MPLS L3 VPN Central Services

« MPLS L3 VPN with Policy Routing (/workbook/view/service-provider-v4/task/mpls-l3-vpn-with-policy-routing-Mjg2MA%3D%3D) | MPLS L3 VPN VPNv4 Route Reflection (/workbook/view/service-provider-v4/task/mpls-l3-vpn-vpnv4-route-reflection-Mjg2Mg%3D%3D) »

Last updated: April 23, 2016

> Note:
>
> **Initial Configuration & Diagrams**: Load the initial configuration files for the section named **Multisite L3VPN**, which can be found in CCIE SPv4 Topology Diagrams & Initial Configurations (http://labs.ine.com/workbook/view/service-provider-v4/task/ccie-spv4-topology-diagrams-initial-configs). Refer to the **Multisite L3VPN Diagram** in order to complete this task.

CONTENTS

Task

- Configure a VRF VPN_A on R2 with Route Distinguisher 100:1 and assign it to the link connecting to R1.
- Configure a VRF VPN_B on R4 with Route Distinguisher 100:2 and assign it to the link connecting to R7.
- Configure a VRF VPN_C on R5 with Route Distinguisher 100:3 and assign it to the link connecting to R8.
- Configure a VRF VPN_D on XR1 with Route Distinguisher 100:4 and assign it to the link connecting to XR2.
- Configure RIPv2 on all the PE-CE links and advertise the CE's Loopback0 interfaces into RIP.
- Configure BGP on R2, R4, R5, and XR1 as follows:
  - Use BGP AS 100.
  - Configure a full mesh of iBGP VPNv4 peerings between the PE routers.
  - Use their Loopback0 interfaces as the source of the BGP sessions.
  - Redistribute between the VRF aware RIP processes and VPNv4 BGP.
- Configure the Route Target Export policies on the PE routers as follows:
  - R2 should export R1's Loopback0 with RT 2.2.2.2:7 and 2.2.2.2:8.
  - XR1 should export XR2's Loopback0 with RT 19.19.19.19:7 and 19.19.19.19:8.
  - R4 should export R7's Loopback0 with RT 100:7.
  - R5 should export R5's Loopback0 with RT 100:8.
  - Do not export the PE to CE links into BGP.
- Configure the Route Target Import policies on the PE routers as follows:
  - R2 and XR1 should both import RTs 100:7 and 100:8.
  - R4 should import RTs 2.2.2.2:7 and 19.19.19.19:7.
  - R5 should import RTs 2.2.2.2:8 and 19.19.19.19:8.
- Once complete, only the following reachability should be achieved:
  - R7 should be able to reach R1 and XR2's Loopback0 networks when sourcing traffic from its own Loopback0.
  - R8 should be able to reach R1 and XR2's Loopback0 networks when sourcing traffic from its own Loopback0.

