
# *Route 109 Overview*

## Disclaimer:

Route 109 is a fictitious network provider that will act as a vessel to explore various technologies being implemented within today's internet service provider landscape. The goal of the Route 109 project is strictly educational and does not serve as any meaningful implementation guide. This project is personal work and holds no affiliation to any external entities. The following report will only serve as basic documentation.

## Introduction:

The purpose of this project was simply to gain a further understanding of service delivery technologies and associated design methodology. Beyond simple exposure to specialized technology, the inherent complexity of this lab provides a better understanding of many aspects of computer networking. The following document will be broken into sections with further discussion regarding the how and whys with accompanying technical discussion.

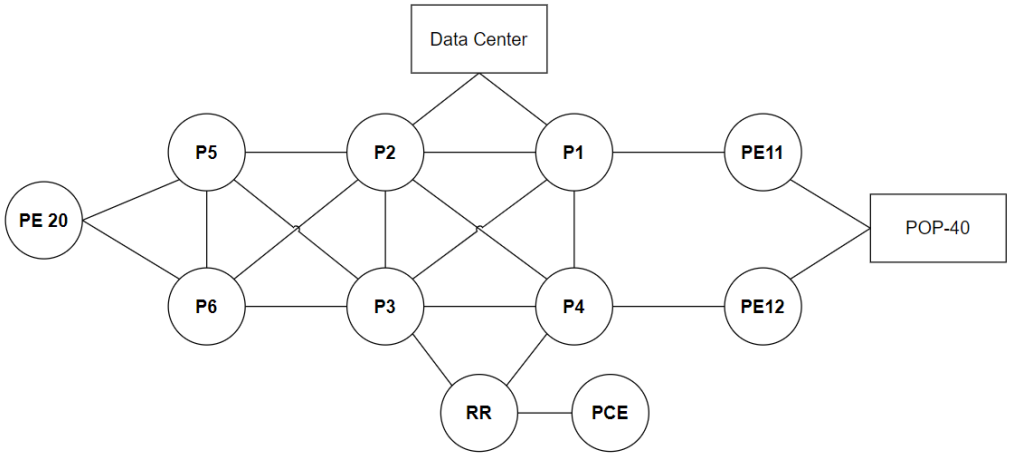
## Design Overview:

The Route 109 project consists strictly of Cisco physical and virtual devices performing routing and switching operations. Besides the traditional networking infrastructure, Linux and Windows devices will simulate various end users and provide supporting services. A single F5 Virtual Edition will be used within the data center environment. A complete list of components can be found below.

| Device | Software | Number Of Nodes |
|---|---|---|
| Cisco IOS-XRv 9000 | 7.7.2 | 13 |
| Cisco Catalyst 8000v | 17.6.5a | 8 |
| Cisco Nexus 9000v | 10.4(1) | 5 |
| Cisco C3650 | 16.12.9 | 1 |
| F5 BIG-IP VE | 17.1.0.3 | 1 |
| VMware ESXI Enterprise | 7.0.3 | 1 |
| Cisco NSO VM | 6.1 | 1 |

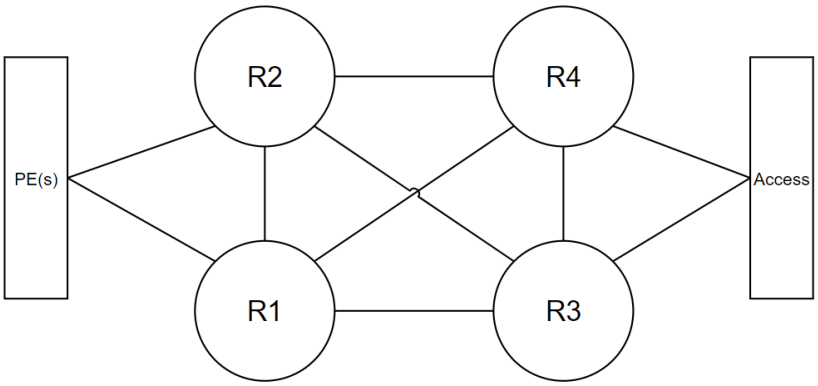
**Topology Diagrams:**

**Core:**



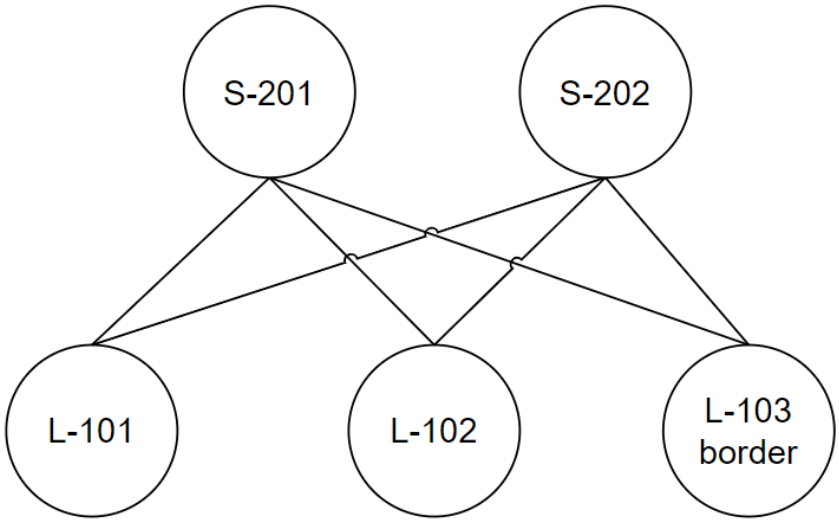

The core has been implemented with redundancy and path options in mind. This design allows for node and link failure while retaining active paths spanning edge regions. The core is broken into multiple IGP domains and orchestrated via a path computational element (PCE), which will be discussed further in future sections. The core consists of only IOS-XR devices.

**Point of Presence (POP):**




The point of presence architecture is relatively straightforward. This design allows equal-cost paths to direct north and south traffic to and from the provider core. Like the core, this realm will be orchestrated via PCE to make adjustments and provide path variation based on operator-specified parameters. The point of presence consists of IOS-XE and IOS-XR devices.

**Data Center:**



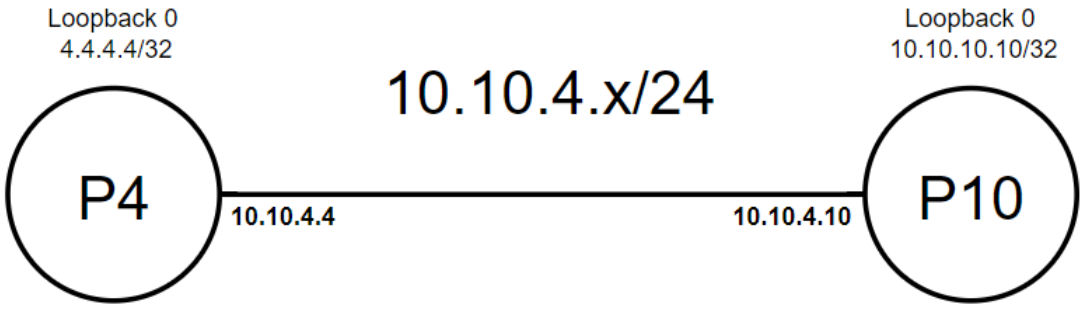

The Data Center module uses a Clos architecture, which allows the provisioning of VXLAN-based services. This design is optimized for east-west traffic patterns and allows seamless host mobility. This module will interact via external border gateway protocol (eBGP) with the service provider network providing external connectivity. The data center consists of only NX-OS devices within the switching fabric.

**Addressing**

In the lab, attempts will be made to retain a predictable addressing scheme. The general practice in use is

*10.n1.n2.n*

N1 is the higher numeric value node, with N2 being the lower value. The final octet value is the node where the address is present. A CIDR length of 24 has been used to keep things simple between nodes. Loopback addressing will be the node number repeated (e.g., 10.10.10.10/32). The diagram below depicts the addressing scheme.



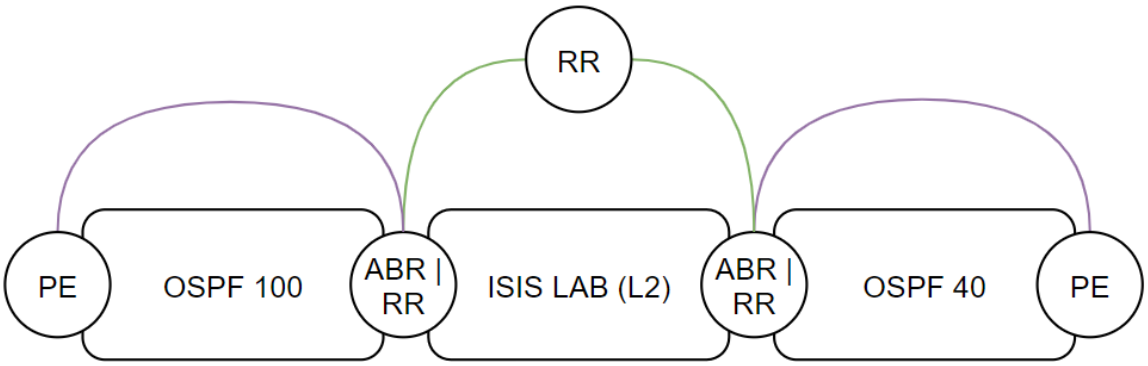

On the data center nodes *IP unnumbered* has been utilized on multiple links. The data center underlay configuration section will discuss the specific IPs used. The loopback addresses have slightly deviated within the point of presence module and will use the 10.40.n.n format.

## Network Underlay:

The following section will discuss the underlay networks for all three modules. This section will only cover basic IGP functions and configuration needs. Additional protocols interacting with the IGP will be discussed further in their respective sections.

### Core & POP:

The core underlay will consist of three separate IGP domains. The separation of domains provides scalability and creates trouble isolation. In this lab, the forwarding (non-ABR) routers in each domain will not know each other. BGP will provide the necessary reachability for certain endpoints across the domains.




As shown above, the breaking up of domains creates a modular design that provides central troubleshooting points when investigating end-to-end forwarding. Data plane forwarding will be handled via segment routing and discussed in future sections. The baseline IGP configurations are displayed in the following code blocks accompanied with brief explanations.

```
!
router ospf 100
 distribute link-state instance-id 100
 segment-routing mpls
 segment-routing forwarding mpls
 segment-routing sr-prefer
 area 0
  prefix-suppression
  mpls traffic-eng
  interface Loopback0
   prefix-sid index 3
  !
  interface GigabitEthernet0/0/0/5
   network point-to-point
  !
  interface GigabitEthernet0/0/0/6
   network point-to-point
  !
 !
!
```

IOS XR node OSPF Configuration

```
router ospf 40
 router-id 10.40.1.1
 prefix-suppression
 segment-routing mpls
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
```

IOS-XE node OSPF Configuration

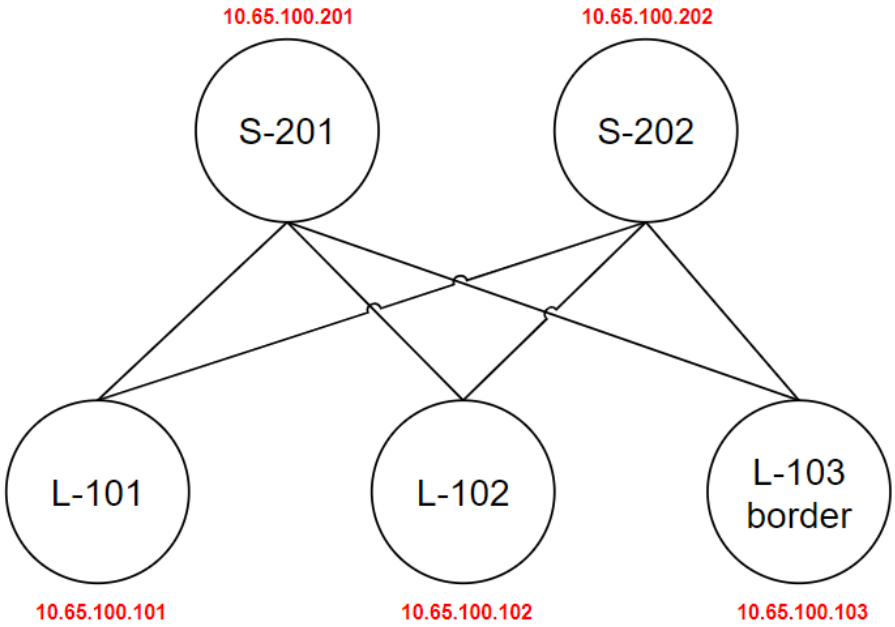
All links between nodes are configured as point to point networks. Segment routing extensions have been enabled, and link state information is being distributed. The link state information will be forwarded via BGP to enable inter-domain traffic engineering. Prefix suppression is being performed to minimize table size.

```
router isis LAB
 net 49.0000.0000.0000.0003.00
 address-family ipv4 unicast
  metric-style wide
  advertise passive-only
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  router-id 3.3.3.3
  segment-routing mpls sr-prefer
 !
 interface Loopback0
  passive
  circuit-type level-2-only
  address-family ipv4 unicast
   prefix-sid index 3
  !
 !
 interface GigabitEthernet0/0/0/0
  circuit-type level-2-only
  point-to-point
  address-family ipv4 unicast
  !
```

IOS-XR node IS-IS Configuration

The IS-IS configuration is similar to OSPF. Metric-style wide has been configured to allow segment routing TLVs to pass properly. Segment routing extensions have also been enabled. Like prefix suppression in OSPF, the advertised passive-only command has been used to limit table size. All links are configured as level 2 only and operate as point-to-point links.

**Data Center:**
Data center devices will utilize OSPF as an IGP underlay. The interconnections will be configured as point-to-point IP unnumbered links. The data center module loopback addressing is shown below:




Date Center Module Loopback Addressing

The OSPF configuration for the data center underlay is simple as it will only serve as a reachability method for BGP endpoints directing the overlay traffic.

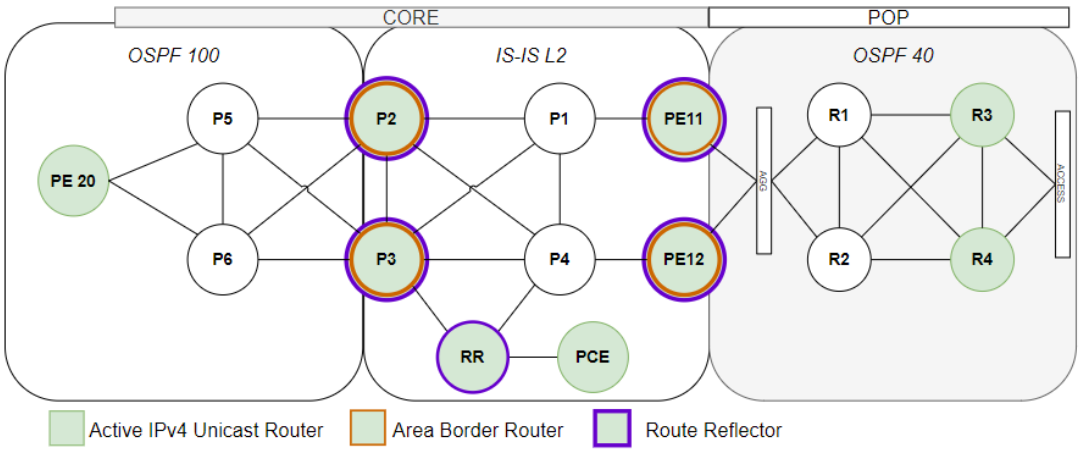
```
router ospf 100
  router-id 10.65.100.201

interface Ethernet1/1
  no switchport
  medium p2p
  ip unnumbered loopback0
  ip ospf network point-to-point
  ip router ospf 100 area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

## BGP

It can be argued that BGP will act as both an under and overlay within this network. For this reason, it will be divided into two separate sections. This section will review the underlay needs while services will be discussed later.

### IPv4 Unicast:

The IPv4 Unicast family will tie together the entire infrastructure. The first use case we will explore is end-to-end reachability throughout the core and POP locations.




IPv4 Unicast Routers

The diagram above displays the routers participating in the IPv4 Unicast address family. The routers acting as an area border router and route reflector perform next-hop-self operations between the domains. In the core layer, the designated route reflector is the relay point for P2, P3, PE11, and PE12. The ABR/RR router acts as the relay point within the OSPF domain. Through the advertisement of loopback interfaces, an operator can achieve end-to-end connectivity (Segment Routing/LDP needs to be used for data plane forwarding). A quick traceroute from PE20 to R4 will demonstrate this ability.

```
traceroute 10.40.4.4 source loopback 0 probe 1 numeric

 1  10.20.6.6 [MPLS: Label 16003 Exp 0] 9 msec
 2  10.5.2.2 8 msec
 3  10.3.1.1 [MPLS: Label 16011 Exp 0] 20 msec
 4  10.11.1.11 13 msec
 5   *
 6  10.4.1.4 17 msec
```

Traceroute between PE20 and R4

As displayed the traceroute probe can navigate all three domains by utilizing a single label towards the ABR. The ABR will then reference its forwarding table, which has entries for both domains. If necessary, the final ABR will place the final label, and traffic will terminate at the correct endpoint. This document will now follow the BGP hops taken in the previous traceroute and explore each device's BGP configuration. The block below will display the BGP configuration present at PE20.

