Inter-VRF routing between the two VRFs:

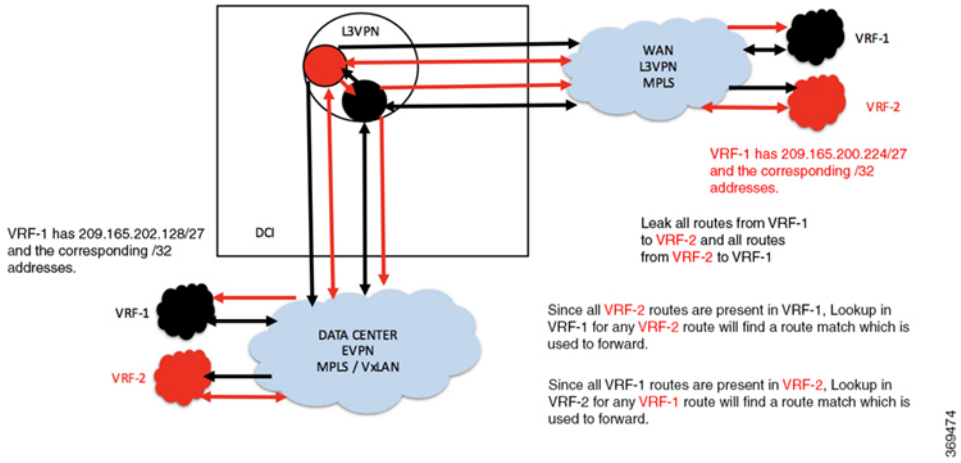
Inter-VRF Routing Between Small Number of VRFs

Leak All Routes

In this example, VRF-1 leaks all routes to VRF-2. VRF-2 leaks all routes to VRF-1. Both VRFs advertise the leaked routes in both VRFs with the context such as, RD and export RT of the leaked VRF.

Both VRFs contain the routes of both of them, so there is a reachability between hosts in both the VRFs.

Figure 15: Inter-VRF Routing - Leak All Routes



Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (400:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (300:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
```

Note: The import route target 400:1 is configured only on the DCI for leaking. Do not configure this route target 400:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 400:1 to prevent unintended import in the network.

```
RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
```

```
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!

Note: The import route target 300:1 is configured only on the DCI for leaking. Do not
configure this route target 300:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 300:1 to
prevent unintended import in the network.
```

Running Configuration

This section shows running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  set extcommunity rt (400:1) additive
  pass
end-policy
!

route-policy vrf-leak-from-vrf-2-to-vrf-1
  set extcommunity rt (300:1) additive
  pass
end-policy
!

vrf VRF-1
address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   1:1
   300:1
   100:1 stitching
   300:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   400:1
   200:1 stitching
   400:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-1
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
```

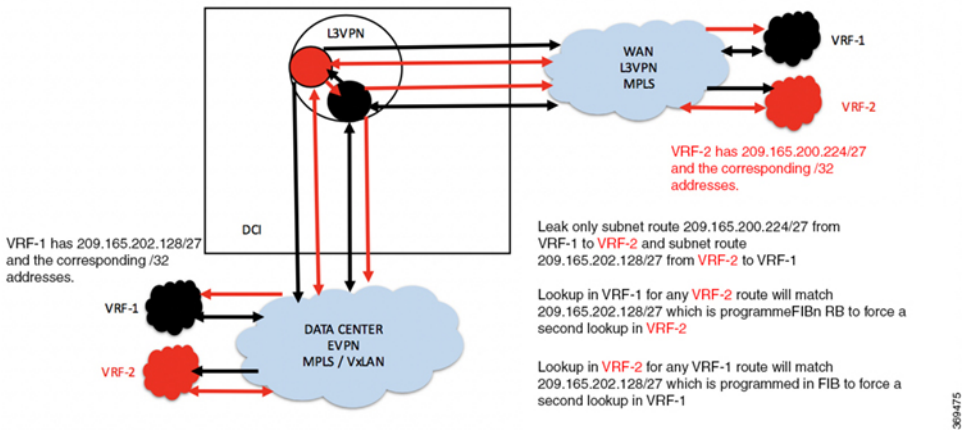
Leak Only Subnet Routes

Alternatively, in the following example, VRFs leak only /24 routes between them and they do not leak /32 route.