Configuration

Click to *collapse*

CONTENTS

```
R1:
router rip
 version 2
 no auto-summary
 network 10.0.0.0
 network 1.0.0.0

R7:
router rip
 version 2
 no auto-summary
 network 10.0.0.0
 network 7.0.0.0

R8:
router rip
 version 2
 no auto-summary
 network 10.0.0.0
 network 8.0.0.0

R2:
vrf definition VPN_A
 rd 100:1
 route-target import 100:7
 route-target import 100:8
 !
 address-family ipv4
 export map R2_EXPORT_MAP
 exit-address-family
!
ip prefix-list R1_LOOPBACK seq 5 permit 1.1.1.1/32
!
route-map R2_EXPORT_MAP permit 10
 match ip address prefix-list R1_LOOPBACK
 set extcommunity rt 2.2.2.2:7 2.2.2.2:8
!
interface GigabitEthernet1.12
 vrf forwarding VPN_A
 ip address 10.1.2.2 255.255.255.0
!
router rip
 !
 address-family ipv4 vrf VPN_A
  redistribute bgp 100 metric transparent
  network 10.0.0.0
  no auto-summary
  version 2
exit-address-family
!
router bgp 100
 template peer-session VPNv4_IBGP_SESSION
  remote-as 100
```

```
  update-source Loopback0
 exit-peer-session
 !
 bgp log-neighbor-changes
 neighbor 4.4.4.4 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 5.5.5.5 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 19.19.19.19 inherit peer-session VPNv4_IBGP_SESSION
 !
 address-family vpnv4
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
  neighbor 5.5.5.5 activate
  neighbor 5.5.5.5 send-community extended
  neighbor 19.19.19.19 activate
  neighbor 19.19.19.19 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN_A
  redistribute rip
 exit-address-family


R4:
vrf definition VPN_B
 rd 100:2
 route-target import 2.2.2.2:7
 route-target import 19.19.19.19:7
 !
 address-family ipv4
 export map R4_EXPORT_MAP
 exit-address-family
!
ip prefix-list R7_LOOPBACK seq 5 permit 7.7.7.7/32
!
route-map R4_EXPORT_MAP permit 10
 match ip address prefix-list R7_LOOPBACK
 set extcommunity rt 100:7
!
!
interface GigabitEthernet1.47
 vrf forwarding VPN_B
 ip address 10.4.7.4 255.255.255.0
!
router rip
 !
address-family ipv4 vrf VPN_B
 redistribute bgp 100 metric transparent
 network 10.0.0.0
 no auto-summary
 version 2
exit-address-family
!
router bgp 100
 template peer-session VPNv4_IBGP_SESSION
  remote-as 100
```

```
  update-source Loopback0
 exit-peer-session
 !
 bgp log-neighbor-changes
 neighbor 2.2.2.2 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 5.5.5.5 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 19.19.19.19 inherit peer-session VPNv4_IBGP_SESSION
 !
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 5.5.5.5 activate
  neighbor 5.5.5.5 send-community extended
  neighbor 19.19.19.19 activate
  neighbor 19.19.19.19 send-community extended
 exit-address-family
!
address-family ipv4 vrf VPN_B
 redistribute rip
exit-address-family


R5:
vrf definition VPN_C
 rd 100:3
 route-target import 2.2.2.2:8
 route-target import 19.19.19.19:8
 !
 address-family ipv4
 export map R5_EXPORT_MAP
 exit-address-family
!
ip prefix-list R8_LOOPBACK seq 5 permit 8.8.8.8/32
!
route-map R5_EXPORT_MAP permit 10
 match ip address prefix-list R8_LOOPBACK
 set extcommunity rt    100:8
!
interface GigabitEthernet1.58
 vrf forwarding VPN_C
 ip address 10.5.8.5 255.255.255.0
!
router rip
 !
 address-family ipv4 vrf VPN_C
 redistribute bgp 100 metric transparent
 network 10.0.0.0
 no auto-summary
 version 2
exit-address-family
!
router bgp 100
 template peer-session VPNv4_IBGP_SESSION
  remote-as 100
  update-source Loopback0
```

```
 exit-peer-session
 !
 bgp log-neighbor-changes
 neighbor 2.2.2.2 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 4.4.4.4 inherit peer-session VPNv4_IBGP_SESSION
 neighbor 19.19.19.19 inherit peer-session VPNv4_IBGP_SESSION
 !
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 4.4.4.4 activate
  neighbor 4.4.4.4 send-community extended
  neighbor 19.19.19.19 activate
  neighbor 19.19.19.19 send-community extended
 exit-address-family
!
address-family ipv4 vrf VPN_C
 redistribute rip
exit-address-family


XR1:
vrf VPN_D
 address-family ipv4 unicast
  import route-target
   100:7
   100:8
  !
  export route-policy EXPORT_POLICY
 !
!
interface GigabitEthernet0/0/0/0.1920
 no ipv4 address
 vrf VPN_D
 ipv4 address 10.19.20.19 255.255.255.0
!
route-policy EXPORT_POLICY
 if destination in (20.20.20.20/32) then
    set extcommunity rt (19.19.19.19:7, 19.19.19.19:8)
 endif
end-policy
!
router bgp 100
 address-family vpnv4 unicast
 !
 neighbor-group VPNv4_IBGP
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  !
 !
 neighbor 2.2.2.2
  use neighbor-group VPNv4_IBGP
 !
 neighbor 4.4.4.4
```

```
  use neighbor-group VPNv4_IBGP
 !
 neighbor 5.5.5.5
  use neighbor-group VPNv4_IBGP
 !
 vrf VPN_D
  rd 100:4
  address-family ipv4 unicast
   redistribute rip
  !
 !
!
router rip
 vrf VPN_D
  interface GigabitEthernet0/0/0/0.1920
  !
  redistribute bgp 100
 !
!
end


XR2:
router rip
 interface Loopback0
 !
interface GigabitEthernet0/0/0/0.1920
 !
!
```