```
#PE20 Configuration

router bgp 10200
 bgp router-id 20.20.20.20
 address-family ipv4 unicast
  additional-paths receive
  maximum-paths ibgp 6
  network 20.20.20.20/32
 !
 neighbor 2.2.2.2
  remote-as 10200
  update-source Loopback0
  address-family ipv4 unicast
   next-hop-self
  !
 !
 neighbor 3.3.3.3
  remote-as 10200
  update-source Loopback0
  address-family ipv4 unicast
   next-hop-self
  !
```

PE20 operates without much configuration. PE20 will act as a basic PE router in this state, only requiring a next-hop change for eBGP learned routes before forwarding. Moving onto an ABR router, the configuration becomes more involved as the router's responsibility increases.

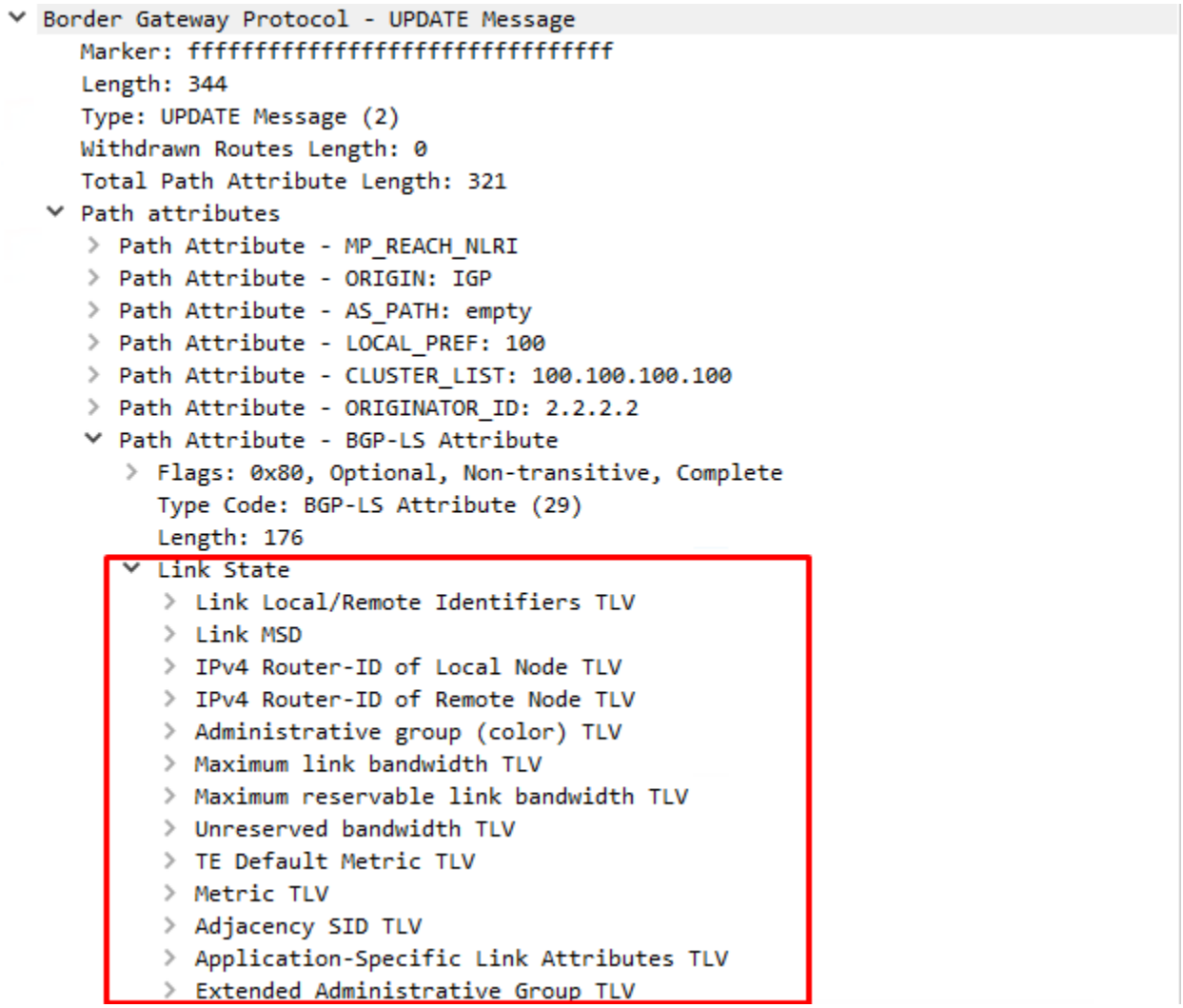
```
#P2 ABR Configuration

router bgp 10200
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy ADDPATH
 !
 !
 neighbor 20.20.20.20
  remote-as 10200
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
   next-hop-self
 !
 neighbor 100.100.100.100
  remote-as 10200
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
   next-hop-self
  !
  address-family link-state link-state
   route-reflector-client
  !
 !
!
```

ABR configuration

The ABR must act as route reflectors to reflect routes into their domain. The route reflector distinction overrides the iBGP loop prevention behavior. Also, the device must change the next hop address due to the separated domains. In IOS-XR, the configuration must include the *ibgp policy out enforce-modifications* for the next-hop-self command to have any impact on iBGP routes.

**BGP Link-State**

While segment routing has yet to be discussed, understand that link-state information must be shared between the domains to control traffic engineering paths. BGP has been adapted to share this information between domain boundaries. This new address family is known as BGP Link State (BGP-LS) AFI 16388, SAFI 71. The link state updates are robust and provide all the necessary details for a traffic controller to map each domain.




Wireshark capture of BGP-LS UPDATE message

Router configuration is minimal to share link-state information. There are two contact points, one in the IGP configuration and another in adding the address family to the appropriate neighbors within the BGP configuration.

```
router isis LAB
 net 49.0000.0000.0000.0002.00
 distribute link-state instance-id 200
 address-family ipv4 unicast

router ospf 100
 distribute link-state instance-id 100

router ospf 40
 distribute link-state instance-id 40
```

IGP link-state distribution configuration

Each IGP process will be redistributed with an instance number. These numbers need to be globally unique.

```
address-family link-state link-state

neighbor 100.100.100.100
  remote-as 10200
  update-source Loopback0
  !
  address-family link-state link-state
```

BGP link-state configuration

In the lab network, the controller that will utilize this information resides within the core module. This allows the ABR to easily share the information for all domains without any involved routing configuration. For the information to reach its final destination, the route reflector will peer with the controller utilizing the BGP-LS family. Reviewing the BGP summary command, the controller has received 98 entries that have been reflected.

```
RP/0/RP0/CPU0:10200-PCE#sh bgp link-state link-state summary | b Neigh
Thu Nov  2 05:34:57.998 UTC
Neighbor     Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
100.100.100.100   0 10200 9173   7767   2507   0      0      5d08h         98
```

BGP link-state summary

The BGP Link-State table is a busy output. Still, it provides the basic information necessary to verify that the links are properly distributed.

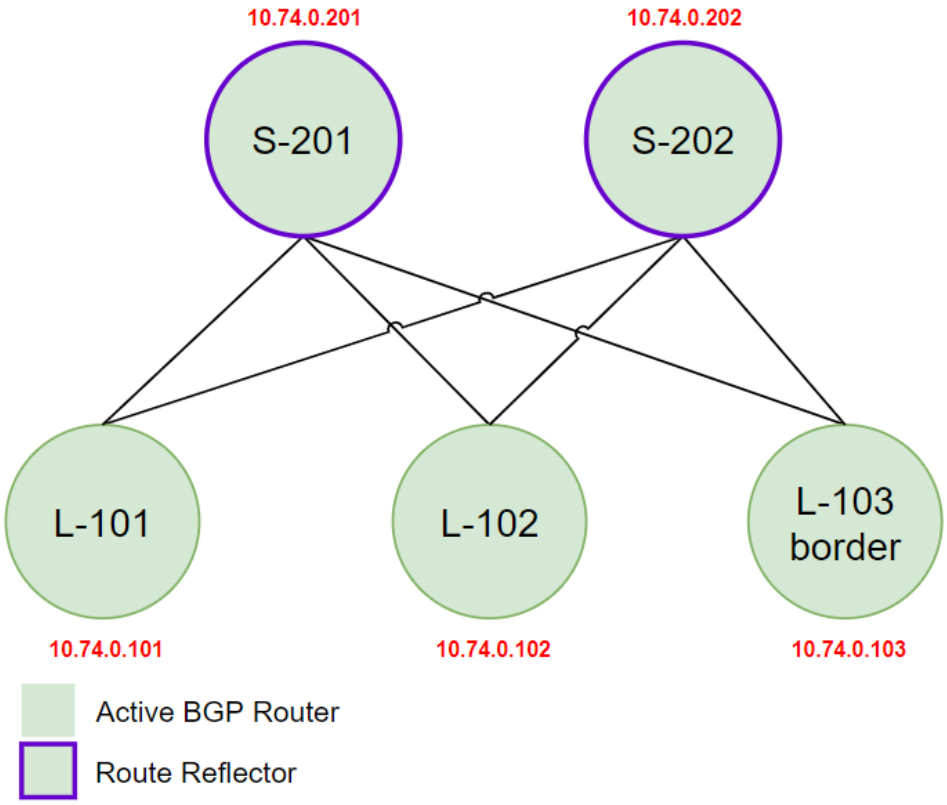
```
RP/0/RP0/CPU0:10200-PCE#sh bgp link-state link-state | b Stat
Thu Nov  2 06:12:15.660 UTC
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Prefix codes: E link, V node, T IP reacheable route, S SRv6 SID, u/U unknown
              I Identifier, N local node, R remote node, L link, P prefix, S SID
              L1/L2 ISIS level-1/level-2, O OSPF, D direct, S static/peer-node
              a area-ID, l link-ID, t topology-ID, s ISO-ID,
              c confed-ID/ASN, b bgp-identifier, r router-ID, s SID
              i if-address, n nbr-address, o OSPF Route-type, p IP-prefix
              d designated router address
   Network            Next Hop            Metric LocPrf Weight Path
*>i[V][L1][IOxc8][N[c10200][b0.0.0.0][s0000.0000.0002.00]]/328
                     2.2.2.2                              100      0 i
*>i[V][L2][IOxc8][N[c10200][b0.0.0.0][s0000.0000.0001.00]]/328
                     2.2.2.2                              100      0 i
*>i[V][L2][IOxc8][N[c10200][b0.0.0.0][s0000.0000.0002.00]]/328
                     2.2.2.2                              100      0 i
```

BGP link-state table

The information shared will become a crucial component allowing computation of end-to-end paths across separate IGP domains. The table contains all links, nodes, and IP reachability information, allowing the controller to understand all available paths.

**Data Center BGP**

Within the data center module, the BGP implementation utilizes the address family of L2VPN EVPN.




Data center topology

The objective of the data center module is to create a fabric. In this deployment, the route reflectors peer with all three leaves, sharing all routes. The peering allows the devices to act as one large switch. The L2VPN EVPN address family allows seamless layer 2 forwarding behavior over the layer 3 underlay. This is known as a spine-leaf architecture. The origin of this architecture comes from a telephone-switching layout developed by Charles Clos. The high-level objective is multiple non-blocking links with predictable traffic patterns. A separate loopback address is used for BGP configuration in this environment.  This is due to VXLAN behavior that would cause BGP to go down if the VXLAN tunnel endpoint shared the same loopback. Also, enabling the correct features to configure the device properly is essential with Cisco Nexus devices. Below is the needed feature(s) and the initial spine configuration.

```
###Features for all nodes
feature ospf
feature bgp
feature pim
feature nv overlay
nv overlay evpn

###Leaf Only
feature fabric forwarding
feature interface-vlan
feature vn-segment-vlan-based

#S201 Spine BGP Configuration
router bgp 65500
  neighbor 10.74.0.101
      remote-as 65500
      update-source loopback74
      address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
  neighbor 10.74.0.102
      remote-as 65500
      update-source loopback74
      address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
  neighbor 10.74.0.103
      remote-as 65500
      update-source loopback74
```

```
      address-family l2vpn evpn
      send-community
      send-community extended
      route-reflector-client
```

Feature and spine BGP configuration

The spine accepts and reflects routes from every leaf peer. Extended communities are necessary to interpret the L2VPN NLRI correctly. The leaf unit configuration is similar to that of the spines. It creates a full mesh with the route-reflectors.

```
router bgp 65500
  neighbor 10.74.0.201
      remote-as 65500
      update-source loopback74
      address-family ipv4 unicast
      address-family l2vpn evpn
      send-community
      send-community extended
  neighbor 10.74.0.202
      remote-as 65500
      update-source loopback74
      address-family ipv4 unicast
      address-family l2vpn evpn
      send-community
      send-community extended
```

Leaf BGP configuration

The data center underlay network is now fit for overlay installation which will be discussed in later sections.

## Segment Routing (SR)

Segment Routing(SR) allows separate domains to behave as a single network. SR is the necessary evolution in forwarding that simplifies both function and configuration. Previously, BGP-free core implementations were configured using an IGP with LDP, creating two separate configuration points. Traffic engineering needs were handled by RSVP, which further complicated configurations. SR aims to simplify the operation by bundling label-forwarding, traffic engineering, and path repair capability within the IGP.

The SR configuration can be placed in a few lines. The first necessary configuration within this specific lab will be to carve an explicit label block for SR. The explicit block is required due to the inter-domain nature. This is recognized as the default value in a single domain, but it is best to define ranges explicitly when considering multi-domain.

```
segment-routing
 global-block 16000 23999
```

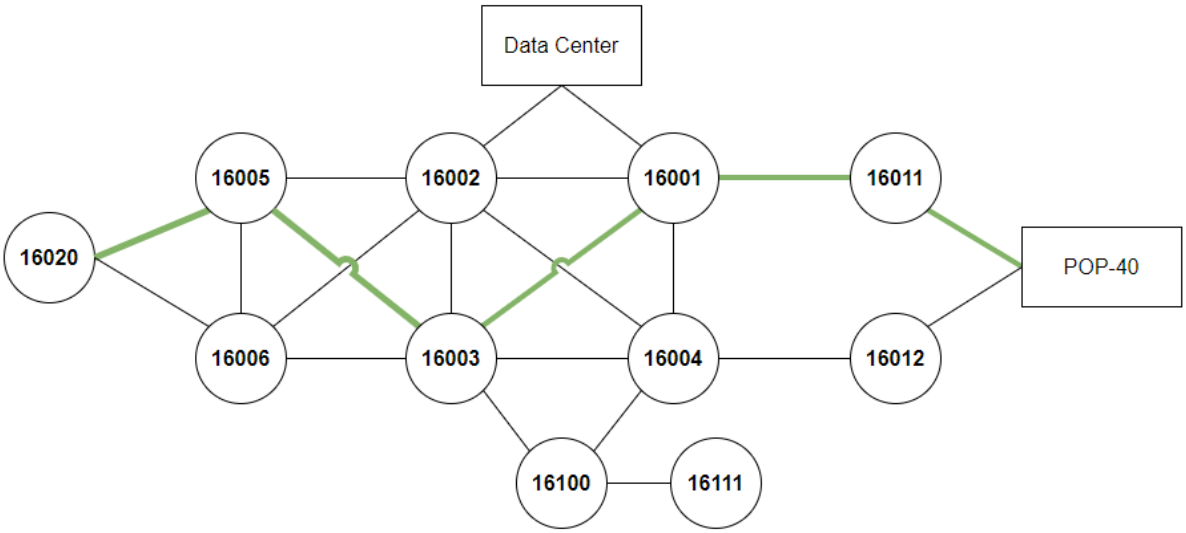
SR label block configuration

Once the block is allocated, IS-IS and OSPF can be configured to enable segment routing extensions.

```
#PE11 Configuration
router isis LAB
 address-family ipv4 unicast
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 11

router ospf 40
 segment-routing mpls
 segment-routing sr-prefer
 area 0
  interface Loopback0
   prefix-sid index 11
```

ISIS SR configuration

Under the interface configuration, a prefix-sid is referenced; this is the node identifier for SR. This is formally known as a node segment identifier (node SID). This SID needs to be globally significant as it acts as the unique ID for said node. The index specification is the node's position within the global block. Index 11 = 16000+11, resulting in a node side for PE11 of 16011.




Example Node SID Path

When bringing router identity to a simple numeric value, it creates an easy-to-grasp infrastructure when considering traffic engineer needs, but it also simplifies the troubleshooting of labeled paths. SR can also tune forwarding down to a specific interface. This forwarding behavior utilizes another label block, the Segment Routing Local Block (SRLB). The router implementation gets convoluted as adjacency sids are allotted automatically and use the dynamic label range of 24000-1048575. This presents an issue when referencing these labels for traffic engineering policy, as they will not persist through reloads. A more sustainable method is manually assigning adjacency labels where granular link use is needed. A manually set label is known and will persist through device events such as reloads. The SRLB is provisioned with an automatic label range of 15000-15999. SID assignment is performed under the specific IGP.

```
router ospf 100
 area 0
  !
  interface GigabitEthernet0/0/0/1
   adjacency-sid index 1
```

Adjacency SID Configuration

```
RP/0/RP0/CPU0:10200-P6#sh mpls forwarding | ex Pfx
Thu Nov  2 12:54:29.282 UTC
Local  Outgoing    Prefix           Outgoing     Next Hop        Bytes
Label  Label       or ID            Interface                    Switched
------ ----------- ---------------- ------------ --------------- ------------
15001  Pop         SRLB (idx 1)     Gi0/0/0/1    10.6.5.5        0
24000  Pop         SR Adj (idx 0)   Gi0/0/0/5    10.6.2.2        0
24001  Pop         SR Adj (idx 0)   Gi0/0/0/6    10.6.3.3        0
24002  Pop         SR Adj (idx 0)   Gi0/0/0/1    10.6.5.5        0
24003  Pop         SR Adj (idx 0)   Gi0/0/0/2    10.20.6.20      0
RP/0/RP0/CPU0:10200-P6#
```

Adjacency-sid verification

Viewing the mpls forwarding table the manually configured SID and the dynamic allocation are both visible for Gi0/0/0/1. The static label can now be called in a traffic engineering policy, forcing the applicable traffic through the specified link. The adjacency SID will also assume a significant role when implementing fast reroute, which will be discussed in future sections. With the basic configuration demonstrated in this section, the network should have end-to-end reachability through SR labels. A traceroute from PE11 to PE20 will verify this claim.

```
RP/0/RP0/CPU0:10200-PE11#traceroute  20.20.20.20 source loopback 0 numeric
probe 1
Thu Nov  2 13:43:55.158 UTC

Type escape sequence to abort.
Tracing the route to 20.20.20.20

 1  10.11.1.1 [MPLS: Label 16003 Exp 0] 17 msec
 2  10.3.1.3 7 msec
 3  10.6.3.6 [MPLS: Label 16020 Exp 0] 49 msec
 4  10.20.6.20 10 msec
```

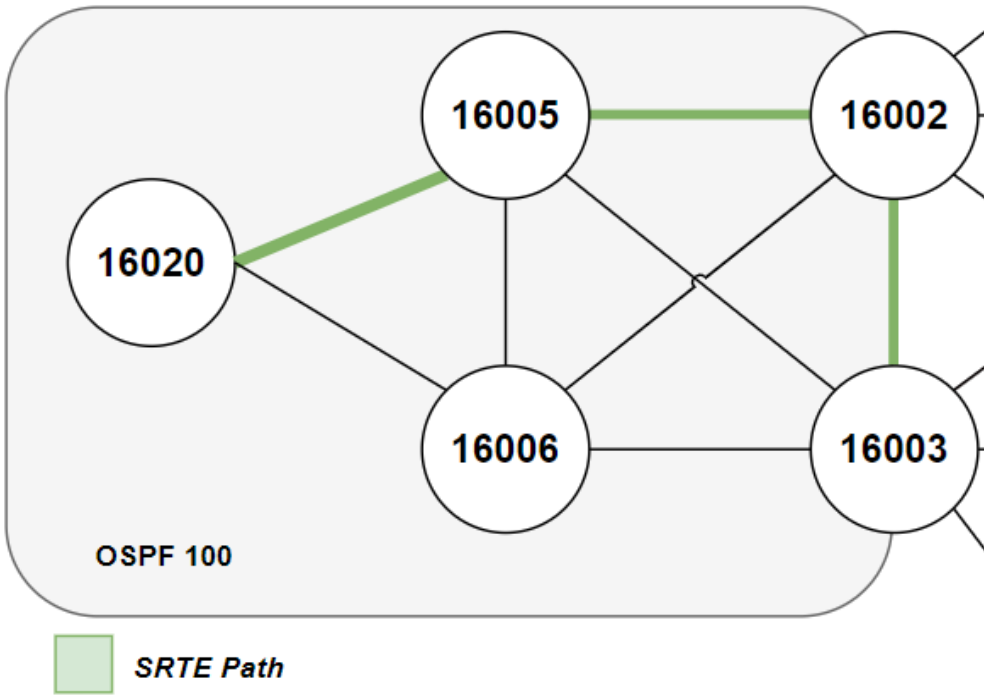
Traceroute with SR (PE11 to PE20)

Referring back to the node side map presented earlier in this section you can see the path imposes the ABR prefix-sid. Upon reception of that initial packet, the ABR performs a lookup for destination 20.20.20.20, which returns the node sid of PE20 (16020). This label is then imposed and directed out the correct interface toward the destination.

Currently, the traffic successfully reaches its destination, but the operator has no granular path control. The network only forwards to the border nodes where the new label is imposed, and the best IGP path is taken. Traffic Engineering will provide the necessary tools to configure granular label paths.

**Traffic Engineering**

Traffic engineering is not a new concept. This has previously been handled through the combination of LDP and RSVP. Segment Routing has again provided new options and methods for granular traffic control with less configuration overhead. For intra and interdomain traffic, Segment Routing Traffic Engineering (SRTE) will utilize policy configuration to enforce the desired path.




Example SRTE path

Focusing on the simple path depicted in the diagram above, a policy can be constructed to force all traffic with a next hop of 3.3.3.3 over the specific path. The policy will consist of an explicit path list directing all traffic over the transport path of P5-P2-P3.

```
segment-list 20-5-2-3
  index 1 mpls label 16005
  index 2 mpls label 16002
  index 3 mpls label 16003
 !
 policy 20_to_3
  color 3 end-point ipv4 3.3.3.3
  autoroute
     include ipv4 all
  !
  candidate-paths
     preference 100
     explicit segment-list 20-5-2-3
```

Explicit segment list configuration

The first configuration is the explicit path, which the SRTE policy will reference. In this path configuration, the indexed instructions will utilize MPLS label values. The label stack imposed

will mirror the explicit path (The first hop will resolve, meaning the 16005 will not be placed, but traffic will be directed out the interface towards P5 with the next label for forwarding.) The policy is given a simple name, and the router will also create its own internal name based on the color and endpoint values. The color value is used to identify traffic intended for specific paths. For instance, multiple tunnels may exist toward node 3.3.3.3, and an operator could classify traffic color to utilize a particular tunnel toward the endpoint (e.g., Color X is designated to tunnel one while Color Y uses tunnel two). The endpoint is the IP value of the tunnel-terminating endpoint. The autoroute configuration forces applicable traffic down the policy-defined path. In the case of this intra-domain example, only 3.3.3.3 will be reachable via the tunnel. Autoroute is only applicable for IGP prefixes. Before committing the policy a traceroute can prove that the traffic is still taking the preferred IGP path.