For forwarding to work, the traffic has to match /32 route in the original VRF.Configure the *Lookup in Source VRF* feature for the leaked /24 prefixes in destination VRF to accomplish the above-mentioned forwarding.

Lookup in VRF-1 for any VRF-2 route matches VRF-2 subnet route, which is programmed to force a second lookup in VRF-2. The second lookup finds an exact match. Similarly, lookup in VRF-2 for any VRF-1 route matches VRF-1 subnet route, which is programmed to force a second lookup in VRF-1. The second lookup finds an exact match.

Figure 16: Inter-VRF Routing - Only Subnet Routes



Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (400:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-1-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-2-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
```

```
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-1-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
```

Note: The import route target 300:1 is configured only on the DCI for leaking. Do not configure this route target 300:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 300:1 to prevent unintended import in the network.

```
RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-2-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-export-rt)# commit
```

Note: The import route target 400:1 is configured only on the DCI for leaking. Do not configure this route target 400:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 400:1 to prevent unintended import in the network.

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (209.165.200.224/27) then
    set extcommunity rt (400:1) additive
    pass
  endif
end-policy
!

route-policy vrf-1-import-policy
  if destination in (209.165.201.0/27) then
    set fallback-vrf-lookup
```

```
    pass
  endif
end-policy
!

route-policy vrf-leak-from-vrf-2-to-vrf-1
  if destination in (209.165.201.0/27) then
    set extcommunity rt (300:1) additive
    pass
  endif
end-policy
!

route-policy vrf-2-import-policy
  if destination in (209.165.200.224/27) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-1-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   300:1
   100:1 stitching
   300:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-2-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   400:1
   200:1 stitching
   400:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-1
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
```

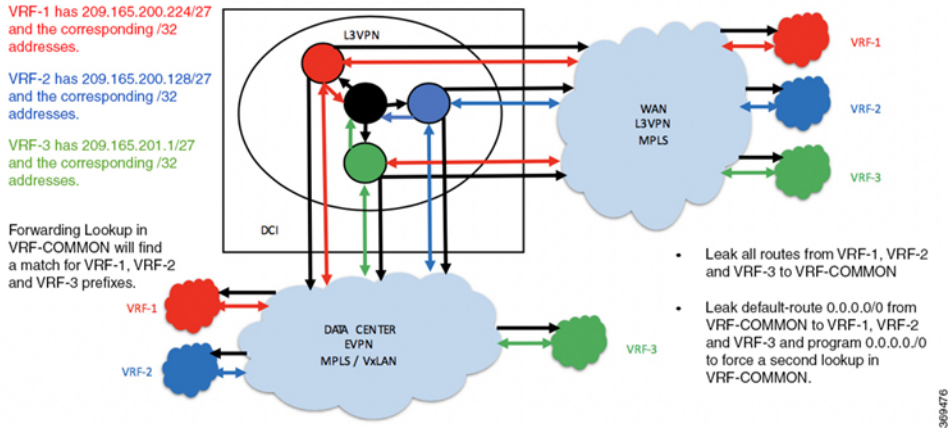
Inter-VRF Routing Between Large Number of VRFs

Leak All Routes

In this example, VRF-1, VRF-2, and VRF-3 leaks all routes to VRF-COMMON. A Static default-route 0.0.0.0/0 in VRF-COMMON is redistributed to BGP and leaked to VRF-1, VRF-2, and VRF-3. The Default-route 0.0.0.0/0 is programmed in VRF-1, VRF-2, and VRF-3 to force a fallback lookup in VRF-COMMON. VRF-1, VRF-2, and VRF-3 advertise the leaked default-route 0.0.0.0/0 with the context such as, RD and export RT of the leaked VRF, but with the local label (local vTEP in the case of VxLAN) of VRF-COMMON.

When hosts in VRF-1, VRF-2, or VRF-3 need to communicate to hosts not in its own VRF, the hosts forward the traffic to DCI. The DCI forces a lookup in VRF-COMMON, where the route finds a match and forwards the traffic to the destination host.