CONTENTS

Verification

Central Services VPNs, sometimes called overlapping VPNs, allows multiple customers of the Service Provider network to access a centralized service in the SP network, for example hosted email, while still maintaining the separation of different customer routing tables. From a technical standpoint, the reason that this design works is that a VPNv4 BGP route can have multiple Route Target values at the same time, which means that the single route can be a member of multiple VPNs at the same time.

This is where the key distinction comes in between the VPNv4 Route Distinguisher and the VPNv4 Route Target. The Route Distinguisher (RD) is used to make the route unique, which allows different customers to use the same IP addressing scheme, for example RFC 1918 space, while the Route Target (RT) defines the route's VPN membership. A VPNv4 route will always have only one Route Distinguisher, but it can have multiple Route Targets.

Specifically in this example the Central Services could be represented by the Loopback0 networks of R1 and XR2, 1.1.1.1/32 and 20.20.20.20/32 respectively. When the PE routers R2 and XR1 export these prefixes into the VPNv4 BGP network, an **export-map** is used to apply a specific policy to the Route Target values. This feature gives you more control over which prefixes get which targets, which could include more than one RT, or no RTs at all.

The first step in verifying this design is to ensure that the VPNv4 routes have been tagged with the proper RT values as they are exported from the VRF into VPNv4 BGP, as seen below.

Note:

Note: BGP templates were used on IOS, and BGP neighbor groups were used on IOS-XR to accomplish the full mesh BGP configuration. Both of these methods ease large and repetitive BGP configurations, but are not required to complete this task.

```
R2#show bgp vpnv4 unicast all 1.1.1.1/32

Paths: (1 available, best #1, table VPN_A)

  Advertised to update-groups:

     1

  Refresh Epoch 1

  Local

    10.1.2.1 (via vrf VPN_A) from 0.0.0.0 (2.2.2.2)

      Origin incomplete, metric 1, localpref 100, weight 32768, valid, sourced, best

       Extended Community: RT:2.2.2.2:7 RT:2.2.2.2:8

      mpls labels in/out 24/nolabel

      rx pathid: 0, tx pathid: 0x0
```

```
RP/0/0/CPU0:XR1#show bgp vpnv4 unicast rd 100:4 20.20.20.20/32

Wed May  6 03:13:37.122 UTC

BGP routing table entry for 20.20.20.20/32, Route Distinguisher: 100:4

Versions:

  Process           bRIB/RIB  SendTblVer

  Speaker                 11          11

    Local Label: 16016

Last Modified: May  5 23:58:00.451 for 03:15:36

Paths: (1 available, best #1)

  Advertised to update-groups (with more than one peer):

    0.2

  Path #1: Received by speaker 0

  Advertised to update-groups (with more than one peer):

    0.2

  Local

    10.19.20.20 from 0.0.0.0 (19.19.19.19)

      Origin incomplete, metric 1, localpref 100, weight 32768, valid, redistributed, best, group-best, import-candidate

      Received Path ID 0, Local Path ID 1, version 11

       Extended community: RT:19.19.19.19:7 RT:19.19.19.19:8
```