```
RP/0/RP0/CPU0:10200-PE20#traceroute 3.3.3.3 so lo 0 probe 1 numeric
Thu Nov  2 14:51:02.024 UTC

Type escape sequence to abort.
Tracing the route to 3.3.3.3

 1  10.20.6.6 [MPLS: Label 16003 Exp 0] 8 msec
 2  10.5.3.3 8 msec
```

Non-SRTE traceroute

Upon activation, the new path can be observed. Also, the routing table will now display the SRTE policy as the next hop for traffic destined to 3.3.3.3.

```
RP/0/RP0/CPU0:10200-PE20#traceroute 3.3.3.3 so lo 0 probe 1 numeric
Thu Nov  2 14:53:08.961 UTC

Type escape sequence to abort.
Tracing the route to 3.3.3.3

 1  10.20.5.5 [MPLS: Labels 16002/16003 Exp 0] 11 msec
 2  10.5.2.2 [MPLS: Label 16003 Exp 0] 17 msec
 3  10.6.2.6 [MPLS: Label 16003 Exp 0] 9 msec
 4  10.6.3.3 20 msec
```

SRTE enabled traceroute

```
RP/0/RP0/CPU0:10200-PE20#sh ip route 3.3.3.3
Thu Nov  2 14:54:02.103 UTC
Routing entry for 3.3.3.3/32
  Known via "ospf 100", distance 110, metric 3, labeled SR, label redist
non FIB, type intra area
  Installed Nov  2 14:53:06.021 for 00:00:56
  Routing Descriptor Blocks
      3.3.3.3, from 3.3.3.3, via srte_c_3_ep_3.3.3.3
      Route metric is 3
  No advertising protos.
```

SRTE entry for the route to 3.3.3.3

The command *show segment-routing traffic-engineering policy color X* can be run to view details and policy status. From this output, an operator can assess the state of the policy and the current path in use.

```
RP/0/RP0/CPU0:10200-PE20#sh segment-routing traffic-eng policy color 3
Thu Nov  2 14:57:42.376 UTC

SR-TE policy database
---------------------

Color: 3, End-point: 3.3.3.3
  Name: srte_c_3_ep_3.3.3.3
  Status:
    Admin: up  Operational: up for 00:04:36 (since Nov  2 14:53:05.949)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: 20_to_3
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Explicit: segment-list 20-5-2-3 (valid)
        Weight: 1, Metric Type: TE
          16005
          16002
          16003
  Attributes:
    Binding SID: 24003
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
```

SRTE policy information

The preceding intra-area demonstration simply represented the specificity achieved when utilizing SRTE. Many additional options will be assessed in future sections.

**Controller Integration (PCE)**

When considering inter-domain traffic engineering options, a central device is needed to collect and act upon the required information. This is known as a Path Computational Element (PCE) in the SR realm. Within this lab, an additional IOS-XR node will operate as the PCE. Edge routers peer with the PCE node to send/receive the necessary information for policy creation. The nodes that peer with the PCE are called Path Computational Clients (PCC). The communication between the PCE and PCC utilizes a specific protocol: Path Computational Element Protocol (PCEP). The PCE uses the information obtained from the BGP link-state address family to calculate paths across domain boundaries. The centralized controller simplifies configuration and allows traffic changes to be implemented quickly. The initial peering configuration is less than ten lines across a peering pair.

```
#PCE node configuration
pce
 address ipv4 111.111.111.111

#PCC node configuration (PE11)
segment-routing
 traffic-eng
  pcc
   source-address ipv4 11.11.11.11
   pce address ipv4 111.111.111.111
      precedence 100
```

PCE and PCC configuration

With peering configuration in place, PCE details can be verified.

```
RP/0/RP0/CPU0:10200-PCE#show pce ipv4 peer
Thu Nov  2 20:14:54.563 UTC

PCE's peer database:
--------------------
Peer address: 10.40.3.3
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation

Peer address: 10.40.4.4
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation

Peer address: 11.11.11.11
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6

Peer address: 20.20.20.20
  State: Up
  Capabilities: Stateful, Segment-Routing, Update, Instantiation, SRv6
```

Peer routers, as seen by the PCE

Configuration of the BGP Link State address family provides the PCE node with a rich database of information that can be utilized to delegate SRTE policies based on various specifications. An onboard demonstration can help showcase this ability by forcing the PCE to calculate a path based on known values.

```
#show pce ipv4 cspf-sr-mpls source 11.11.11.11 destination 20.20.20.20
metric-type latency
Thu Nov  2 20:18:32.245 UTC

Computation Algorithm: Dijkstra, Duration: 1 msec
Path[0], from 11.11.11.11 to 20.20.20.20, cost=22:
Hop0: 16002 Node (2.2.2.2)
Hop1: 16005 Node (5.5.5.5)
Hop2: 16020 Node (20.20.20.20)
```

PCE path computation demonstration

In the example above, the PCE could examine all available paths between two nodes existing in separate domains and determine the optimal path based on latency. The *cspf-sr-mpls* option in the command specifies that the requested calculation should be performed with a constrained shortest path first SR computation. To push this policy from the PCE to the PCC (PE11), the

configuration needed at the PCE node is similar to the manually configured policy demonstrated in the previous section.

```
Pce
 segment-routing
  traffic-eng
   peer ipv4 11.11.11.11
        policy 11_20_Latency
        color 100 end-point ipv4 20.20.20.20
        candidate-paths
        preference 100
        dynamic mpls
        metric
        type latency
```

PCE based policy configuration

All traffic engineering configurations conducted on the PCE node will be placed under the top-level PCE configuration mode. The PCE node must specify a peer target to know where to send the specific configured policies. Like the manual policy, a name, color, and endpoint must be set for the policy to be deemed valid.  Under the candidate paths the dynamic path type is selected. This allows the PCE to choose the path that best matches the metric constraint dynamically. In this case, the PCE has considered all paths and selects the option with the lowest latency. If this path fails, the PCE will assess the remaining paths and determine a new low-latency option. Reviewing the SRTE policies at PE11 reveals that the policy has successfully been pushed from the PCE.

```
RP/0/RP0/CPU0:10200-PE11#sh segment-routing traffic-eng policy color 100
Thu Nov  2 20:44:16.093 UTC

SR-TE policy database
---------------------

Color: 100, End-point: 20.20.20.20
  Name: srte_c_100_ep_20.20.20.20
  Status:
      Admin: up  Operational: up for 00:16:25 (since Nov  2 20:27:50.626)
  Candidate-paths:
      Preference: 100 (PCEP) (active)
      Name: 11_20_Latency
      Requested BSID: dynamic
      PCC info:
      Symbolic name: 11_20_Latency
      PLSP-ID: 3
      Constraints:
      Protection Type: protected-preferred
```

```
    Maximum SID Depth: 10
    Dynamic (pce 111.111.111.111) (valid)
    Metric Type: LATENCY,   Path Accumulated Metric: 22
          16002 [Prefix-SID, 2.2.2.2]
          16005 [Prefix-SID, 5.5.5.5]
          16020 [Prefix-SID, 20.20.20.20]
Attributes:
    Binding SID: 24004
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
```

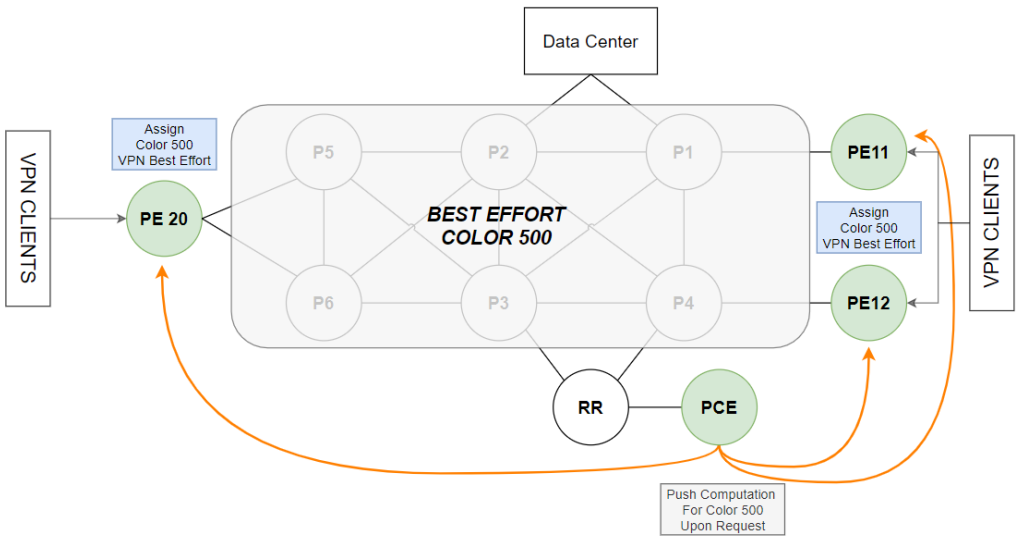
PCE originated policy on the peer node

## Traffic Steering

SRTE traffic steering can be handled in various ways. While this document will aim to cover most use cases, the options continue to grow. Within the Route 109 network, both automated and manual traffic steering options will be utilized and documented.

## On-Demand Nexthop

On-demand Nexthop (ODN) allows setting various policies that will be instantiated and torn down as needed. A multi-domain environment, such as in this lab, simplifies operations by enabling the PCE to handle the necessary stitching of labels while offloading computation responsibility to a centralized device.




VPN customer best effort color deployment

The diagram above demonstrates the high-level overview of the ODN process. L3VPN client routes are tagged with a specific color upon ingress. These routes are then reflected to other applicable provider edge nodes. If an existing on-demand policy is present, the traffic will be steered into said policy based on the color value. The computation for this policy can be performed locally or via the controller. Within this multi-domain lab, this will be handled by the PCE. The configuration on the edge nodes is minimal and delegates control of the path to the PCE.

```
segment-routing
 traffic-eng
  on-demand color 500
   dynamic
      pcep
```

PCC on-demand color configuration

With the current setup, any traffic received possessing the extended community color of 500 will trigger a request toward the PCE to compute the necessary path. With no constraints, the path returned will reflect the existing IGP path through each domain. Reviewing a route received via VPNv4, the color 500 has triggered the PCE computation. The PCE extracts the next-hop from the BGP route and calculates a policy for said node.

```
RP/0/RP0/CPU0:10200-PE11#sh bgp vrf VPN 192.168.100.20
Sat Nov  4 19:41:26.633 UTC
BGP routing table entry for 192.168.100.20/32, Route Distinguisher: 11:100
Versions:
  Process         bRIB/RIB  SendTblVer
  Speaker              30          30
Last Modified: Nov  4 18:57:37.216 for 00:43:49
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
      20.20.20.20 C:500 (bsid:24005) (metric 20) from 20.20.20.20
(20.20.20.20)
      Received Label 24001
      Origin IGP, metric 0, localpref 100, valid, internal, best,
group-best, import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 30
```

```
      Extended community: Color:500 RT:10200:100
      SR policy color 500, up, registered, bsid 24005, if-handle 0x00000024
```

VPNv4 route recieved with extcommunity value of 500

```
RP/0/RP0/CPU0:10200-PE11#sh segment-routing traffic-eng policy  color 500
Sat Nov  4 19:43:35.661 UTC

SR-TE policy database
---------------------

Color: 500, End-point: 20.20.20.20
  Name: srte_c_500_ep_20.20.20.20
  Status:
      Admin: up  Operational: up for 00:46:01 (since Nov  4 18:57:34.050)
  Candidate-paths:
      Preference: 200 (BGP ODN) (inactive) (shutdown)
      Requested BSID: dynamic
      Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 10
      Dynamic (inactive)
      Metric Type: TE,   Path Accumulated Metric: 0
      Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
      PCC info:
      Symbolic name: bgp_c_500_ep_20.20.20.20_discr_100
      PLSP-ID: 1
      Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 10
      Dynamic (pce 111.111.111.111) (valid)
      Metric Type: TE,   Path Accumulated Metric: 22
            16002 [Prefix-SID, 2.2.2.2]
            16020 [Prefix-SID, 20.20.20.20]
  Attributes:
      Binding SID: 24005
      Forward Class: Not Configured
      Steering labeled-services disabled: no
      Steering BGP disabled: no
      IPv6 caps enable: yes
      Invalidation drop enabled: no
```

Policy Received from PCE

The traceroute demonstration verified the label stack defined by the PCE had been imposed with the VPNv4 label residing at the bottom of the stack.

```
RP/0/RP0/CPU0:10200-PE11#traceroute vrf VPN 192.168.100.20 source loopback
100
Sat Nov  4 19:45:46.738 UTC

Type escape sequence to abort.
Tracing the route to 192.168.100.20

 1  10.11.1.1 [MPLS: Labels 16002/16020/24001 Exp 0] 39 msec  5 msec  11
msec
 2  10.2.1.2 [MPLS: Labels 16020/24001 Exp 0] 32 msec  6 msec  13 msec
 3  10.5.2.5 [MPLS: Labels 16020/24001 Exp 0] 55 msec  7 msec  11 msec
 4  10.20.5.20 10 msec  *  6 msec
```

Multi-domain traceroute with proper label stack defined by the PCE computation

A simple change to the policy could also provide a dynamic low latency path for customers who decide to purchase the service. Adding the color value of 600 will trigger a computation request for a low-latency path toward the PCEP.

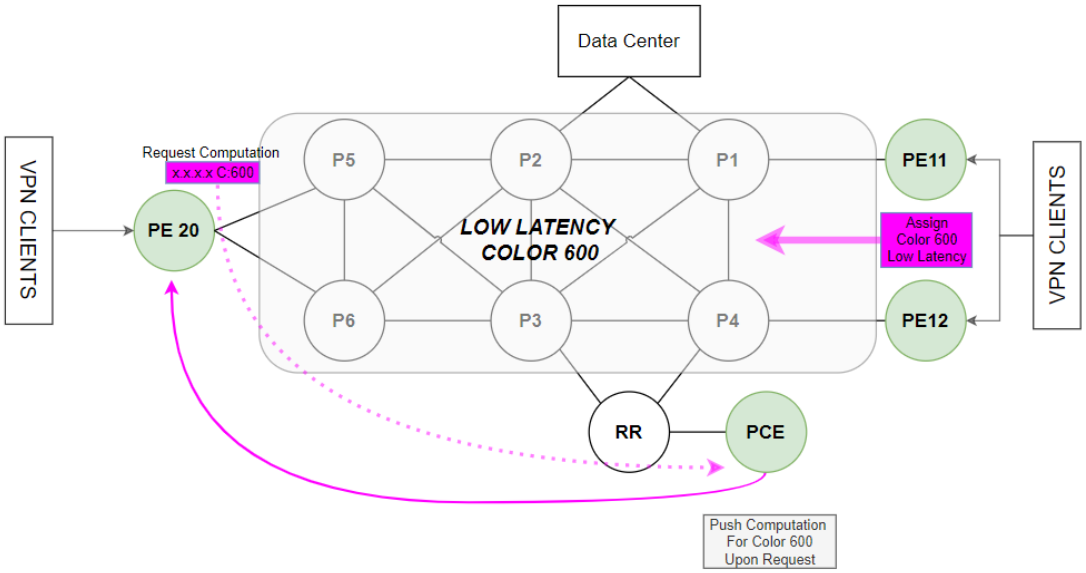
```
on-demand color 600
   dynamic
      pcep
      !
      metric
      type latency
```

IOS-XR On Demand Policy Latency

The color distinction is handled via VRF export on the IOS-XR edge node. A route policy specifies an extended community list for route export. The route reflector reflects the route with the color value to other edge nodes.

```
extcommunity-set opaque VPN-LL
  600
end-set

route-policy VPN-LL
  set extcommunity color VPN-LL
End-policy

vrf CXw-24
 address-family ipv4 unicast
  import route-target
   10200:24
  export route-policy VPN-LL
  export route-target   10200:24
```

IOS-XR router policy and VRF export configuration

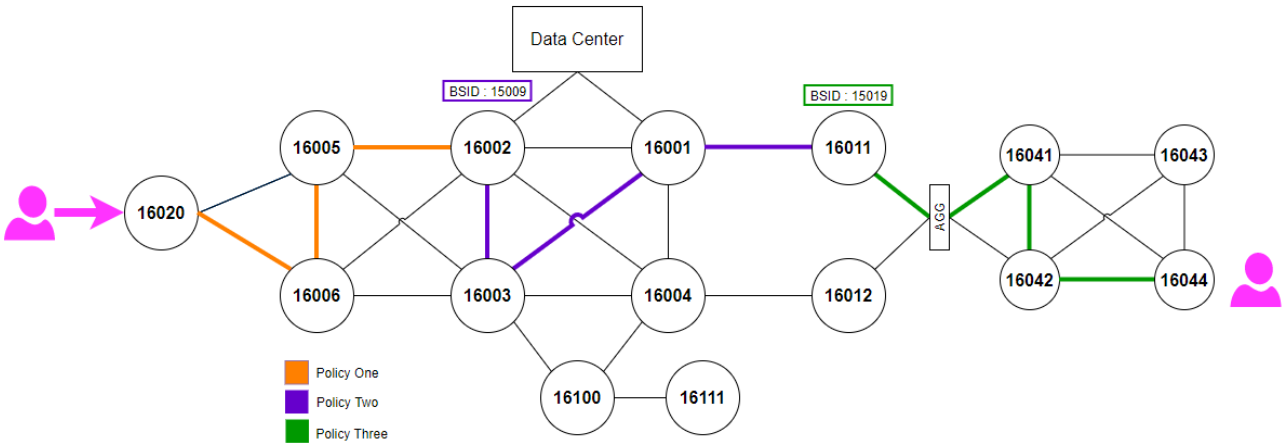



Low Latency Color Group Flow

With minimal configuration operators can maintain various traffic groups providing specific behavior and constraints through the use of color distinctions and PCE driven policy. On-demand next hop provides a simplified approach to scale with network/customer growth.

## Binding SID Stitching

Binding SIDs (BSIDS) can steer traffic across domains with absolute control. Beyond multi-domain control ability, this practice can also add label hierarchy to large single-domain deployments. SID stitching will allow traffic from one policy to be immediately placed into another. Each policy within the chain will enable operators to influence traffic in any way they see fit to meet requirements.




BSID Path Overview

The diagram above displays the separation between policies. The explicit BSID value is pulled from the Segment Routing Local Block (15,000 - 15,999). Beginning with Policy-ONE, an explicit segment list will direct traffic over the desired path. The final label within the explicit list will be the BSID value of Policy-TWO. This instruction will cause the traffic to be steered into the following policy.

```
#PE20 16020
segment-list LIST-POL-1
   index 1 mpls label 16006
   index 2 mpls label 16005
   index 3 mpls label 16002
   index 4 mpls label 15009

policy Policy-ONE
   color 700 end-point ipv4 2.2.2.2
   candidate-paths
      preference 100
      explicit segment-list LIST-POL-1


#P2 16002 (ABR)
segment-list LIST-POL-2
   index 1 mpls label 16003
   index 2 mpls label 16001
   index 3 mpls label 16011
   index 4 mpls label 15019

  policy Policy-TWO
   binding-sid mpls 15009
   color 700 end-point ipv4 11.11.11.11
   candidate-paths
      preference 100
      explicit segment-list LIST-POL-2

#PE11 16011
  segment-list LIST-POL-3
   index 1 mpls label 16041
   index 2 mpls label 16042
   index 3 mpls label 16044

  policy Policy-THREE
   binding-sid mpls 15019
   color 700 end-point ipv4 10.40.4.4
   candidate-paths
      preference 100
      explicit segment-list LIST-POL-3
```

BSID policy configuration across domains

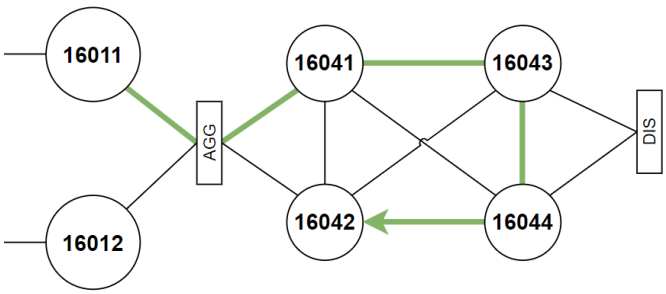
Through the use of a traceroute the explicit path can be visualized. The three separate policy paths have been highlighted in the code block below. The bottom label signaled via the VPNv4 peerings remains unchanged as the traffic is sent through multiple policies.