Figure 17: Inter-VRF Routing - Leak All Routes



Note You can enable lookup in source VRF in forwarding for a prefix leaked using **set fallback-vrf-lookup** command in the destination VRF. By default, the source VRF label for MPLS encapsulation or source VRF VNI for VxLAN encapsulation is advertised instead of destination VRF label or VNI.

For VxLAN encapsulation, if the source VRF is not configured with the VNI, then the prefix is not advertised from the destination VRF until the source VRF VNI is available. If the source VRF does not require a VNI, this default behavior can be overwritten by using the **export to vrf allow-imported-vpn disable-adv-source-vrf-vni** command in the source VRF. In this case VRF-COMMON under address family. This causes the destination VRF to send its own VNI rather than the source VRF VNI.

Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (500:1) additive
```

```
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:ROUTER(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0 eq 32) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:ROUTER(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1 stitching
!
RP/0/0/CPU0:router(config-vrf)# export route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 4:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 400:1 stitching
!
```

Note: The import route target 600:1 is configured only on the DCI for leaking. Do not configure this route target 600:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 600:1 to prevent unintended import in the network.

```
RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/0/CPU0:router(config)# vrf VRF-3
```

```
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 300:1 stitching
```

Note: The import route target 500:1 is configured only on the DCI for leaking. Do not configure this route target 500:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 500:1 to prevent unintended import in the network.

```
/* Configuration to disable sending source VRF VNI when advertising leaked
“fallback-vrf-lookup” prefix
from destination VRF */

RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
RP/0/0/CPU0:router(config-vrf)#address-family ipv6 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-to-vrf-common
  set extcommunity rt (600:1) additive
  pass
end-policy
!
route-policy vrf-leak-from-vrf-common-to-vrf
  if destination in (0.0.0.0/0) then
    set extcommunity rt (500:1) additive
    pass
  endif
end-policy
!
route-policy vrf-import-policy
  if destination in (0.0.0.0/0 eq 32) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-COMMON
address-family ipv4 unicast
  import route-target
   400:1
   600:1
   400:1 stitching
   600:1 stitching
  !
```

```
  export route-policy vrf-leak-from-vrf-common-to-vrf
  export route-target
   4:1
   400:1 stitching
  !
!

vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   500:1
   100:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
  1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   500:1
   200:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
vrf VRF-3
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   3:1
   500:1
   300:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   3:1
   300:1 stitching
  !
!
```

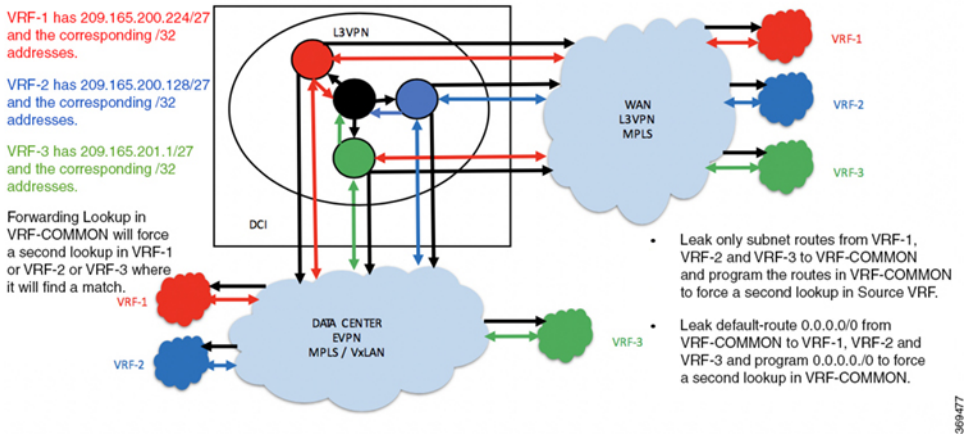
Leak Only Subnet Routes

Alternatively, in this example, VRF-1, VRF-2 and VRF-3 leaks only /24 routes to VRF-COMMON. They do not leak /32 routes. For forwarding to work, the traffic has to match /32 route in the original VRF. Configure the “Lookup in Source VRF” feature for the leaked /24 prefixes in VRF-COMMON.

In this example, VRF-1, VRF-2, and VRF-3 leaks only /24 routes to VRF-COMMON. A static default-route 0.0.0.0/0 in VRF-COMMON is redistributed to BGP and leaked to VRF-1, VRF-2, and VRF-3. The Default-route 0.0.0.0/0 is programmed in VRF-1, VRF-2, and VRF-3 to force a fallback lookup in VRF-COMMON.VRF-1, VRF-2, and VRF-3 advertise the leaked default-route 0.0.0.0/0 with the context such as, RD and export RT of the leaked VRF, but with the local label (local vTEP in the case of VxLAN) of VRF-COMMON.

When hosts in VRF-1, VRF-2, or VRF-3 need to communicate to hosts not in its own VRF, the hosts forward the traffic to DCI. The DCI forces a lookup in VRF-COMMON, where the route finds a match against /24 prefix, forcing a second look up in the original VRF, where it matches against /32 entry and forwards the traffic to the destination host.

Figure 18: Inter-VRF Routing - Leak Only Subnet Routes



Note You can enable lookup in source VRF in forwarding for a prefix leaked using **set fallback-vrf-lookup** command in the destination VRF. By default, the source VRF label for MPLS encapsulation or source VRF VNI for VxLAN encapsulation is advertised instead of destination VRF label or VNI.

For VxLAN encapsulation, if the source VRF is not configured with the VNI, then the prefix is not advertised from the destination VRF until the source VRF VNI is available. If the source VRF does not require a VNI, this default behavior can be overwritten by using the **export to vrf allow-imported-vpn disable-adv-source-vrf-vni** command in the source VRF. In this case VRF-COMMON under address family. This causes the destination VRF to send its own VNI rather than the source VRF VNI.

Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
```

```
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-3-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.202.128/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-rpl)# if destination in (0.0.0.0/0) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (500:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-common-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (209.165.200.224/27, 209.165.201.0/27,
209.165.202.128/27) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0 eq 32) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-common-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 4:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 400:1 stitching

Note: The import route target 600:1 is configured only on the DCI for leaking. Do not
configure this route target 600:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 600:1 to
prevent unintended import in the network.

RP/0/0/CPU0:ROUTER(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
```

```
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-3
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-3-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 300:1 stitching

Note: The import route target 500:1 is configured only on the DCI for leaking. Do not
configure this route target 500:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 500:1 to
prevent unintended import in the network.

/* Configuration to disable sending source VRF VNI when advertising leaked
“fallback-vrf-lookup” prefix
from destination VRF */

RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
RP/0/0/CPU0:router(config-vrf)#address-family ipv6 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-common
  if destination in (209.165.200.224/27) then
```

```
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-2-to-vrf-common
  if destination in (209.165.201.0/27) then
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-3-to-vrf-common
  if destination in (209.165.202.128/27) then
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-common-to-vrf
  if destination in (0.0.0.0/0) then
    set extcommunity rt (500:1) additive
    pass
  endif
end-policy
!
route-policy vrf-common-import-policy
  if destination in (209.165.200.224/27, 209.165.201.0/27, 209.165.202.128/27) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

route-policy vrf-import-policy
  if destination in (0.0.0.0/0 eq 32) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-COMMON
address-family ipv4 unicast
  import route-policy vrf-common-import-policy
  import route-target
   400:1
   600:1
   400:1 stitching
   600:1 stitching
  !
  export route-policy vrf-leak-from-vrf-common-to-vrf
  export route-target
   4:1
   400:1 stitching
  !
!
vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   500:1
```

```
   100:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   500:1
   200:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
vrf VRF-3
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   3:1
   500:1
   300:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-3-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   3:1
   300:1 stitching
  !
!
```