On the remote PEs, the import policy matches against the Route Target values to determine whether the route should be imported from VPNv4 into a local VRF. As long as the VPNv4 route has one of the RTs matched by the import policy, the route will be imported. In this specific case R4 imports 2.2.2.2:7 and 19.19.19.19:7, while R5 imports 2.2.2.2:8 and 19.19.19.19:8. The actual values used are arbitrary since there is no hierarchy to Route Targets. In practical implementations most service providers use external applications to track the Route Distinguisher and Route Target values assigned to specific customers and services to keep their configurations more manageable.

```
R4#show bgp vpnv4 unicast all

BGP table version is 104, local router ID is 4.4.4.4

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,

              x best-external, a additional-path, c RIB-compressed,

Origin codes: i - IGP, e - EGP, ? - incomplete

RPKI validation codes: V valid, I invalid, N Not found


     Network          Next Hop            Metric LocPrf Weight Path

Route Distinguisher: 100:1

 *>i 1.1.1.1/32       2.2.2.2                  1    100      0 ?

Route Distinguisher: 100:2 (default for vrf VPN_B)

 *>i 1.1.1.1/32       2.2.2.2                  1    100      0 ?

 *>  7.7.7.7/32       10.4.7.7                 1         32768 ?

 *>  10.4.7.0/24      0.0.0.0                  0         32768 ?

 *>i 20.20.20.20/32   19.19.19.19              1    100      0 ?

Route Distinguisher: 100:4

 *>i 20.20.20.20/32   19.19.19.19              1    100      0 ?


R4#show ip route vrf VPN_B bgp


Routing Table: VPN_B

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP

       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

       E1 - OSPF external type 1, E2 - OSPF external type 2

       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

       ia - IS-IS inter area, * - candidate default, U - per-user static route

       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP

       a - application route

       + - replicated route, % - next hop override


Gateway of last resort is not set


      1.0.0.0/32 is subnetted, 1 subnets

B        1.1.1.1 [200/1] via 2.2.2.2, 1d18h

      20.0.0.0/32 is subnetted, 1 subnets

B        20.20.20.20 [200/1] via 19.19.19.19, 1d21h



R5#show bgp vpnv4 unicast all

BGP table version is 100, local router ID is 5.5.5.5

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,

              x best-external, a additional-path, c RIB-compressed,

Origin codes: i - IGP, e - EGP, ? - incomplete

RPKI validation codes: V valid, I invalid, N Not found


     Network          Next Hop            Metric LocPrf Weight Path

Route Distinguisher: 100:1

 *>i 1.1.1.1/32       2.2.2.2                  1    100      0 ?

Route Distinguisher: 100:3 (default for vrf VPN_C)

 *>i 1.1.1.1/32       2.2.2.2                  1    100      0 ?
```

```
 *>  8.8.8.8/32       10.5.8.8                 1         32768 ?
 *>  10.5.8.0/24      0.0.0.0                  0         32768 ?
 *>i 20.20.20.20/32   19.19.19.19              1    100      0 ?
Route Distinguisher: 100:4
 *>i 20.20.20.20/32   19.19.19.19              1    100      0 ?


R5#show ip route vrf VPN_C bgp


Routing Table: VPN_C
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      1.0.0.0/32 is subnetted, 1 subnets
B        1.1.1.1 [200/1] via 2.2.2.2, 1d18h
      20.0.0.0/32 is subnetted, 1 subnets
B        20.20.20.20 [200/1] via 19.19.19.19, 1d21h
```

Note that since R2 is not importing the RT values that XR1 is exporting, and vice-versa, and likewise R4 is not importing the RT values that R5 is exporting, and vice-versa, these routes do not appear in their local VPNv4 topologies. For example R4 sees the route to 20.20.20.20/32 in the VPNv4 table, but R2 does not.