```
Policy-ONE | Policy-TWO | Policy-THREE
RP/0/RP0/CPU0:10200-PE20#traceroute vrf CXw-24 24.24.24.40 source lo24
Tue Nov  7 12:20:59.466 UTC
Tracing the route to 24.24.24.40

 1  10.20.6.6 [MPLS: Labels 16005/16002/15009/24003 Exp 0]
 2  10.6.5.5 [MPLS: Labels 16002/15009/24003 Exp 0]
 3  10.5.2.2 [MPLS: Labels 15009/24003 Exp 0]
 4  10.3.2.3 [MPLS: Labels 16001/16011/15019/24003 Exp 0]
 5  10.3.1.1 [MPLS: Labels 16011/15019/24003 Exp 0]
 6  10.11.1.11 [MPLS: Labels 15019/24003 Exp 0]
 7  10.40.11.40 [MPLS: Labels 16042/16044/24003 Exp 0]
 8  10.2.1.2 [MPLS: Labels 16044/24003 Exp 0]
 9  10.4.2.4 [MPLS: Label 24003 Exp 0]
 10 192.168.44.0 4 msec
```

Another benefit regarding the use of stitching policies is the control of the label stack depth. Using multiple policies, the label stack is constantly refreshed along the end-to-end path and never gets larger than four labels deep. Without the BSID, the label stack would possess seven separate values at ingress.

**Static Route SRTE**

While static routing presents tradeoffs and increased administrative overhead, it still fulfills a need within various networks. SRTE policy can be called in static route configuration to force traffic through a specific policy.




Static route SRTE path

To quickly demonstrate this method, a policy will be created that diverts all traffic from PE11 to POP-R2 in a ring pattern. This will be configured via an explicit segment list.

```
segment-list RING
   index 1 mpls label 16041
   index 2 mpls label 16043
   index 3 mpls label 16044
   index 4 mpls label 16042

 policy POL-RING
   color 8 end-point ipv4 10.40.2.2
   candidate-paths
      preference 100
      explicit segment-list RING

router static
 address-family ipv4 unicast
 10.40.2.2/32 sr-policy srte_c_8_ep_10.40.2.2
```

Static route SRTE policy configuration

The next-hop in the routing table now points directly to the SRTE policy and a traceroute will verify that the traffic is flowing as intended.

```
RP/0/RP0/CPU0:10200-PE11#sh ip route 10.40.2.2
Tue Nov  7 20:16:10.270 UTC

Routing entry for 10.40.2.2/32
  Known via "static", distance 1, metric 0
  Installed Nov  7 20:14:08.258 for 00:02:02
  Routing Descriptor Blocks
      directly connected, via srte_c_8_ep_10.40.2.2
      Route metric is 0, Wt is 1
  No advertising protos.

RP/0/RP0/CPU0:10200-PE11#traceroute 10.40.2.2 source loopback 0

 1  10.40.11.40 [MPLS: Labels 16043/16044/16042 Exp
 2  10.3.1.3 [MPLS: Labels 16044/16042 Exp 0]
 3  10.4.3.4 [MPLS: Label 16042 Exp 0]
 4  10.4.2.2 9 msec  *  4 msec
```

Table and trace verification of SRTE static route

Again. At the same time, static routes are not an ideal option they are also present in most networking environments and the ability to add traffic engineering policy provides welcomed flexibility.

**Summary**

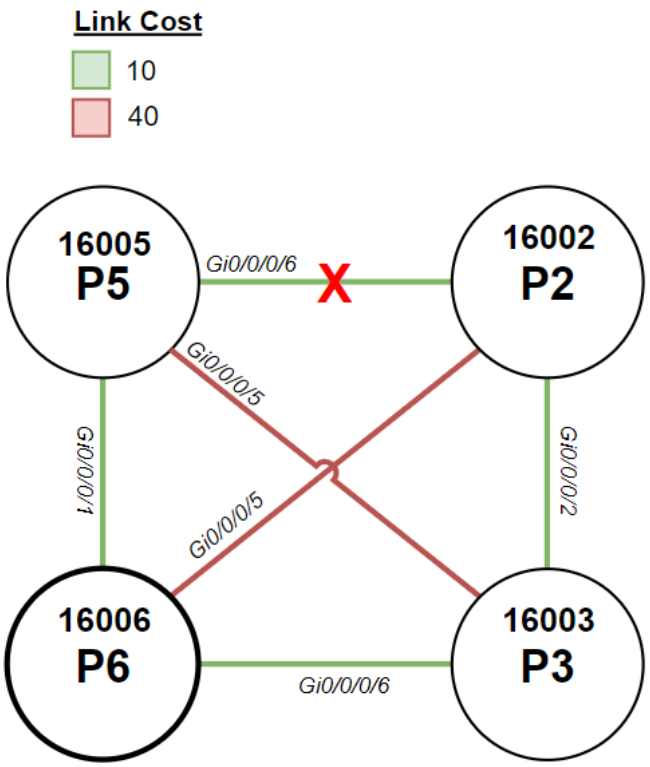
This section served as the basis for all references going forward regarding SRTE policy creation and manipulation. SRTE is a broad topic that has many use cases. The previous demonstration provided a basic understanding of the technology's ability.

# TI-LFA

Topology Independent Loop-Free Alternate replaces the previous technologies Loop-Free Alternate (LFA) and Remote Loop-Free Alternate (RLFA). The high-level objective of these technologies is to provide the rapid repair of forwarding paths upon failure of either link or node. TI-LFA leans on segment routing as it is also an extension of the IGP process. Using node and adjacency SIDs, TI-LFA can adequately protect all topologies, hence the distinction of *topology independence.*

**Operation Examination**

From a configuration standpoint, TI-LFA is only a few lines placed under the IGP configuration. But, it is essential to understand how the technology determines a suitable backup path. There is a specific method used to select the most suitable option.




Initial TI-LFA example topology

Activating TI-LFA on Gig0/0/0/6 will force the IGP to automatically select and install a backup path in the forwarding table. This entry will include all appropriate labels to route traffic around the failure while avoiding loops or traffic impairment.

```
router ospf 100
 area 0

  interface GigabitEthernet0/0/0/6
   cost 10
   network point-to-point
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa enable
```

TI-LFA interface configuration

The OSPF routes command displays the most pertinent information regarding the selection of the backup route.

```
RP/0/RP0/CPU0:1020O-P5#sh ospf routes 2.2.2.2/32 backup-path
Fri Dec  1 23:28:37.039 UTC

Topology Table for ospf 100 with ID 5.5.5.5

Codes: O - Intra area, O IA - Inter area
       O E1 - External type 1, O E2 - External type 2
       O N1 - NSSA external type 1, O N2 - NSSA external type 2

O    2.2.2.2/32, metric 11
       10.5.2.2, from 2.2.2.2, via GigabitEthernet0/0/0/6, path-id 1
           Backup path: TI-LFA, Repair-List: P node: 3.3.3.3         Label: 16003
              10.6.5.6, from 2.2.2.2, via GigabitEthernet0/0/0/1, protected bitmap 0000000000000001
              Attributes: Metric: 31, Interface Disjoint, SRLG Disjoint
```

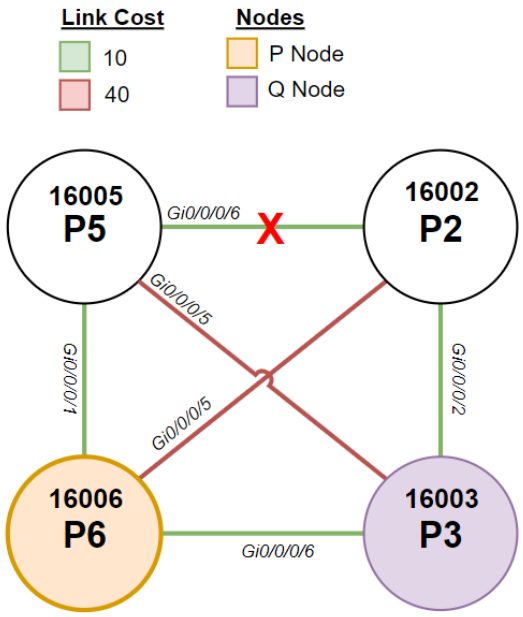
OSPF route backup path information

In this initial topology node P5 has selected an outgoing interface of Gi0/0/0/1 with a label of 16003 to route around the failure. To understand this behavior, some definitions must be introduced.

***P Space**: a set of nodes reachable (using pre-convergence paths) from node Source without using the protected link.*

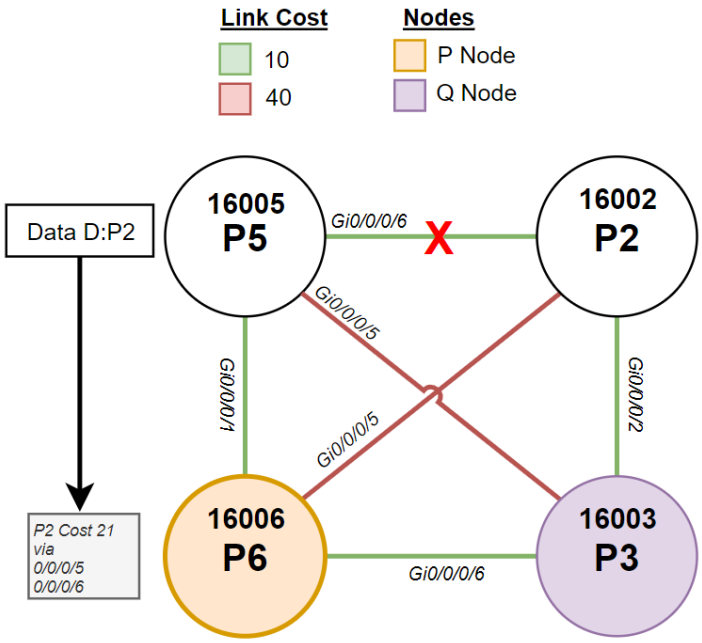
***Q Space**: a set of nodes that can reach (using pre-convergence paths) destination D without using protected link L*

Reviewing the diagram again, the P and Q space nodes can be revealed.

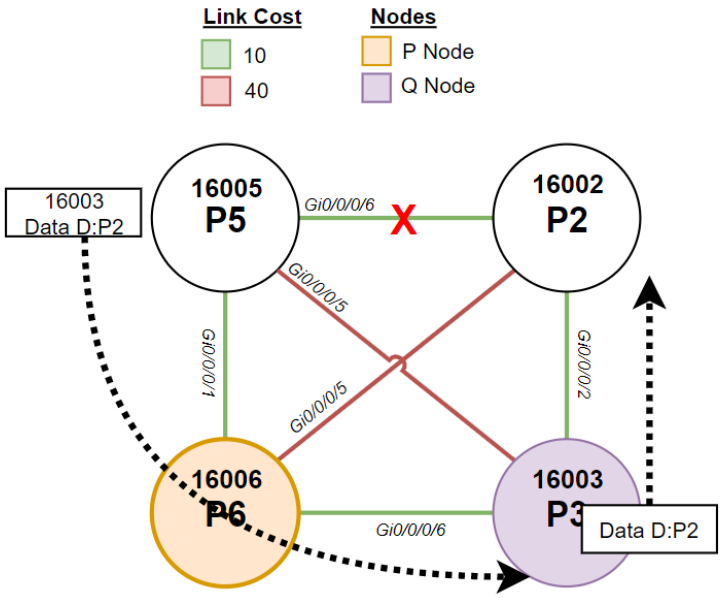



P and Q node distinction

These definitions can add unnecessary confusion. To simplify these terms, the P Space can be viewed as the routers that node P5 or the Point of Local Repair (PLR) can reach without traversing the protected link. This determination is based on cost, and documentation states that ECMP paths should also be disqualified. The Q Space is the point where a given node can reach the destination without needing to traverse the protected link. In the diagram above, P3 can reach the destination of 2.2.2.2/32 without ever needing to cross the protected link.

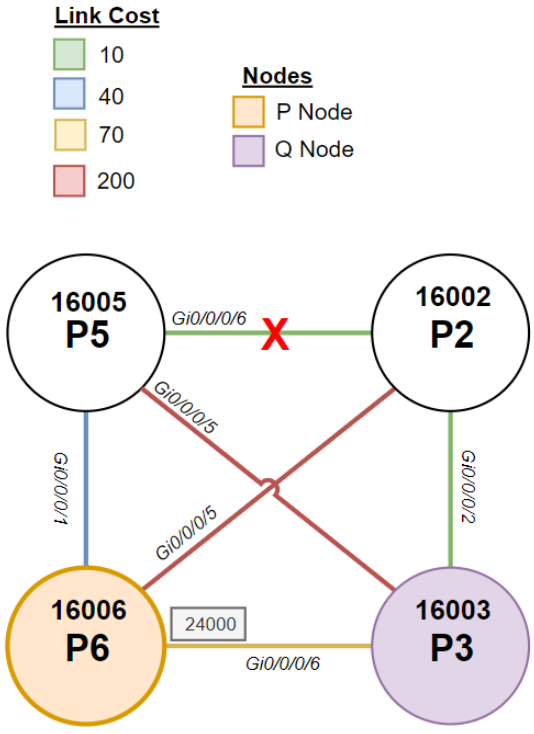



If traffic were to forward down the post-convergence path(s), the ECMP paths within node P6's table could potentially blackhole traffic. TI-LFA avoids this issue by taking action faster than IGP convergence and pushing the correct labels to prevent traffic loss. In the example above, upon link failure, the node SID of 16003 will be utilized on traffic destined toward 2.2.2.2/32. With this label, traffic will not be able to create a loop. When the traffic arrives unlabeled at P3, the destination of 2.2.2.2/32 will utilize Gig0/0/0/2 and have no reason to route back towards the broken link. When the IGP reconverges to accommodate the newly failed link, TI-LFA will have already placed the traffic flow on the correct path. This negates the need to swing traffic a second time, which was sometimes necessary for previous fast re-route technologies.




TI-LFA driven label imposition

Changing the link costs slightly demonstrates the flexibility of TI-LFA. These link alterations present more loop potential in the environment.




Link cost alteration diagram

With the new example, traffic is forced directly into a loop. Upon failure of the protected link, traffic will be sent to P6, but without IGP convergence, the traffic will hairpin towards P5. This is known as a micro loop. To avoid this, TI-LFA will use adjacency sids to force the traffic towards the Q space.

```
RP/0/RP0/CPU0:1020O-P5#sh ospf routes 2.2.2.2/32 backup-path
Sat Dec  2 OO:38:38.852 UTC

Topology Table for ospf 100 with ID 5.5.5.5

Codes: O - Intra area, O IA - Inter area
       O E1 - External type 1, O E2 - External type 2
       O N1 - NSSA external type 1, O N2 - NSSA external type 2

O    2.2.2.2/32, metric 11
       10.5.2.2, from 2.2.2.2, via GigabitEthernet0/0/0/6, path-id 1
           Backup path: TI-LFA, Repair-List: P node: 6.6.6.6         Label: 3
                                Q node: 3.3.3.3         Label: 24000
              10.6.5.6, from 2.2.2.2, via GigabitEthernet0/0/0/1, protected bitmap 0000000000000001
              Attributes: Metric: 121, Interface Disjoint, SRLG Disjoint
```

TI-LFA adjacency SID use

When the protected link fails traffic will now be sent with the adjacency SID value of 24000. Again, once traffic arrives at P3 unlabeled, it will follow the shortest path toward the destination. If traffic were to arrive at P6 with only the label 16003, the shortest path would still be through the failed link, again causing a micro loop. The adjacency SID forces traffic to P3, which can complete the path.

**Configuration Options**

TI-LFA can be enabled on all IGP interfaces. Operators can place the basic configuration on the interfaces they intend to protect, and the IGP will handle the rest. The one modifier is the type of protection in place. Although only link protection was used in this demonstration, node and Shared Risk Link Group (SRLG) are also available. TI-LFA is supported by both OSPF and ISIS

```
router ospf 100
area 0
interface GigabitEthernet0/0/0/1
   cost 40
   network point-to-point
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa enable
   fast-reroute per-prefix tiebreaker node-protecting index 100
```

TI-LFA OSPF configuration

```
router isis LAB
interface GigabitEthernet0/0/0/2
address-family ipv4
fast-reroute per-prefix
fast-reroute per-prefix ti-lfa
fast-reroute per-prefix tiebreaker node-protecting index 100
```

TI-LFA ISIS configuration

**Summary**

TI-LFA provides sub 50ms path convergence with little to no intervention from the operator. Using Segment Routing labels, traffic can be steered in various ways while avoiding traffic loops and loss. TI-LFA marks another consolidation brought upon by Segment Routing while providing a beneficial feature set.

## Overlay Services

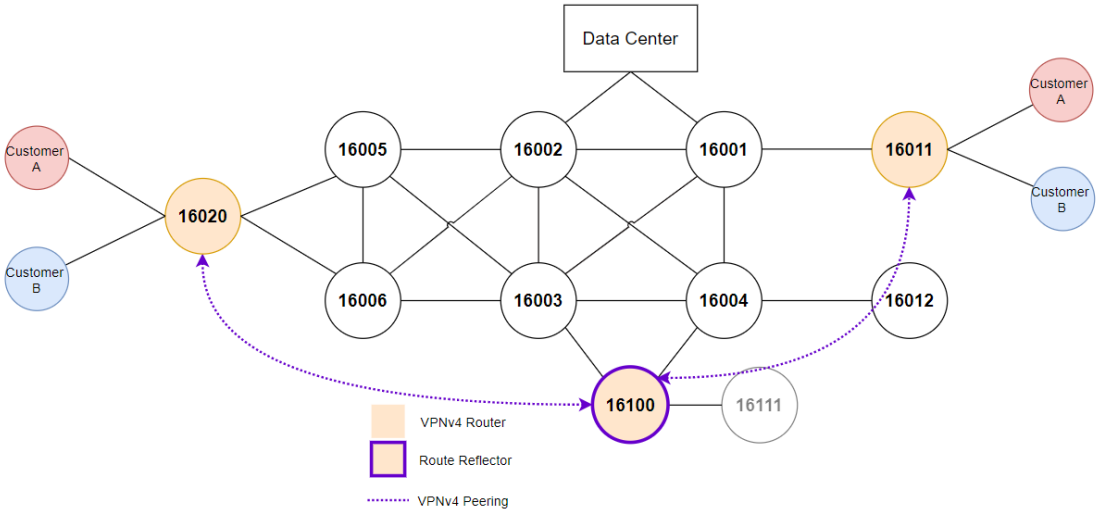
Overlay services are the heart of provider networks. Customers are not buying lines of service to peruse the core routing network. The main objective is to pair customers with content, provide private transit, or enable further services. The overlay service needs seamless connectivity to resources with little customer involvement. The following section will investigate multiple overlay options and the services that they enable/provide.

### L3VPN

Layer 3 VPNs are one of the most well-established services provided. Layer 3 VPNs enable private customer connectivity over the provider network's shared medium. The concept is that customers can interconnect multiple sites across many geographic areas while retaining private transport. The traffic placed within the VPN is isolated from other customer circuits. L3VPN uses VRF separation alongside various identifiers to keep traffic segmented throughout the path.

### VPNv4 Address Family

The base of the L3VPN  service is the BGP address family VPNv4 or VPNv6. This family enables using the extended communities of Route-Distinguisher (RD) and Route-Targets (RT). Combining these two values provides unique identifiers that aid in separating the traffic between individual customers.




VPNv4 Peering will utilize the same loopback addresses advertised through the IPv4 unicast address family. On IOS-XR nodes, no special consideration is needed regarding the extended communities. On IOS-XE devices, the use of the communities must be explicitly configured to avoid issues.