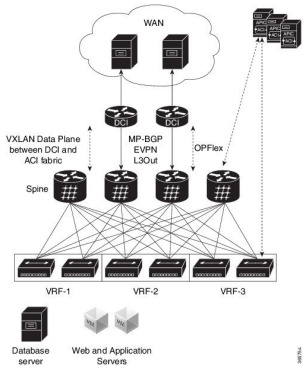
OpFlex

OpFlex is an open and extensible policy protocol used for transferring the policy information between a network policy controller such as the Cisco Application Policy Infrastructure Controller (APIC) and network elements such as routers that are configured as Data Center Interconnect (DCI) gateway. The policies are distributed using the Cisco® Application Centric Infrastructure (ACI) infrastructure within the fabric to the spine nodes. The spine nodes send policies to the DCI gateway through the OpFlex framework. An OpFlex framework resides between the spines and the DCIs. It enables the distribution of the DCI policy model from the fabric to the DCI gateways. DCI gateway acts as an OpFlex agent and the spine acts a policy repository. Fabric tenant interconnect (FTI) is the OpFlex agent application that runs on the DCI to generate and apply the tenant device configuration on the DCI. Policies configure the DCI service for a given tenant on the DCI gateway.

OpFlex Topology

Consider the topology where OpFlex framework is used between the DCI gateway and the Cisco ACI spine switches to automate fabric-facing tenant provisioning on the DCI gateway. When you configure a new external Layer 3 outside (L3Out) policy for a tenant on the Cisco Application Policy Infrastructure Controller (APIC), the controller programs all related information associated with that tenant, such as VRF instance name and BGP extended community route-target attributes for the Cisco ACI spine switches. The OpFlex framework running on the spine switches reads the L3Out managed object and converts it to the OpFlex model. This information is then pushed to the DCI gateway, which acts as a policy element for the OpFlex framework. On the DCI, the fabric facing configuration for the tenant VFR is auto-generated.

Figure 19: OpFlex Topology



Restrictions

The OpFlex feature is supported with the following restrictions:

- OpFlex feature is not supported on ASR 9000 series router with power PC based route-processor.
- FTI cannot generate configuration for multiple RTs of one address family in a tenant VRF provisioned in one fabric.
- The FTI configuration gets deleted during the following scenarios:
  - If the complete DCI configuration gets removed.
  - If the fabric gets removed or the corresponding parts of each fabric get removed.
  - If the last OpFlex peer gets removed.
- On exhaustion of FTI configuration pools, the OpFlex notifications to add tenants are ignored. If existing tenants are deleted, the new tenants must be added again to enable OpFlex notifications to be re-sent to the DCI.
- FTI supports only Type 0 RT format: 2 byte ASN + 4 byte value. Type 1 and Type 2 RT formats are not supported.
- XML configuration and oper schema are not supported for FTI configuration and show commands.
- While removing the OpFlex peers:
  - When OpFlex peer is removed from an OpFlex session, the session is shut down and the corresponding tenant configuration is marked as stale. The stale entries are deleted after the sweep timer expires. The OpFlex session is updated with the remaining peers.

- When the last OpFlex peer is removed from an OpFlex session, the session is shut down and the corresponding tenant configuration is deleted. The OpFlex session is not updated.

Configure OpFlex

Perform the following tasks to configure the OpFlex session to automate fabric-facing tenant provisioning on the DCI gateway. This includes the one-time configuration that must be done on the DCI to enable DCI hand-off from an ACI fabric.

Configure BGP

Perform this task to enable address-family under BGP routing process for fabric and WAN peering.

```
Router# configure
Router(config)# router bgp 1234
Router(config-bgp)# bgp router-id 198.51.100.1
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# commit
```

Configure BGP Session on the Fabric Side

Perform this task to configure BGP session on the fabric side.

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# neighbor 209.165.201.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback2
Router(config-bgp-nbr)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# commit
```

Configure BGP Session on the WAN Side

Perform this task to configure BGP session on the WAN side.

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# neighbor 209.165.200.226
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback2
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# import re-originate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# commit
```

Configure DCI Underlay for Fabric and WAN Interfaces

Perform this task to configure DCI underlay for fabric facing interface and WAN facing interface. Perform this task on both the interfaces.

```
Router# configure
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 209.165.200.226 255.255.255.224
Router(config-if)# commit
```

Configure IGP for ACI and WAN Reachability

Perform this task to configure IGP for ACI and WAN reachability.

```
Router# configure
Router(config)# router ospf 100
Router(config-ospf)# area 0
Router(config-ospf-ar)# interface GigabitEthernet 0/0/0/1
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# exit
Router(config-ospf)# area 100
Router(config-ospf-ar)# nssa
Router(config-ospf-ar)# interface loopback0
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# interface GigabitEthernet 0/0/0/0
Router(config-ospf-ar)# commit
```