CONTENTS >

```
R4#show bgp vpnv4 unicast all 20.20.20.20/32
BGP routing table entry for 100:2:20.20.20.20/32, version 10
Paths: (1 available, best #1, table VPN_B)
  Not advertised to any peer
  Refresh Epoch 1
  Local, imported path from 100:4:20.20.20.20/32 (global)
    19.19.19.19 (metric 3) (via default) from 19.19.19.19 (19.19.19.19)
      Origin incomplete, metric 1, localpref 100, valid, internal, best
       Extended Community: RT:19.19.19.19:7 RT:19.19.19.19:8
      mpls labels in/out nolabel/16016
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for 100:4:20.20.20.20/32, version 11
Paths: (1 available, best #1, no table)
  Not advertised to any peer
  Refresh Epoch 1
  Local
    19.19.19.19 (metric 3) (via default) from 19.19.19.19 (19.19.19.19)
      Origin incomplete, metric 1, localpref 100, valid, internal, best
       Extended Community: RT:19.19.19.19:7 RT:19.19.19.19:8
      mpls labels in/out nolabel/16016
      rx pathid: 0, tx pathid: 0x0



R2#show bgp vpnv4 unicast all 20.20.20.20/32
% Network not in table
```

This is due to an optimization of inbound VPNv4 filtering of the BGP process, and is the default and desirable behavior. Specifically what is occurring here is that when R2 receives VPNv4 routes from its peers, it looks at the Route Target values that are in the BGP extended communities fields. If none of the Route Target values of the route match a local import policy, the route is automatically discarded. This helps to keep the size of the VPNv4 BGP table smaller, as routes for customers that a local PE is not servicing can be discarded.

This filtering can be viewed in real time by observing the output of the **debug ip bgp vpnv4 unicast update**, as seen below.

```
R2#debug ip bgp vpnv4 unicast update
BGP updates debugging is on for address family: VPNv4 Unicast


R2#clear bgp vpnv4 unicast 19.19.19.19 in
R2#
BGP(4): 19.19.19.19 rcvd UPDATE w/ attr: nexthop 19.19.19.19, origin ?, localpref 100, metric 1,  extended community RT:19.19.19.19:7 RT:19.
19.19.19:8
BGP(4): 19.19.19.19 rcvd  100:4:20.20.20.20/32, label 16016 -- DENIED due to:  extended community not supported;
BGP(4): 19.19.19.19 rcvd UPDATE w/ attr: nexthop 19.19.19.19, origin ?, localpref 100, metric 0
BGP(4): 19.19.19.19 rcvd  100:4:10.19.20.0/24, label 16007 -- DENIED due to:  extended community not supported;
```

The output *extended community not supported* means that there is not a RT that matches a local import policy. Note that the first VPNv4 prefix, 100:4:20.20.20.20/32 has RT values 19.19.19.19:7 and 19.19.19.19:8, while the second VPNv4 prefix 100:4:10.19.20.0/24 has no RT values at all. This is due to the fact that the export route-policy configured on XR1 did not match 10.19.20.0/24, which effectively means that no other PE can import this route.

The final result of this design should be that R7 and R8's Loopback0s can reach the Loopback0s of R1 and XR2, while no other connectivity is permitted.

```
R7#ping 1.1.1.1 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 7.7.7.7
!!!!!
 Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

R7#ping 20.20.20.20 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.20.20.20, timeout is 2 seconds:
Packet sent with a source address of 7.7.7.7
!!!!!
 Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms

R7#ping 1.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
.....
 Success rate is 0 percent (0/5)

R8#ping 1.1.1.1 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 8.8.8.8
!!!!!
 Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

R8#ping 20.20.20.20 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.20.20.20, timeout is 2 seconds:
Packet sent with a source address of 8.8.8.8
!!!!!
 Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/4 ms

R8#ping 1.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
.....
 Success rate is 0 percent (0/5)
```

« MPLS L3 VPN with Policy Routing (/workbook/view/service-provider-v4/task/mpls-l3-vpn-with-policy-routing-Mjg2MA%3D%3D) | MPLS L3 VPN VPNv4 Route Reflection (/workbook/view/service-provider-v4/task/mpls-l3-vpn-vpnv4-route-reflection-Mjg2Mg%3D%3D) »