```
# IOS-XR PE Configuration

address-family vpnv4 unicast

neighbor 100.100.100.100
  remote-as 10200
  update-source Loopback0
  address-family vpnv4 unicast
   next-hop-self

#IOS-XE PE Configuration

address-family vpnv4
  neighbor 100.100.100.100 activate
  neighbor 100.100.100.100 send-community both
  neighbor 100.100.100.100 next-hop-self
```

IOS-XR and IOS-XE VPNv4 peeing configuration

Any route reflector configuration would remove the next-hop-self distinction and specify route-reflector-client under the configured peers. While the peering will now be up, sharing routes over the VPNv4 session warrants a discussion regarding VRFs, RDs, and RTs.

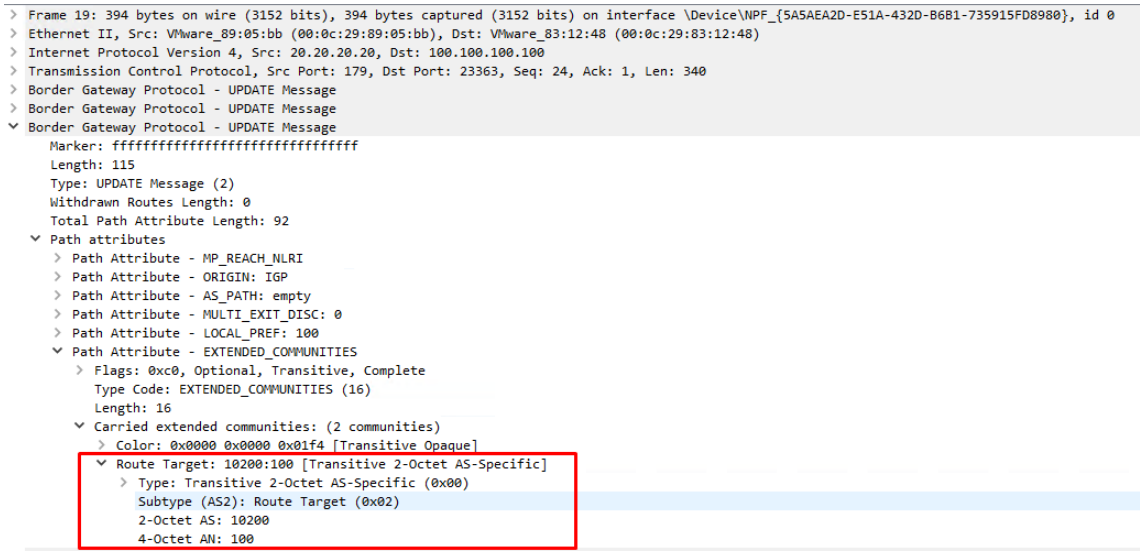
**L3VPN: Provider Edge**

At the provider edge router, a customer will connect to the provider network. VRFs are utilized to keep customer networks separated. Focusing on IOS-XR, the configuration is relatively similar to a traditional VRF-Lite deployment with the addition of Route-Target values under the address-family configuration.

```
vrf CXw-24
 address-family ipv4 unicast
  import route-target
   10200:24
  !
  export route-policy VPN-LL
  export route-target
   10200:24
```

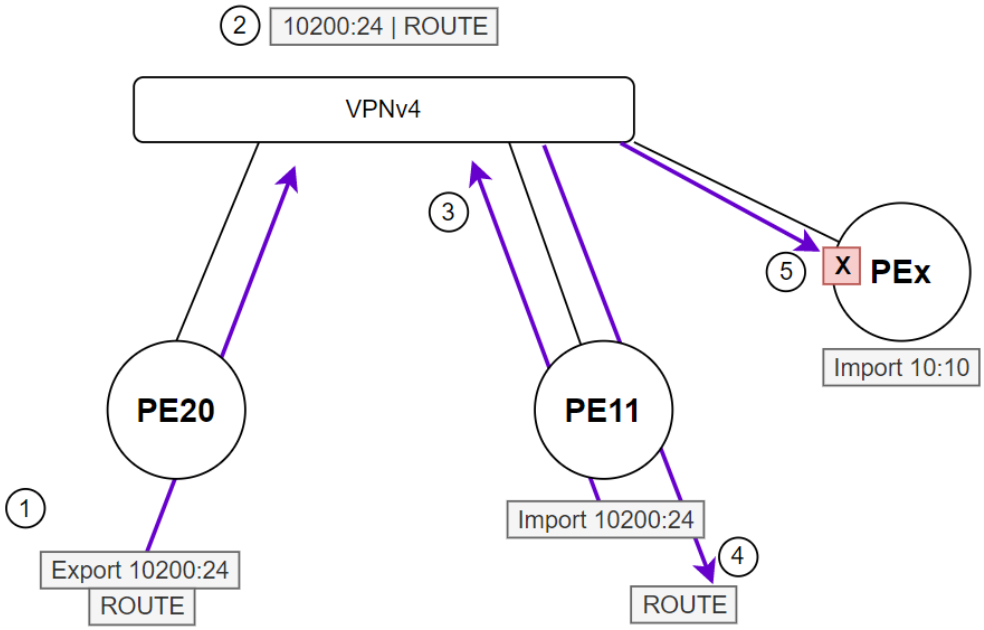
VPNv4 route-target configuration

A route target is now included within the VPNv4 NLRI UPDATE message.




BGP VPNv4 route advertisement

The purpose of the route-target value is to manage the import and export of routes into the customer VRF over the VPNv4 network.



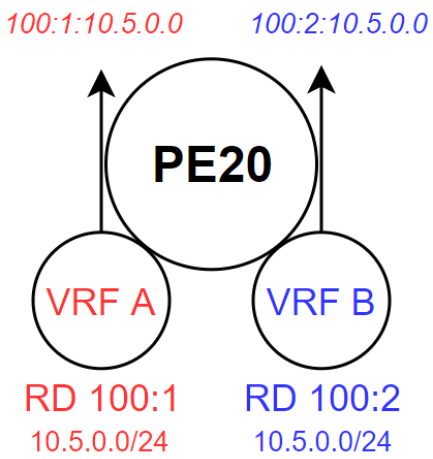

Simplified VPNv4 Import/Export

In the demonstrated lab configuration, all VPNv4 routes will be reflected to all peers. If a router does not possess the VRF utilizing the specific import target, that route will be dropped upon ingress. This behavior can be altered with the command *retain route-target all* which will retain all VPNv4 routes received. This override has specific use cases that are not discussed in this document.

```
address-family vpnv4 unicast
  retain route-target all
```

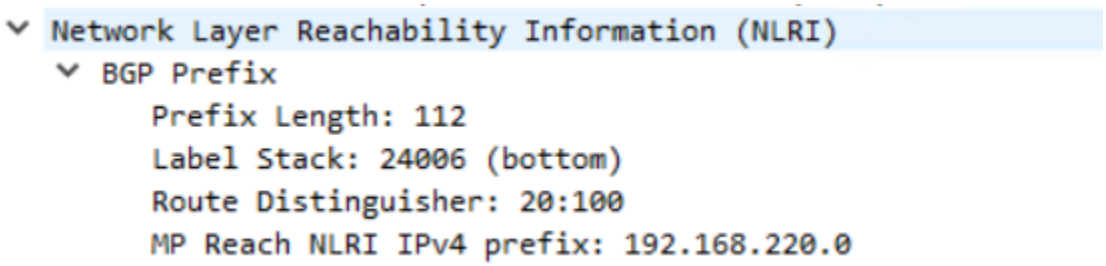
VPNv4 retention override.

The Route Distinguisher acts as a local identifier at an edge node. The main benefit of the RD is the ability to implement overlapping address space across customer VRFs.



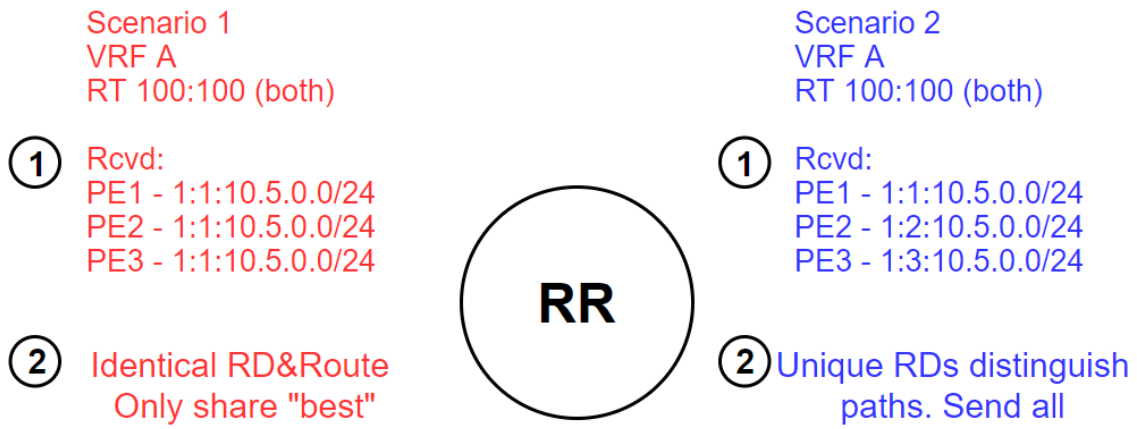

Route distinguisher advertisement

The RD is retained when advertised to other edge nodes. When customer routes are advertised from a VRF, an additional label is also sent with the BGP update. This label is used as the bottom of the stack label when sending traffic to the destination and allows the egress PE to identify the VRF to which the route belongs. With these attributes, multiple customers can share the same IP address ranges and be considered unique across the provider network. Below is a Wireshark capture displaying the advertised RD and bottom of the stack or VPN label.




Wireshark capture of VPNv4 update

RD values can be reused across the same customer VPNs on separate edge nodes, but this method may cause routing inefficiency. For instance, if multiple paths exist to a prefix but identical RDs are used across nodes, the route-reflectors will only share the best route. With the same RD used across multiple nodes, even if a customer has dual uplink across PE nodes, only one route will be shared.  Using different RD values, the route-reflectors consider all routes unique.




Unique RD Scenario

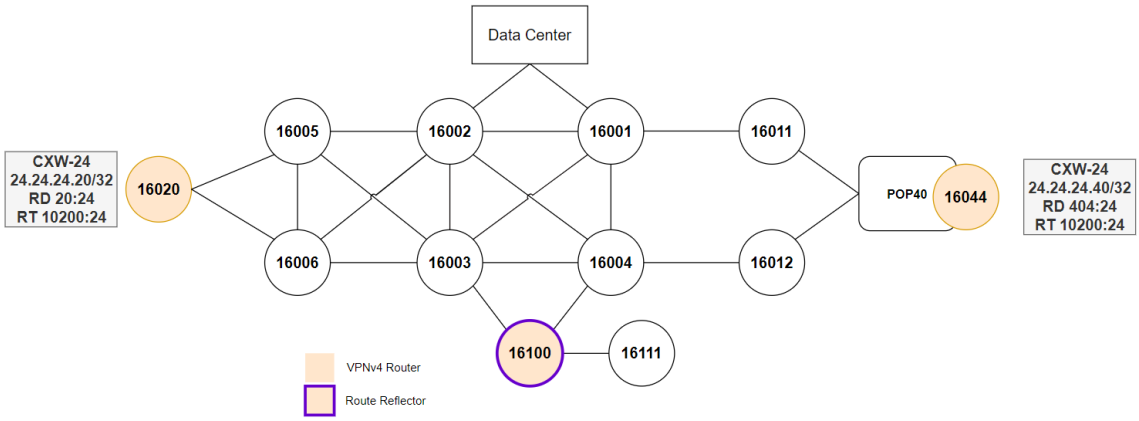
Customer peering is achieved utilizing eBGP via VRF-based IPv4 Unicast sessions. These sessions act identically to a global BGP session, with the difference being the added isolation enabled by the VRF. The BGP configuration can be completed after the VRF is created and assigned to an interface.

```
#IOS-XE BGP COnfiguration
vrf NAME
  rd value:value
  address-family ipv4 unicast
   network 24.24.24.24/32
  !
  neighbor X.X.X.X
   remote-as xxxx
   address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
```

IOS-XR VRF BGP configuration

In IOS-XR the RD value is set within the BGP configuration while in IOS-XE the RD is configured within the global VRF configuration.

Focusing on previously discussed values, a route propagation demonstration can be examined.




VPNv4 route propagation

Router PE20 (16020) receives a route for 24.24.24.20/32 from a directly attached customer via eBGP peering.

```
RP/0/RP0/CPU0:10200-PE20#sh ip route vrf CXw-24 24.24.24.20/32

Routing entry for 24.24.24.20/32
  Known via "bgp 10200", distance 20, metric 0
  Tag 80, type external
  Installed Nov  6 14:31:44.996 for 6d07h
  Routing Descriptor Blocks
      192.168.224.0, from 192.168.224.0, BGP external
      Route metric is 0
  No advertising protos.
```

BGP route on PE20

With this route being received via a configured VRF BGP peer, the route will be pushed into the VPNv4 address family. This propagation can be examined from the VPNv4 table.

```
RP/0/RP0/CPU0:10200-PE20#sh bgp vpnv4 unicast vrf CXw-24 advertised
neighbor 100.100.100.100
Sun Nov 12 22:22:44.064 UTC
Route Distinguisher: 20:24
24.24.24.20/32 is advertised to 100.100.100.100
  Path info:
      neighbor: 192.168.224.0   neighbor router id: 23.23.254.253
      valid  external  best  import-candidate
Received Path ID 0, Local Path ID 1, version 10
  Attributes after inbound policy was applied:
      next hop: 192.168.224.0
      MET ORG AS EXTCOMM
      origin: IGP  neighbor as: 80  metric: 0
      aspath: 80
      extended community: Color:600 RT:10200:24
  Attributes after outbound policy was applied:
      next hop: 20.20.20.20
      MET ORG AS EXTCOMM
      origin: IGP  neighbor as: 80  metric: 0
      aspath: 80
      extended community: Color:600 RT:10200:24
```

VPNv4 route install within BGP table

The route will be sent out toward peer 100.100.100.100, the route-reflector.
The route reflector has received the route update from PE20 and will send the route to the configured reflector clients.

```
RP/0/RP0/CPU0:10200-RR#sh bgp vpnv4 unicast | b 20:24
Sun Nov 12 22:27:33.984 UTC

Route Distinguisher: 20:24
*>i24.24.24.20/32       20.20.20.20              0     100   0 80 i
```

Route reflector VPNv4 table

Node 16044 will receive the route and examine the configured import route-targets. With a valid match, the route will be imported to the appropriate VRFs configured with the specified RT.

```
RP/0/RP0/CPU0:POP40-XR4#sh bgp vpnv4 unicast vrf CXw-24 24.24.24.20/32
Sun Nov 12 22:30:01.592 UTC
BGP routing table entry for 24.24.24.20/32, Route Distinguisher: 404:24
Versions:
  Process         bRIB/RIB  SendTblVer
  Speaker              22          22
Last Modified: Nov  7 20:42:47.119 for 5d01h
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  80
      20.20.20.20 C:600 (bsid:24005) (metric 3) from 100.100.100.100
(20.20.20.20)
      Received Label 24005
      Origin IGP, metric 0, localpref 100, valid, internal, best,
group-best, import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 18
      Extended community: Color:600 RT:10200:24
      Originator: 20.20.20.20, Cluster list: 100.100.100.100
      SR policy color 600, up, registered, bsid 24005, if-handle 0x0000001c

      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route
Distinguisher: 20:24
```

Some syntax highlights have been added to emphasize key values in the output above. The BGP prefix has been received with a label advertisement. This label will be placed at the bottom of the stack and retained across the provider network. When traffic arrives at the PE node, the bottom label will be used to forward traffic to the correct VRF instance. The RT import value has matched the configured VRF value on the local node, allowing the route to be retained. The Originator and Cluster value display that the route was sent from PE20 and reflected by 100.100.100.100. Finally, the source RD is also shared within the update.  The same route will be pushed into the IPv4 Unicast VRF table.

```
RP/0/RP0/CPU0:POP40-XR4#sh bgp vrf CXw-24 ipv4 unicast | i 24.24.24.20
Sun Nov 12 22:40:17.427 UTC
*>i24.24.24.20/32       20.20.20.20 C:600
```

IPv4 route table for route originally received via VPNv4

The propagation of the route allows end to end communication via L3VPN. A traceroute from the customer router connected to node 16044 displays the interdomain path utilized.

```
CX-W-HQ-POP40#traceroute vrf MPLS-NET 24.24.24.20 source lo100
Type escape sequence to abort.
Tracing the route to 24.24.24.20
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.44.1 14 msec 3 msec 2 msec
  2 10.4.1.1 6 msec 5 msec 5 msec
  3 10.40.11.11 12 msec 5 msec 5 msec
  4 10.11.1.1 8 msec 7 msec 5 msec
  5 10.3.1.3 6 msec 5 msec 5 msec
  6 10.6.3.6 7 msec 7 msec 5 msec
  7 10.20.6.20 6 msec 4 msec 4 msec
  8 192.168.224.0 5 msec *  20 msec
```

This configuration can be scaled across multiple routers. VPNv4 sessions are all that is needed in order to share the BGP routes depicted.

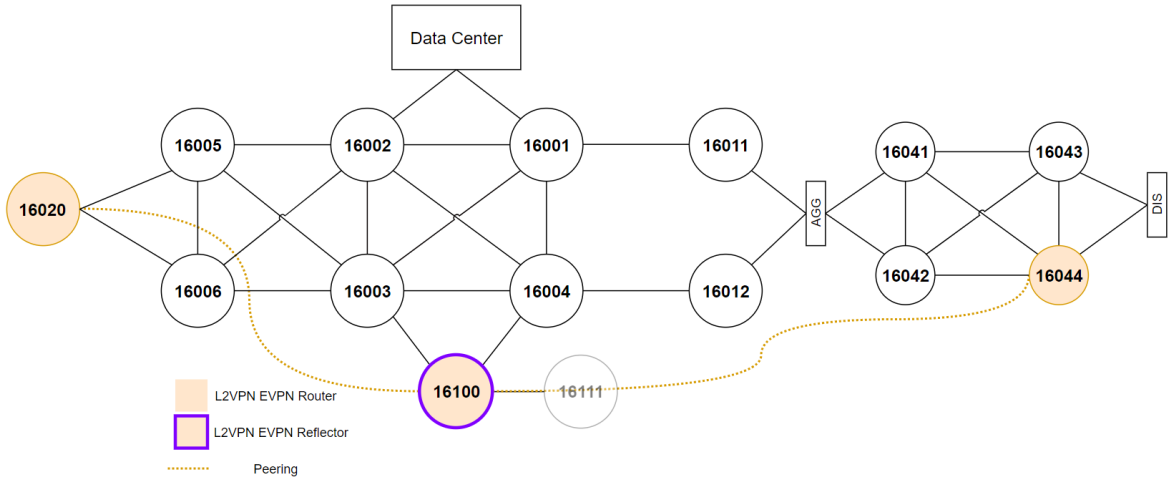
To further align previous discussions, segment routing is also a key factor in allowing L3VPN to operate in the manner demonstrated. The underlay must be a labeled path to pass this traffic without knowing the final destination along the transit routers. In this case, segment routing will provide a labeled path to traverse the provider network.

## L2VPN

Layer 2 VPNs (L2VPN) grant customers increased control and simplicity. Unlike the L3VPN service, which requires provider network interactions, L2VPN provides a wire-like service. The provider passes traffic transparently to provide the customer with a service that behaves as a layer 2 link.  Previously, L2VPN service creation was configuration intensive and would require the instantiation of many pseudo-wires. Today, most L2VPN implementations will be centered around Ethernet Virtual Private Network (EVPN). Specicifially the BGP address-family L2VPN EVPN. The BGP family allows the sharing of layer 2 addressing over a layer 3 medium. With MPLS/SR labeling layer 2 connectivity can be achieved across the provider core.
L2VPN allows different options for connection behavior as well. The service can act as a traditional point-to-point, Virtual Private Wire Service (VPWS) or as a shared medium known as Virtual Private LAN Service (VPLS).

## BGP L2VPN EVPN VPWS

The BGP L2VPN EVPN address family inherits many of the same principles of VPNv4, such as the use of route targets and distinguishers.




L2VPN path and node role

The diagram above depicts the peering configuration that will be utilized for the VPWS demonstration. BGP configuration enables the address family for each peer, and the proper distinctions are made for route-reflection and next hop self.