Configure MPLS towards WAN

Perform this task to configure MPLS on the DCI.

```
Router# configure
Router# mpls ldp
Router(config-ldp)# interface GigabitEthernet 0/0/0/1
Router(config-ldp-if)# exit
Router(config-ldp)# exit
Router(config)# interface Loopback0
Router(config-if)# ipv4 address 209.165.200.227 255.255.255.224
Router(config-if)# exit
Router(config)# interface nve 1
Router(config-if)# source-interface loopback 0
Router(config-if)# commit
```

Configure FTI Auto-Configuration Parameters

Perform this task to configure FTI auto-configuration parameters.

```
Router# configure
Router(config)# dci-fabric-interconnect
Router(config-fti)# auto-configuration-pool
Router(config-fti-acp)# bgp-as 1234
Router(config-fti-acp)# bridge group bg1
Router(config-fti-acp)# vrf vrf1 ipv4-address 198.51.100.1
Router(config-fti-acp)# bd-pool 1 1000
Router(config-fti-acp)# vni-pool 1 1000
Router(config-fti-acp)# local-vtep nve 1
Router(config-fti-acp)# commit
```

Configure OpFlex Session

This task enables the fabric tenant interconnect to setup an OpFlex session with the spine.

```
Router# configure
Router(config)# dci-fabric-interconnect
Router(config-fti)# fabric 1001
Router(config-fti-fabric)# opflex-peer 192.0.2.1
Router(config-fti-fabric)# exit
Router(config-fti)# identity 203.0.113.1
Router(config-fti)# commit
```

OpFlex using Loopback Interface

The OpFlex using Loopback Interface feature prevents flapping of OpFlex session when one of the physical connections from the Data Center Interconnect (DCI) to the spine goes down. The loopback IP address which serves as the identity of the OpFlex session is used to establish the connection to the spine.

When an OpFlex session is established between the DCI and the spine where the OpFlex server is running, the session uses the physical IP address of the interface connection between the DCI and the spine to send information. When the physical connection goes down, the OpFlex session is brought down. If there is another physical connection between the spine and the DCI, the session gets re-established with this new physical IP address. However, this causes a flap. This feature enables you to have the OpFlex session up and running even when one of the physical connections to the spine goes down.

To enable this feature, use **identity loopback intf-name** command.

If you use the **identity loopback ip-address** command when the loopback IP address is unreachable, the OpFlex session falls back to using the physical IP address and continues to use even after the loopback becomes reachable. When you use the **identity loopback intf-name** command, the OpFlex session is up only as long as the loopback interface is reachable. However, only one of these two commands can be configured at a given time.

Configure OpFlex using Loopback Interface

Perfrom these tasks to configure OpFlex using loopback interface.

Configuration Example

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# dci-fabric-interconnect
RP/0/RSP0/CPU0:router(config-fti)# auto-configuration-pool
RP/0/RSP0/CPU0:router(config-fti-acp)# bgp-as 100
RP/0/RSP0/CPU0:router(config-fti-acp)# bridge group bg 1001
RP/0/RSP0/CPU0:router(config-fti-acp)# bd-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# bvi-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# vni-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# local-vtep nve 1001
RP/0/RSP0/CPU0:router(config-fti-acp)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1001
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.1
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.2
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1002
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.3
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1003
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.4
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# identity Loopback0
RP/0/RSP0/CPU0:router(config-fti)# commit
```

Running Configuration

This section shows OpFlex using loopback interface running configuration.

```
dci-fabric-interconnect
 auto-configuration-pool
  bgp-as 100
  bridge-group bg1001
```

```
 bd-pool 1001 4000
 bvi-pool 1001 4000
 vni-pool 1001 4000
 local-vtep nve 1001
!
fabric 1001
 opflex-peer 192.0.2.1
 opflex-peer 192.0.2.2
!
fabric 1002
 opflex-peer 192.0.2.3
!
fabric 1003
 opflex-peer 192.0.2.4
!
identity Loopback0
!
```

Related Topics

- OpFlex using Loopback Interface, on page 101

Associated Commands

show dci-fabric-interconnect