```
#Route Reflector
  !
  address-family l2vpn evpn
   route-reflector-client
  !

#PE
  !
  address-family l2vpn evpn
   next-hop-self
  !
```

Route reflector and PE BGP configuration

With peering established, the session can exchange routes. To generate routes, additional L2VPN configuration needs to be in place.

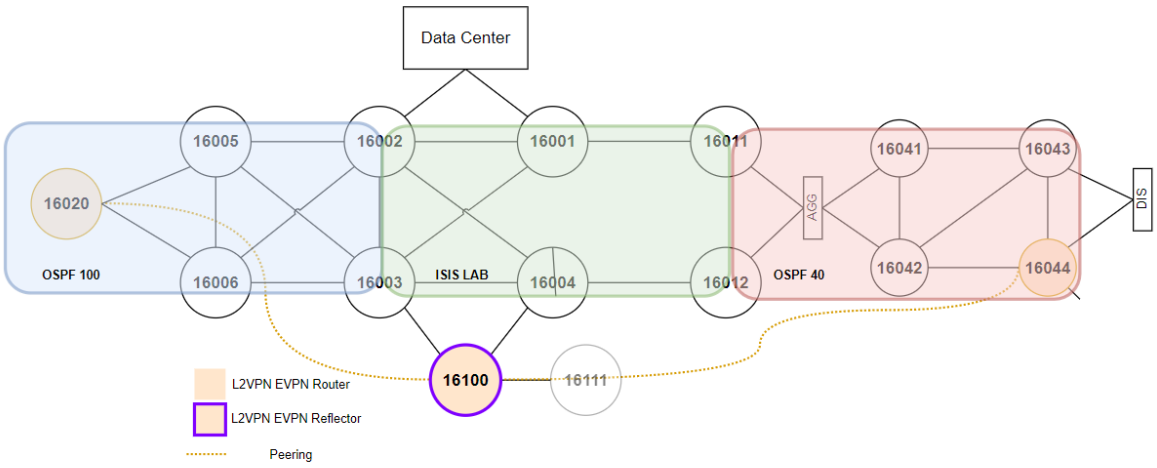
```
l2vpn
xconnect group 20_2_4
  p2p xc224
   interface GigabitEthernet0/0/0/1.77
   neighbor evpn evi 1001 target 10001 source 20001
      pw-class ONE
!
interface GigabitEthernet0/0/0/1.77 l2transport
 encapsulation dot1q 77
```

L2VPN configuration

The L2VPN configuration above will enable basic layer 2 forwarding behavior for the specified interface. The group distinguisher is simply an operator-selected name. The P2P configuration specifies the cross-connect (xc) name. The included interface(s) are specified, and the EVPN neighbor statement is defined.

The EVPN instance (EVI) acts as a portion of the route target (both import and export). The RT creation is handled automatically within IOS-XR. The target and source value specified after the EVI will represent the remote and local attachment circuits (AC). An attachment circuit is where traffic will ingress/egress for a specific group.

The final group configuration specifies a pw-class, which is short for pseudowire class. This class configuration allows operators to set parameters such as encapsulation type (MPLS) and preferred paths for the layer 2 traffic. In the case of this lab, the multi-domain nature will force the use of an SR path, which will be called explicitly in the first example.




EVPN domain traversal

To stitch traffic across the domains, an explicit SR policy will be created.

```
#PE20
segment-routing
traffic-engineering
 segment-list PW-PATH-01
   index 1 mpls label 16005
   index 2 mpls label 16002
   index 3 mpls label 16001
   index 4 mpls label 16011
   index 5 mpls label 16041
   index 6 mpls label 16044


  policy PseudoWire_Pol1
   color 77 end-point ipv4 10.40.4.4
   candidate-paths
      preference 100
      explicit segment-list PW-PATH-01

l2vpn
pw-class ONE
  encapsulation mpls
   preferred-path sr-te policy srte_c_77_ep_10.40.4.4
```

L2VPN explicit SRTE policy

This traffic will need explicit instruction at both ends of the labeled path to function properly. Once both ends have been configured and the SR policy has become active, the customer devices will have a continuous layer 2 path between sites.

```
RP/0/RP0/CPU0:1020O-PE20#sh l2vpn xconnect group 20_2_4
Thu Nov 16 21:46:33.163 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
        LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                  Segment 1                       Segment 2
Group      Name       ST   Description            ST      Description            ST
------------------------   ----------------------------   ----------------------------
20_2_4     xc224      UP   Gi0/0/0/1.77           UP      EVPN 1001,10001,10.40.4.4
                                                                                UP
------------------------------------------------------------------------------------
```

L2VPN xc status

Adding the detail specification to the *show l2vpn xconnect group <name>* command will provide additional output that can be useful during troubleshooting. With the configured interfaces from the L2VPN group in a UP state, routes will be sent and received over the BGP L2VPN EVPN session.

```
RP/0/RP0/CPU0:10200-PE20#sh bgp l2vpn evpn rd 10.40.4.4:1001 [1][0000.0000.0000.0000.0000][10001]/120
Thu Nov 16 21:51:57.967 UTC
BGP routing table entry for [1][0000.0000.0000.0000.0000][10001]/120, Route Distinguisher: 10.40.4.4:1001
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 3           3
Last Modified: Nov 14 06:24:34.217 for 2d15h
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    10.40.4.4 from 100.100.100.100 (10.40.4.4)
      Received Label 24008
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 3
      Extended community: EVPN L2 ATTRS:0x02:0 RT:10200:1001
      Originator: 10.40.4.4, Cluster list: 100.100.100.100
RP/0/RP0/CPU0:10200-PE20#
```

L2VPN routes scoped to rd

Highlighted in red are the automatically configured values previously a manual task within the VPNv4 address family. The values are easy to decipher as the route-distinguisher is RID:EVI, and the route target is ASN:EVI. The RD and RTs serve the same purpose as they did in VPNv4. All routes belonging to EVI 1001 will be imported and exported at each respective PE. The RID:EVI combination will distinguish overlapping values at the PE.

With bi-directional SR policies in place and assigned to the pseudowire class, the layer 2 traffic will now be able to flow across the core. This will be demonstrated by establishing an OSPF peering across the client routers.

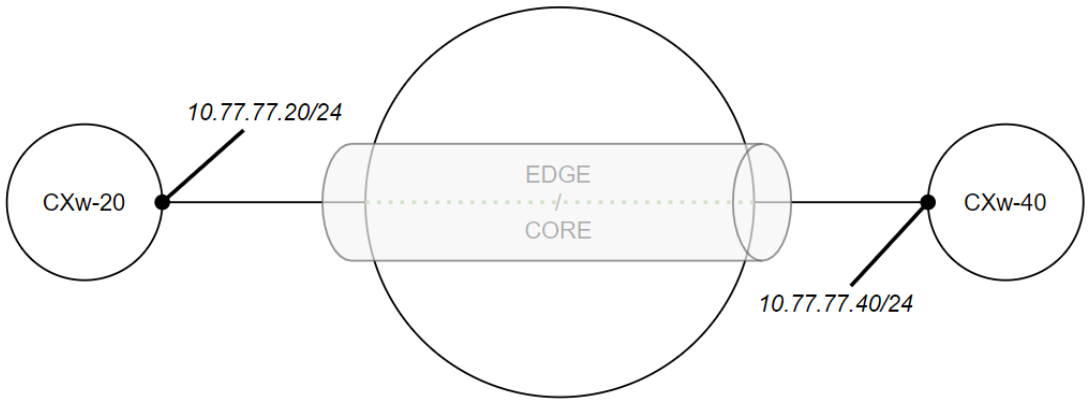
```
12 2.7374… 10.77.77.20   224.0.0.5   156 OSPF   Hello Pa
› Frame 12: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface \Device\NPF_{5A5AEA2D
› Ethernet II, Src: VMware_fe:84:18 (00:0c:29:fe:84:18), Dst: VMware_2e:5c:18 (00:0c:29:2e:5c:18)
› MultiProtocol Label Switching Header, Label: 16002, Exp: 0, S: 0, TTL: 254
› MultiProtocol Label Switching Header, Label: 16001, Exp: 0, S: 0, TTL: 254
› MultiProtocol Label Switching Header, Label: 16011, Exp: 0, S: 0, TTL: 254
› MultiProtocol Label Switching Header, Label: 16041, Exp: 0, S: 0, TTL: 254
› MultiProtocol Label Switching Header, Label: 16044, Exp: 0, S: 0, TTL: 254
› MultiProtocol Label Switching Header, Label: 24008, Exp: 0, S: 1, TTL: 254
› PW Ethernet Control Word
› Ethernet II, Src: 05:38:81:00:00:4d (05:38:81:00:00:4d), Dst: Cisco_0c:29:59 (00:05:00:0c:29:59)
› Internet Protocol Version 4, Src: 10.77.77.20, Dst: 224.0.0.5
› Open Shortest Path First
```

OSPF Hello with label stack

```
  3 3.0050… 100.64.77.… 100.64.77.… ICMP 156 Echo (ping) request  id=0x002f, seq=0/0, ttl=255 (reply in 4)

> Frame 3: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface \Device\NPF_{5A5AEA2D-E51A-43…
> Ethernet II, Src: VMware_fe:84:18 (00:0c:29:fe:84:18), Dst: VMware_2e:5c:18 (00:0c:29:2e:5c:18)
> MultiProtocol Label Switching Header, Label: 16002, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 16001, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 16011, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 16041, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 16044, Exp: 0, S: 0, TTL: 254
> MultiProtocol Label Switching Header, Label: 24008, Exp: 0, S: 1, TTL: 254
> Ethernet II, Src: VMware_59:05:38 (00:0c:29:59:05:38), Dst: VMware_1f:2e:4a (00:0c:29:1f:2e:4a)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 77
> Internet Protocol Version 4, Src: 100.64.77.20, Dst: 100.64.77.40
> Internet Control Message Protocol
```

Ping across L2VPN with label stack.

As displayed in the Wireshark captures, the node pushes the explicit label path on the traffic upon ingress. Through this process, the client routers are abstracted from all core routing and see the link as a direct connection to the remote peer. This service allows clients to link multiple sites in various ways as the service no longer requires layer 3 interactions with provider routers.




Customer router behaivor

**L2VPN EVPN SR ODN**

With many clients, the explicit path configuration would quickly become cumbersome. Luckily, the On-Demand policy behavior can be leveraged to steer traffic over PCE-delivered paths. This allows increased flexibility and configuration ease. This can be done via the L2VPN EVPN address family or the EVPN configuration, where specific EVIs can be selected.

```
#PE20 Configuration
segment-routing
traffic-eng
 on-demand color 74
   dynamic
      Pcep

evpn
 evi 1001
  bgp
   route-policy export EVPN-DYNAMIC

route-policy EVPN-DYNAMIC
  if evpn-route-type is 1 then
      set extcommunity color EVPN-74
  Endif

extcommunity-set opaque EVPN-74
  74
end-set
```

This configuration is very similar to what has already been performed in previous sections. With this, routes will be tagged with a specific color upon egress.

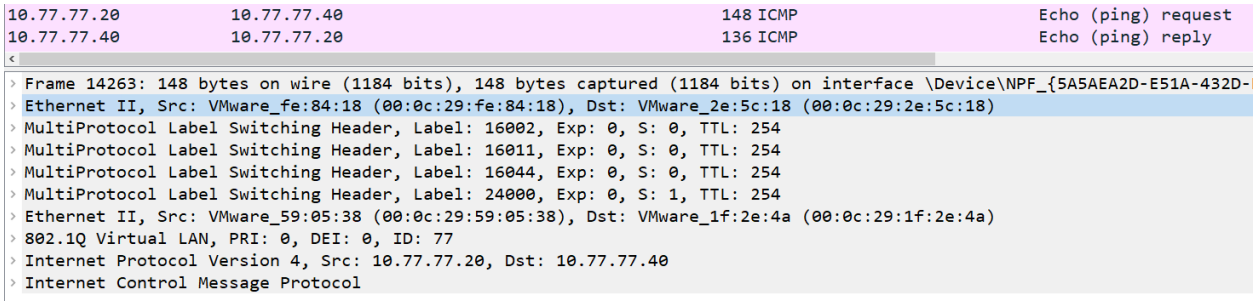
```
RP/0/RP0/CPU0:10200-PE20#sh bgp l2vpn evpn  rd 10.40.4.4:1001 | b Route Dis
Sat Dec  2 13:07:57.922 UTC
Route Distinguisher: 10.40.4.4:1001
*>i[1][0000.0000.0000.0000.0000][10001]/120
                     10.40.4.4 C:74               100      0 i
```

With the configuration again mirrored on both edge nodes, pings can flow with the new label stack handed out by the PCE device.

```
Color: 74, End-point: 10.40.4.4
  Name: srte_c_74_ep_10.40.4.4
  Status:
    Admin: up  Operational: up for 00:53:46 (since Dec  2 12:15:30.476)
  Candidate-paths:
    Preference: 200 (BGP ODN) (inactive) (shutdown)
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Dynamic (inactive)
        Metric Type: TE,   Path Accumulated Metric: 0
    Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: bgp_c_74_ep_10.40.4.4_discr_100
        PLSP-ID: 25
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Dynamic (pce 111.111.111.111) (valid)
        Metric Type: TE,   Path Accumulated Metric: 33
          16002 [Prefix-SID, 2.2.2.2]
          16011 [Prefix-SID, 11.11.11.11]
          16044 [Prefix-SID, 10.40.4.4]
```

EVPN SR ODN policy

A Wireshark capture also confirms the appropriate label stack is in use.




Wireshark capture of On-demand VPWS traffic

With the 9000v, it was noticed that the traffic would not flow over the policy automatically. While the PCC/PCE recognized the specific color value and triggered the calculation process, the layer 2 traffic would only steer to the policy with explicit configuration in the pw-class.

**EVPN VPLS**

Virtual Private LAN Service allows a full mesh of connectivity to client sites. This service operates as a shared segment where client devices will learn the MAC addresses from the provider-connected interface. Again, this option provides the customer with many options for interconnecting sites.

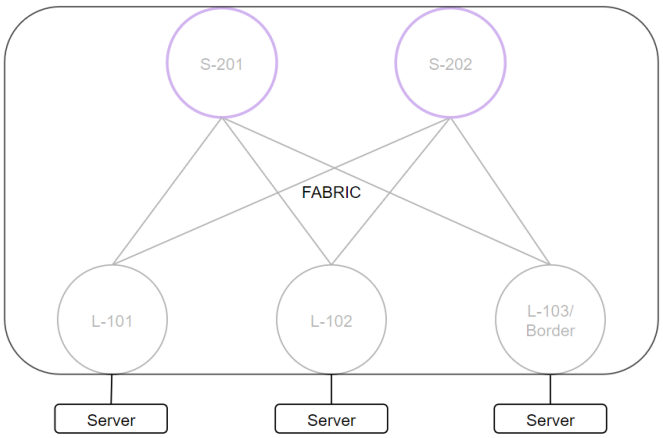
Unfortunately, this is again a limitation of the virtualized ASR platform. Below, the generic configuration will be shared with a Cisco Doc link.

```
#Create EVPN Instance
evpn
 evi 8000
  bgp
   rd 20:8000
   route-target import 10200:8000
   route-target export 10200:8000
  !
  advertise-mac
  !
 !

#Assign Interface and Bridge-Domain in L2VPN

l2vpn
 bridge group 800
  bridge-domain BD-800
   interface GigabitEthernet0/0/0/1.800
evi 8000

#Failure Message on ASR XRv 9000

RP/0/RP0/CPU0:10200-PE20(config-l2vpn-bg-bd-evi)#sh configuration failed
Tue Nov 21 08:51:59.725 UTC
!! SEMANTIC ERRORS: This configuration was rejected by
!! the system due to semantic errors. The individual
!! errors with each failed configuration command can be
!! found below.

l2vpn
 bridge group 800
  bridge-domain BD-800
   interface GigabitEthernet0/0/0/1.800
```

```
!!% Invalid argument: VPLS Bridge domains not supported on this platform
   !
  !
 !
!
end
```

**Data Center (VXLAN)**

Continuing from the discussion surrounding EVPN use in the core, the focus will now shift to the Data Center. Data Center EVPN aims to solve a different issue than EVPN over the core, although the underlying functions utilize the same address family. In the Data Center, EVPN is used for VXLAN fabrics. This fabric allows seamless east-west traffic while enabling host mobility. VXLAN layers a good amount of additional configuration that will be explained in the coming sections.

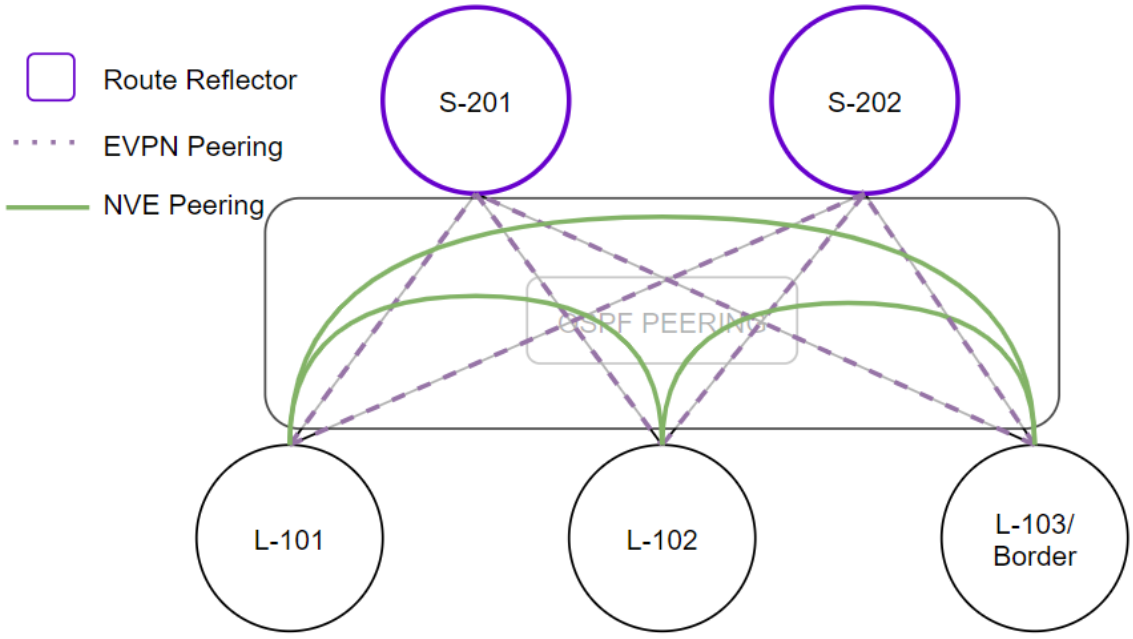



Basic DC fabric

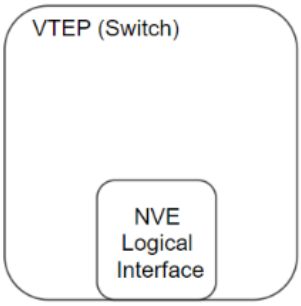
The high-level goal is to configure the overlay service to emulate a single switching domain. In reality, this domain is disaggregated across multiple nodes. This configuration option alleviates the need to span layer 2 domains and leverages EVPN to transport the needed information over BGP sessions. VXLAN also allows multi-tenancy, allowing the single fabric to act as multiple fabrics with various levels of segmentation. The example outlined in this document will examine simple Layer 2 and Layer 3 instances.

**VXLAN Layer 2 Concepts**

When VXLAN acts as a Layer 2 domain, the forwarding behavior from a client perspective is no different than connecting directly to the same switch with VLAN access configured on each port. The operator's role is more involved in enabling this behavior. While the underlay of the Data Center fabric has already been discussed, further references will be made to understand the overlay behavior.




In VXLAN, the addition of a Network Virtualization Interface or Network Virtualization Edge (NVE) (depending on the documentation author) is utilized to encapsulate MAC traffic and trigger advertisements of reachability information. This interface will also be synonymous with the Virtual Tunnel End Point (VTEP). To clarify the terminology, an NVE is an interface where encapsulation/decapsulation occurs, and the VTEP is the device that handles encapsulation/decapsulation.



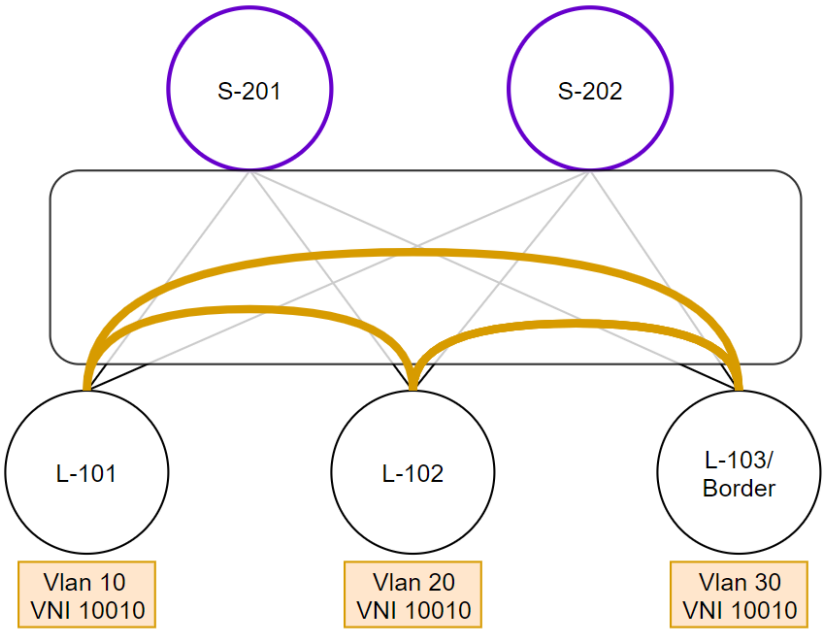

NVE/VTEP clarification

An NVE depends upon a loopback interface to source traffic. A good practice is not reusing loopbacks between services (BGP, OSPF, IS-IS) in VXLAN. NVE's are only present on leaf devices. The configuration displayed in the remainder of this section will be taken exclusively from Cisco NX-OS devices.

```
#Source Loopback
interface loopback192
  description NVE Loopback
  ip address 192.168.1.101/32
  ip router ospf 100 area 0.0.0.0
  ip pim sparse-mode

#NVE
interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback192
```

The NVE configuration above specifies the use of loopback 192 as the source interface and also instructs to utilize BGP as the desired control plane protocol.  With the configuration displayed, the NVE is not fully functional but ready to encapsulate/decapsulate traffic. The following configuration will examine getting traffic to flow through the NVE.

Another significant feature of VXLAN is the expanded ID space. Using a Virtual Network Identifier (VNI), the space ranges from 1 to 16777215. It is essential not to confuse this with the regular configurable VLAN range, which remains the usual 1 to 4094. The VNI is utilized in the overlay to separate traffic and provide unique values.




As the diagram depicts, the VNI adds another lidentification layer. With VXLAN, the VLANs local to the switch are no longer globally significant across the domain as they were in legacy layer 2 data centers. The VNI now acts as the global value. With identical VNI values, hosts attached to the three separate VLANs can reach each other across the fabric. VNI distinction is made under the VLAN configuration mode.

```
vlan 100
  vn-segment 10010
```

VNI configuration

Assigning the VLAN to access ports and or trunks south of the VXLAN fabric edge does not require any new style configuration. After a VLAN is assigned to the desired ports, the NVE interface must be made aware that this VNI needs to be advertised across the fabric.

```
interface nve1
  member vni 10010
      ingress-replication protocol bgp
```

VNI association to the NVE

At this time, the MAC address learned on VLAN 100 (VNI 10010) will be picked up by the EVPN process and forwarded to other interested peers. To address all layer 2 forwarding caveats, VXLAN must provide a mechanism to handle Broadcast, Unknown Unicast, and Multicast (BUM) traffic. Two methods have been provided, the first being the addition of protocol-independent multicast (PIM) in the underlay. In the PIM-driven solution, multiple VNIs are assigned to a specific multicast group, and BUM traffic is sent to the specified group

address to reach all intended recipients. In the configuration demonstrated, BGP ingress replication will be utilized. Ingress replication uses a specific VXLAN route type, allowing the auto-discovery of peer NVEs. When this peering is established, overlay tunnels may form to handle BUM traffic between interested nodes. Just as VLANs constrain flooding domains, VNI-specific BUM traffic will also be isolated.

```
10200-DC-L102# sh bgp l2vpn evpn vni-id 10010 route-type 3
BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 10.65.100.102:32867  (L2VNI 10010)
BGP routing table entry for [3]:[0]:[32]:[192.168.1.101]/88, version 74
Paths: (1 available, best #1)
Flags: (0x000012) (high32 00000000) on xmit-list, is in l2rib/evpn, is not
in HW

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop
             Imported from
10.65.100.101:32867:[3]:[0]:[32]:[192.168.1.101]/88
  AS-Path: NONE, path sourced internal to AS
      192.168.1.101 (metric 81) from 10.74.0.201 (10.65.100.201)
      Origin IGP, MED not set, localpref 100, weight 0
      Extcommunity: RT:65500:10010 ENCAP:8
      Originator: 10.65.100.101 Cluster list: 10.65.100.201
      PMSI Tunnel Attribute:
      flags: 0x00, Tunnel type: Ingress Replication
      Label: 10010, Tunnel Id: 192.168.1.101
```

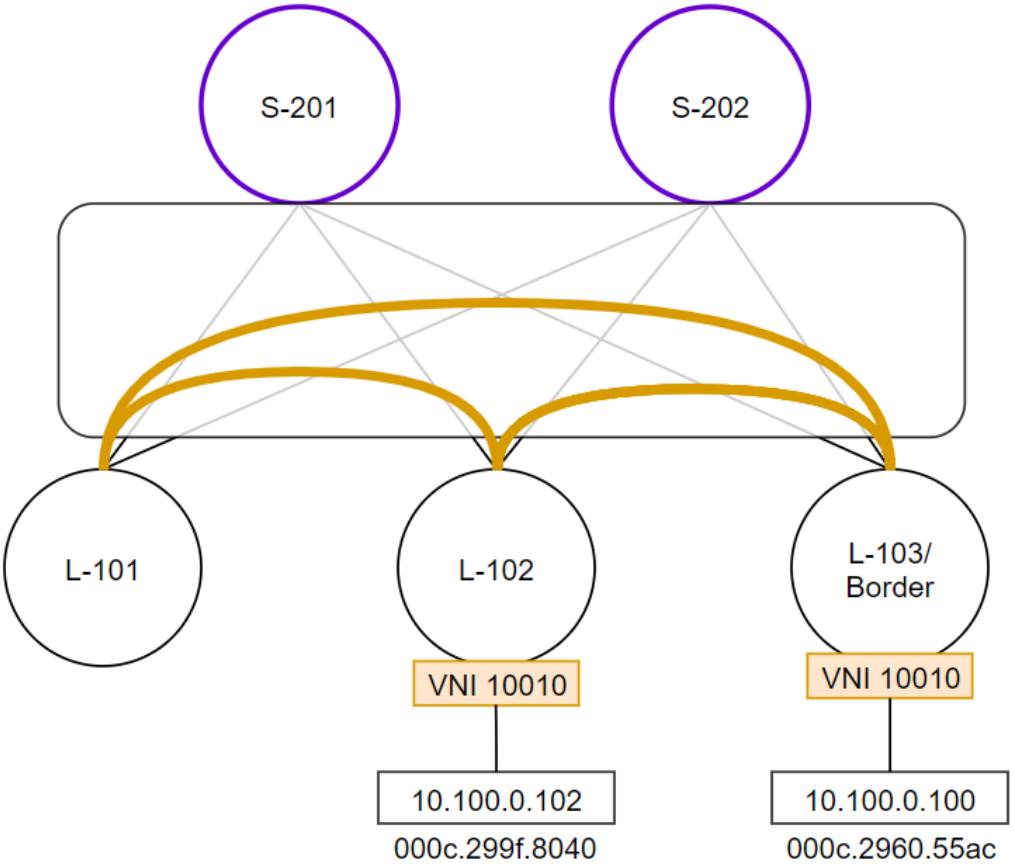
Ingress replication route type 3

With the VNI configured on the NVE(s) the EVPN routes will be shared across peers. The figure above shows the type-3 route used to auto-discover and establish BUM traffic-specific tunneling. The Provider Multicast Service Interface (PMSI) tunnel attribute shares essential information to distinguish tunnels. The values used are the tunnel type, which is zero and represents ingress replication, and the tunnel label, which will be identical across peers as that value is the configured VNI. Finally, the tunnel ID specifies the leaf where the advertisement originated. After route type 3 information has been shared, leafs where the VNI has been configured will form peer relationships.

```
10200-DC-L102# sh nve peers detail | i Learn|Peer-Ip
Peer-Ip: 192.168.1.101
      Learnt CP VNIs    : 10010
Peer-Ip: 192.168.1.103
      Learnt CP VNIs    : 10010
```

Verifying peering with L-101 and L-103

The detail command output has been narrowed to display only the peer IP and the VNIs that have been learned. With the addition of two end hosts at L102 and L103, the EVPN MAC advertisements can be further examined.




Hosts associated with VNIs

Beginning at L-102, the directly connected MAC address for host 10.100.0.102 now has an EVPN entry and is being advertised northbound to the two spine units.

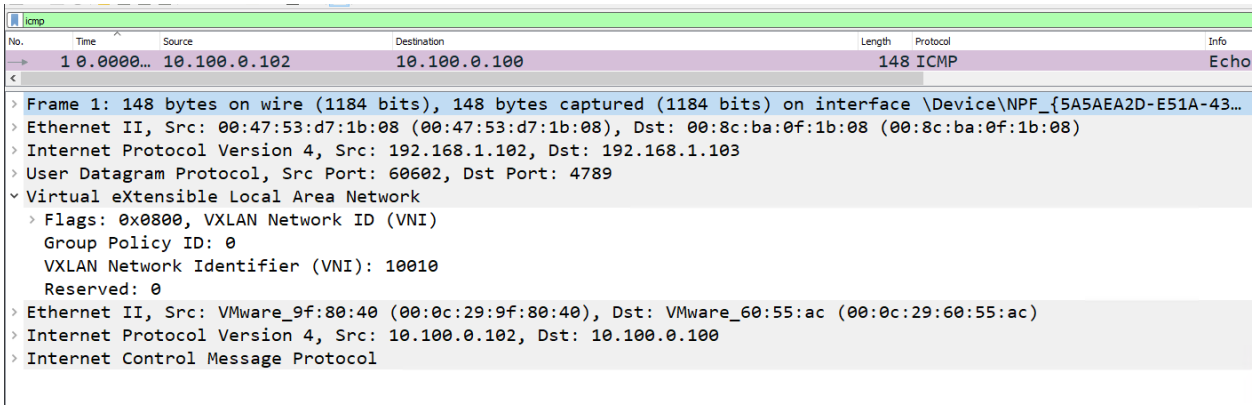
```
#From L-102
BGP routing table entry for
[2]:[0]:[0]:[48]:[000c.299f.8040]:[0]:[0.0.0.0]/216, version 56
Paths: (1 available, best #1)
Flags: (0x000102) (high32 00000000) on xmit-list, is not in l2rib/evpn

  Advertised path-id 1
  Path type: local, path is valid, is best path, no labeled nexthop
  AS-Path: NONE, path locally originated
      192.168.1.102 (metric 0) from 0.0.0.0 (10.65.100.102)
      Origin IGP, MED not set, localpref 100, weight 32768
      Received label 10010
      Extcommunity: RT:65500:10010 ENCAP:8

  Path-id 1 advertised to peers:
      10.74.0.201       10.74.0.202
```

MAC entry for the recently attached host at L-102

The route type displayed is an EVPN route-type 2. This route type carries MAC and IP information and is the most common route seen in VXLAN. Extended communities in VXLAN are integral to identifying and steering traffic. Highlighted above, the two communities observed are the route-target and the encapsulation value. The route-target within VXLAN is automatically derived utilizing *ASN:VNI.* The encapsulation value of 8 identifies VXLAN encapsulation. Node L-102 has also learned routes from peer L-103 for the MAC address of its connected host.

```
#L-102
BGP routing table entry for [2]:[0]:[0]:[48]:[000c.2960.55ac]:[0]:[0.0.0.0]/216, version 67
Paths: (1 available, best #1)
Flags: (0x000212) (high32 00000000) on xmit-list, is in l2rib/evpn, is not in HW

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop, in rib
             Imported from
10.65.100.103:32867:[2]:[0]:[0]:[48]:[000c.2960.55ac]:[0]:[0.0.0.0]/216
  AS-Path: NONE, path sourced internal to AS
       192.168.1.103 (metric 81) from 10.74.0.201 (10.65.100.201)
       Origin IGP, MED not set, localpref 100, weight 0
       Received label 10010
       Extcommunity: RT:65500:10010 ENCAP:8
       Originator: 10.65.100.103 Cluster list: 10.65.100.201
```

With the information shared to both necessary peers, two-way communication between the hosts succeeds. The Wireshark capture displays the stacked nature of VXLAN. The external portion of the packet identifies the VTEP addresses, while the data beyond the VXLAN header is the original information.

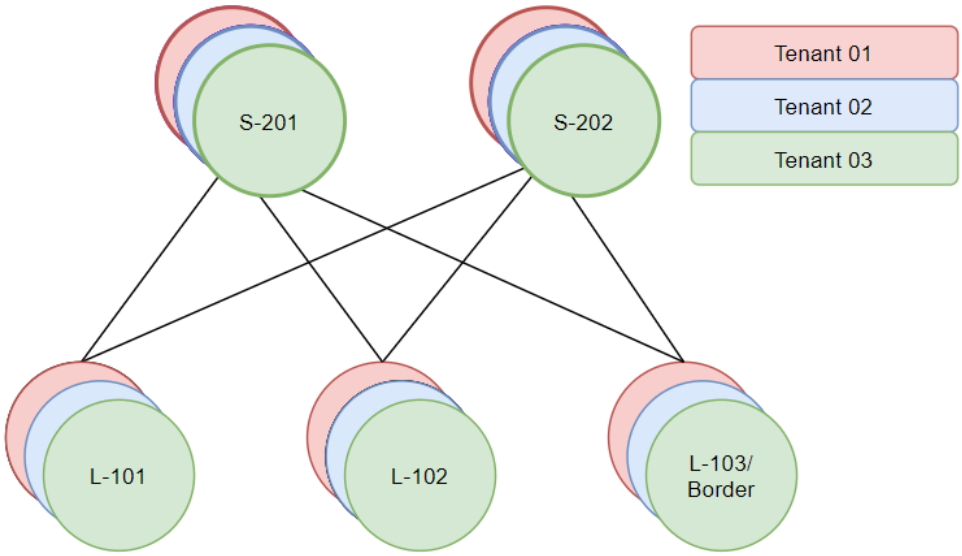



Wireshare capture of host communication from L-102 to L-103

This initial demonstration has proven the concept of layer 2 VXLAN forwarding. Clients behave as if they were hosted on the same layer 2 segment but are passing through a layer 3 fabric.

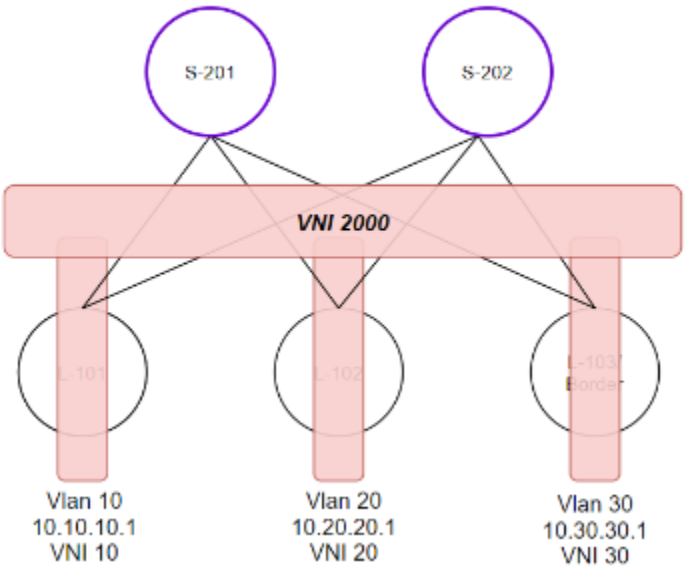
## VXLAN Layer 3 Concepts

The previous section only explored the possibility of layer 2 forwarding over the fabric. In the following section Layer 3 behaviors will be explored. The demonstration will go beyond a single VLAN and allow routing between VLANs. Also, integration with connections external to the fabric will be investigated. Layer 3 VNI (L3VNI) is a construct that provides routing between different VLANs and creates an isolated tenant in the VXLAN architecture.




VXLAN VRF separated tenant diagram

Through this configuration, tenants can independently operate while utilizing the same shared underlay. Furthermore, the L3VNI acts as the connection between all the tenant VLANs that require routing. As mentioned, the fabric acts as a layer 2 switch, but the L3VNI alters that behavior and now allows Layer 3 routing. With this change comes additional configuration that will be explored through the following example in the diagram below.




L3VNI topology

Tenant 01 Will utilize L3VNI 2000 to allow routing across the three respective VLANs. The first step is to create a VRF and VLAN and then associate the VNI.

```
vlan 2000
  name TEN-01-VNI
  vn-segment 2000

vrf context TEN-01
  rd auto
  address-family ipv4 unicast
      route-target both auto
      route-target both auto evpn
  vni 2000

interface Vlan2000
  no shutdown
  vrf member TEN-01
  no ip redirects
  ip forward

interface nve 1
 member vni 2000 associate-vrf
```

L3VNI VRF and VNI creation

The configuration above can be placed at every leaf node. This configuration creates a VRF to propagate IP routing information throughout the fabric using extended communities. These communities will be explored once valid routes have been shared.  The empty interface VLAN 2000  enables the passing of traffic between all members of the VRF. This will be explained in further detail in an upcoming section, which will surround using the Router MAC (RMAC) attribute in BGP advertisements. Specific configuration will also be needed at leaf nodes under each local VLAN interface placed within Tenant-01’s domain.

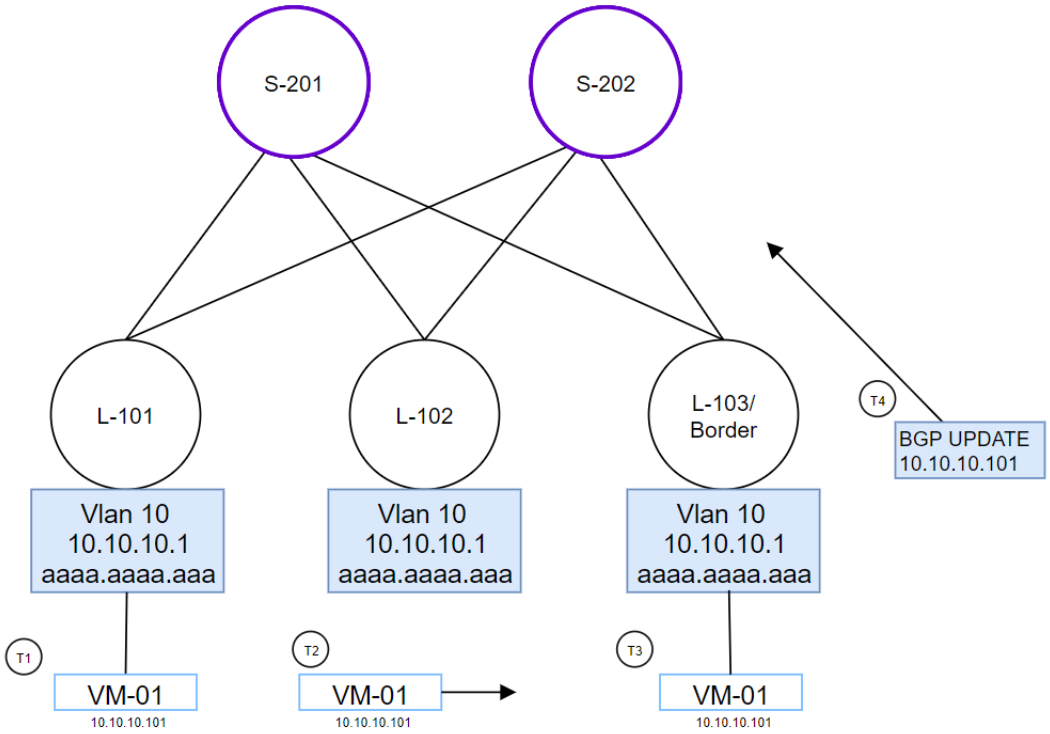
```
#L-101

feature fabric-forwarding

fabric forwarding anycast-gateway-mac aaaa.aaaa.aaaa

interface Vlan10
  description TEN-01 Vlan 10
  no shutdown
  vrf member TEN-01
  ip address 10.10.10.1/24
  fabric forwarding mode anycast-gateway
```

Local VLAN interface with fabric forwarding

The introduction of the fabric forwarding command warrants a separate discussion regarding this feature set.

**Fabric Forwarding**

The Fabric Forwarding feature set, in the context of VXLAN, allows host mobility. The concept is that the VLAN interface(s) share a common MAC address. This means a host can be moved anywhere within the participating fabric and continue forwarding traffic utilizing the same MAC. This negates the need to send out any unnecessary ARP requests and allows host transition without additional configuration. Using an anycast gateway also moves L3 forwarding to every edge node. There is no longer a need to push all Layer 3 routing up to “the core.” The following diagram will be explored to demonstrate Anycast forwarding and the accompanying EVPN behavior. It is important to note that the Anycast Gateway and the Router MAC (RMAC) are two separate components.




Virtual machines move across the fabric

A VM may move for various reasons within the data center compute realm. Anycast gateway aims to alleviate any worry around network transition with said moves. When VM-01 encounters an issue that warrants a move, the VM platform will initiate a “hot move” to another available compute host. With Anycast Gateway, all gateway addresses and MACs are identical across client VLANs, meaning the hosts see no changes in the underlying network. With the burden removed from the host, the network must absorb this responsibility.  In time slot T1, referencing the route table at node L-102, it can be observed that VM-01’s BGP information is as expected. The route is being learned from L-101’s VTEP address (192.168.1.101) relayed by S-201 (10.65.100.201).

```
10200-DC-L102# sh bgp vrf TEN-01 l2vpn evpn 10.10.10.101
BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 10.65.100.102:5      (L3VNI 2000)
BGP routing table entry for
[2]:[0]:[0]:[48]:[000c.2910.811e]:[32]:[10.10.10.101]/272, version 88
Paths: (1 available, best #1)
Flags: (0x000202) (high32 00000000) on xmit-list, is not in l2rib/evpn, is
not in HW

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop
             Imported from
10.65.100.101:32777:[2]:[0]:[0]:[48]:[000c.2910.811e]:[32]:[10.10.10.10
1]/272
  AS-Path: NONE, path sourced internal to AS
      192.168.1.101 (metric 81) from 10.74.0.201 (10.65.100.201)
      Origin IGP, MED not set, localpref 100, weight 0
      Received label 10 2000
      Extcommunity: RT:65500:10 RT:65500:2000 ENCAP:8 Router
MAC:0000.4352.1b08
      Originator: 10.65.100.101 Cluster list: 10.65.100.201

  Path-id 1 not advertised to any peer
```

VM-01 route entry at L-102 prior to move

Moving forward to time slot T3 after the VM has moved viewing the entry from L-102 revelas a new attribute.

```
10200-DC-L102# sh bgp vrf TEN-01 l2vpn evpn 10.10.10.101
BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 10.65.100.102:5      (L3VNI 2000)
BGP routing table entry for
[2]:[0]:[0]:[48]:[000c.2910.811e]:[32]:[10.10.10.101]/272, version 121
Paths: (1 available, best #1)
Flags: (0x000202) (high32 00000000) on xmit-list, is not in l2rib/evpn, is
not in HW

  Advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop
             Imported from
10.65.100.103:32777:[2]:[0]:[0]:[48]:[000c.2910.811e]:[32]:[10.10.10.10
1]/272
  AS-Path: NONE, path sourced internal to AS
      192.168.1.103 (metric 81) from 10.74.0.201 (10.65.100.201)
      Origin IGP, MED not set, localpref 100, weight 0
      Received label 10 2000
      Extcommunity: RT:65500:10 RT:65500:2000 ENCAP:8 MAC Mobility
Sequence:00:1
             Router MAC:0001.2441.1b08
      Originator: 10.65.100.103 Cluster list: 10.65.100.201

  Path-id 1 not advertised to any peer
```

Route entry at L-102 after VM move is completed

The MAC mobility attribute is the single attribute that sorts the issue of MAC moves within the VXLAN fabric. In slot T4, upon receiving traffic from VM-01, node L-103 will realize that the MAC previously learned from node L-101 has moved. This will trigger a route advertisement originating from node L-103 with the new attribute of MAC Mobility Sequence incremented to 1. The sequence number represents the most recent version of the advertisement. With a sequence of 1, the route from L-103 becomes the newest and is installed in the routing table of L-101 and L-102. As demonstrated, the VM incurs no additional configuration and can move freely throughout the fabric. The one basic requirement that may cause issues is that the correct VLAN must be present on the switch to which the host will be migrated. The traffic will be dropped without a valid VLAN interface and fabric forwarding configuration.

**L3VNI Routing**

Returning to the example of Tenant-01 within the VXLAN fabric. Adding the Anycast Gateway at the three respective VLAN interfaces (10,20,30) marks the last step before achieving functional routing.

```
10200-DC-L101# sh bgp l2vpn evpn vni-id 2000 | b Networ

   Network              Next Hop          Metric      LocPrf      Weight
Path
Route Distinguisher: 10.65.100.101:5      (L3VNI 2000)
*>i[2]:[0]:[0]:[48]:[000c.2905.4e06]:[32]:[10.20.20.102]/272
                        192.168.1.102                      100         0 i
*>i[2]:[0]:[0]:[48]:[000c.2988.468d]:[32]:[10.30.30.103]/272
                        192.168.1.103                      100         0 i
```

Host network entries shared to L-101 from L-102 and L-103

At node L-101 both the host routes from L-102 and L-103 have been received. This allows end hosts to participate in inter-vlan routing.

```
tc@box:~$ ifconfig eth0 | grep inet
          inet addr:10.10.10.101  Bcast:10.10.10.255  Mask:255.255.255.0
tc@box:~$
tc@box:~$ ping 10.30.30.103
PING 10.30.30.103 (10.30.30.103): 56 data bytes
64 bytes from 10.30.30.103: seq=0 ttl=62 time=9.997 ms
64 bytes from 10.30.30.103: seq=1 ttl=62 time=7.940 ms
64 bytes from 10.30.30.103: seq=2 ttl=62 time=7.975 ms
```

While end-to-end communication has been established, the routing behavior still requires further investigation to increase understanding—particularly the addition of the Router MAC attribute in the BGP EVPN advertisement.

```
10200-DC-L101# sh bgp vrf TEN-01 ipv4 unicast 10.30.30.103
BGP routing table information for VRF TEN-01, address family IPv4 Unicast
BGP routing table entry for 10.30.30.103/32, version 11
Paths: (1 available, best #1)
Flags: (0x8008001a) (high32 00000000) on xmit-list, is in urib, is best
urib route, is in HW
  vpn: version 16, (0x00000000100002) on xmit-list

  Advertised path-id 1, VPN AF advertised path-id 1
  Path type: internal, path is valid, is best path, no labeled nexthop, in
rib
            Imported from
10.65.100.103:32797:[2]:[0]:[0]:[48]:[000c.2988.468d]:[32]:[10.30.30.10
3]/272
  AS-Path: NONE, path sourced internal to AS
      192.168.1.103 (metric 81) from 10.74.0.201 (10.65.100.201)
      Origin IGP, MED not set, localpref 100, weight 0
      Received label 30 2000
      Extcommunity: RT:65500:30 RT:65500:2000 ENCAP:8 Router
MAC:0001.2441.1b08
      Originator: 10.65.100.103 Cluster list: 10.65.100.201
```

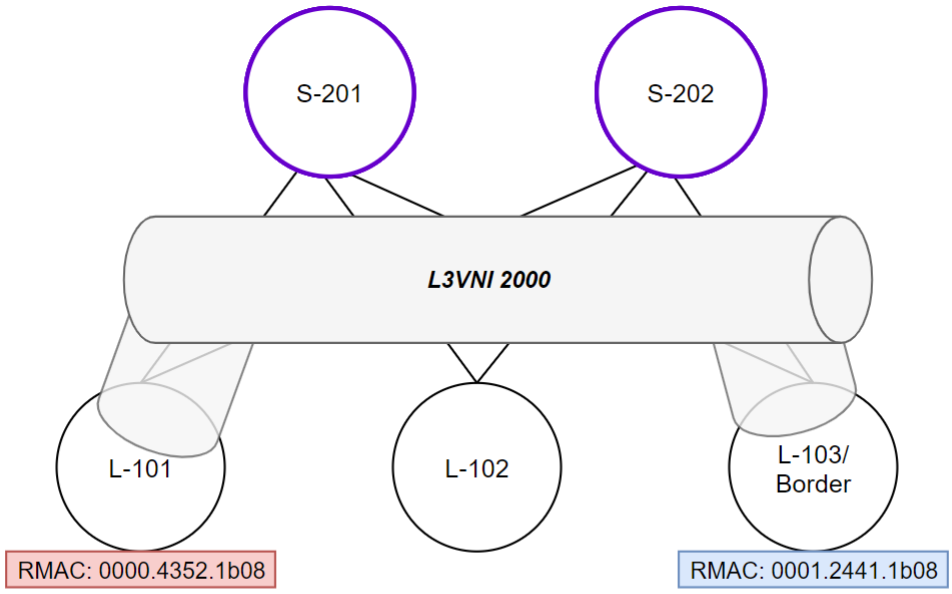
Router MAC (RMAC) attribute in l2vpn evpn advertsiement

The Router MAC value shared via EVPN is the MAC address belonging to the SVI of the L3VNI. In this case, the received Router MAC is from L-103's VLAN 2000 interface.

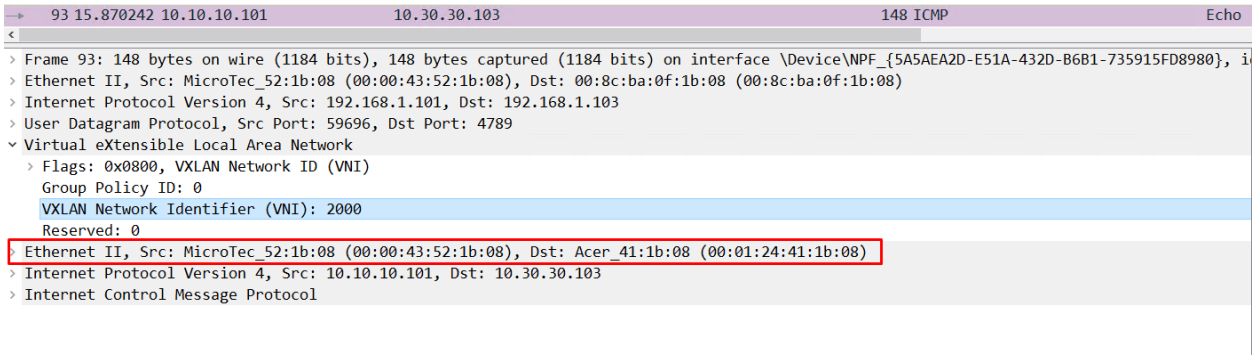
```
10200-DC-L103bgw# sh int vl 2000 | i SVI
  Hardware is EtherSVI, address is  0001.2441.1b08
```

L3VNI VLAN 2000 interface MAC

The Router MAC is used to forward traffic to another VTEP as VXLAN needs inner MAC values. This requires traffic traversing from L-101 to L-103 over L3VNI 2000, which utilizes the Router MAC value received in the BGP update.




L3VNI use in traffic forwarding diagram

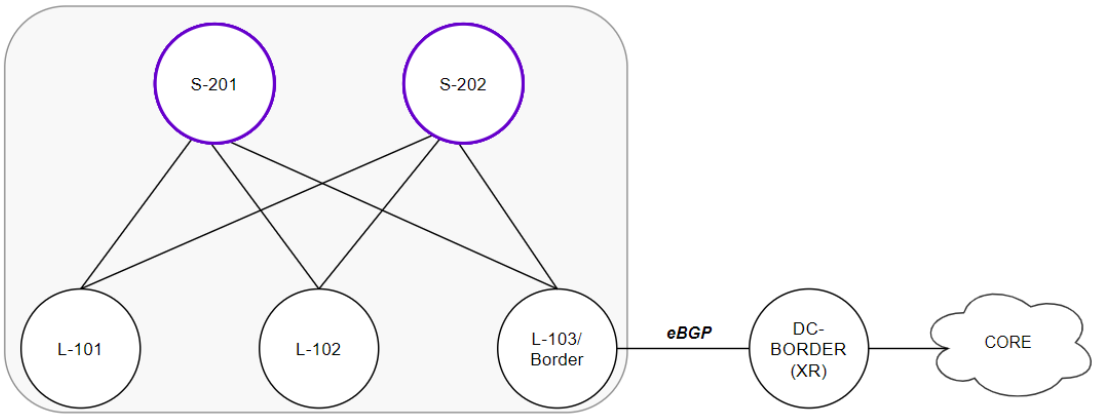



Wireshark capture of RMAC in use

Once the traffic has been received at the remote leaf, a route lookup can occur within the VRF associated with L3VNI 2000 (TEN-01). After the lookup, traffic is traditionally forwarded through the leaf and to the desired endpoint.

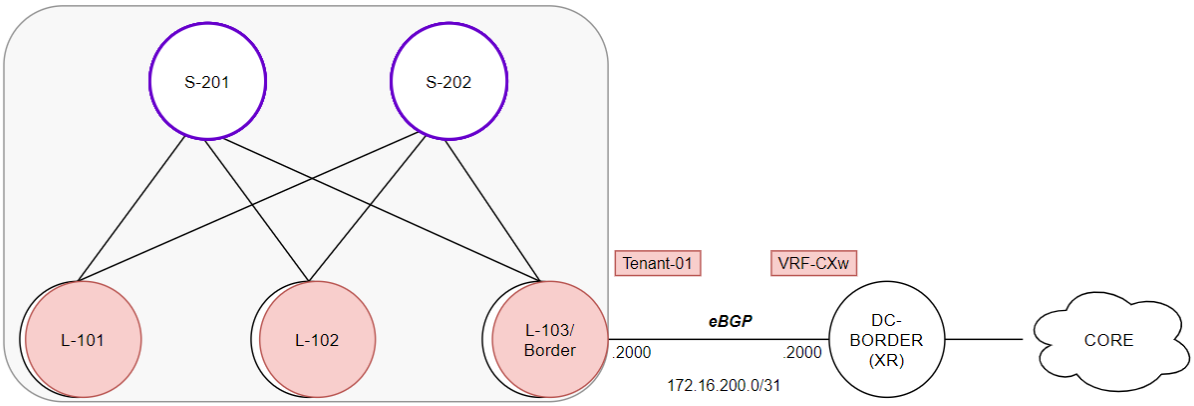
## External Connectivity

Thus far, the methods described have only accounted for traffic within the fabric. To communicate outside the fabric, nodes need to be able to connect and advertise to external destinations. Node L-103 has been designated as the border leaf, which will act as the exit/entry point for the fabric. Due to segment routing and fabric forwarding limitations on Nexus devices, L-103 has been peered directly to an additional IOS-XR router named DC-BORDER.




Data center external connectivity diagram

BGP being the basis for EVPN VXLAN external connectivity is an easy addition to the fabric. The DCBorder node can utilize default IPv4 Unicast peerings or steer MPLS L3VPN traffic to further extend private customer networks to the data center space. The connection between DC-BORDER and L-103 will be a routed link to the Nexus device, and the BGP peering will happen over sub-interfaces. A generic peering is configured over sub-interface 179 and will serve as the global IPv4 peering. Any Tenant VRF can be assigned to a specific sub-interface and retain separation through the entire path when joined with MPLS L3VPN. To demonstrate this ability, the existing VPNv4 network for VRF CXw-24 will be integrated into the TENANAT-01 VXLAN configuration.




L3VPN integration with VXLAN

On node L-103, a subinterface will be configured and associated with the TEN-01 VRF. Once the basic IP configuration is in place, a new neighbor configuration will point toward the DC-BORDER node.

```
interface Ethernet1/9.2000
  encapsulation dot1q 2000
  vrf member TEN-01
  ip address 172.16.200.0/31
  no shutdown

router bgp 65500
vrf TEN-01
      neighbor 172.16.200.1
      remote-as 10200
      address-family ipv4 unicast
```

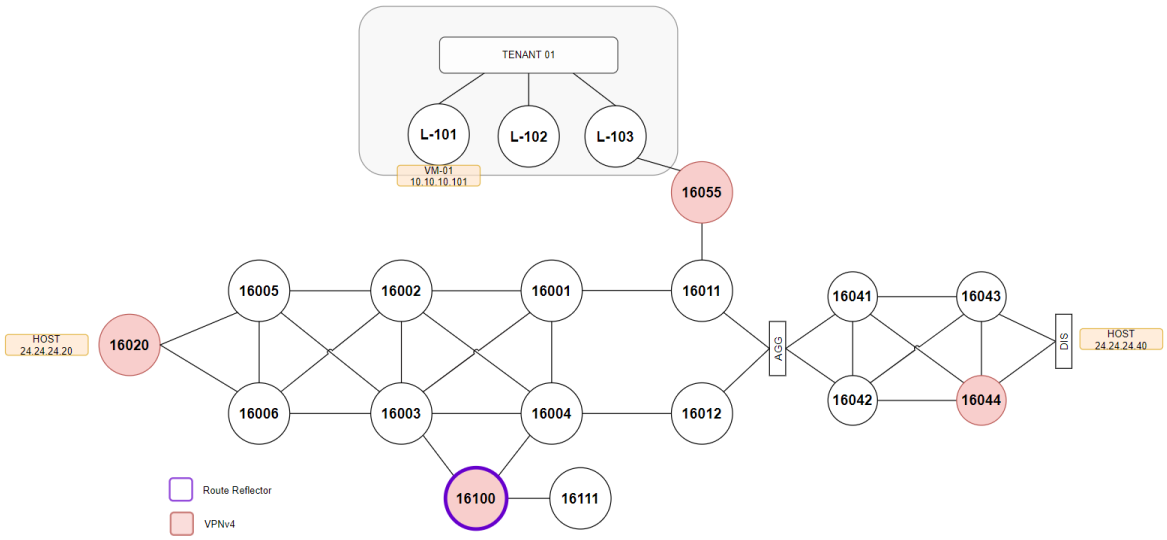
Nexus VRF specific BGP configuration

There is no further configuration needed from the Nexus side to bring this peering up. The focus will now turn to the DC-BORDER XR node.

```
interface GigabitEthernet0/0/0/0.2000
 vrf CXw-24
 ipv4 address 172.16.200.1 255.255.255.254
 encapsulation dot1q 2000
!

router bgp 10200
vrf CXw-24
  rd 55:24
  address-family ipv4 unicast
  !
  neighbor 172.16.200.0
   remote-as 65500
   address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
```

IOS-XR peering configuration to L-103 border node.

The configuration above will establish peering and advertise routes into the VXLAN VRF. This traffic will need to flow across multiple domains in the SP core, which forces the need to engage the PCE node to calculate on-demand paths. The routes from DC-BORDER will be colored with the value of 600 as the on-demand policies already exist in the network.



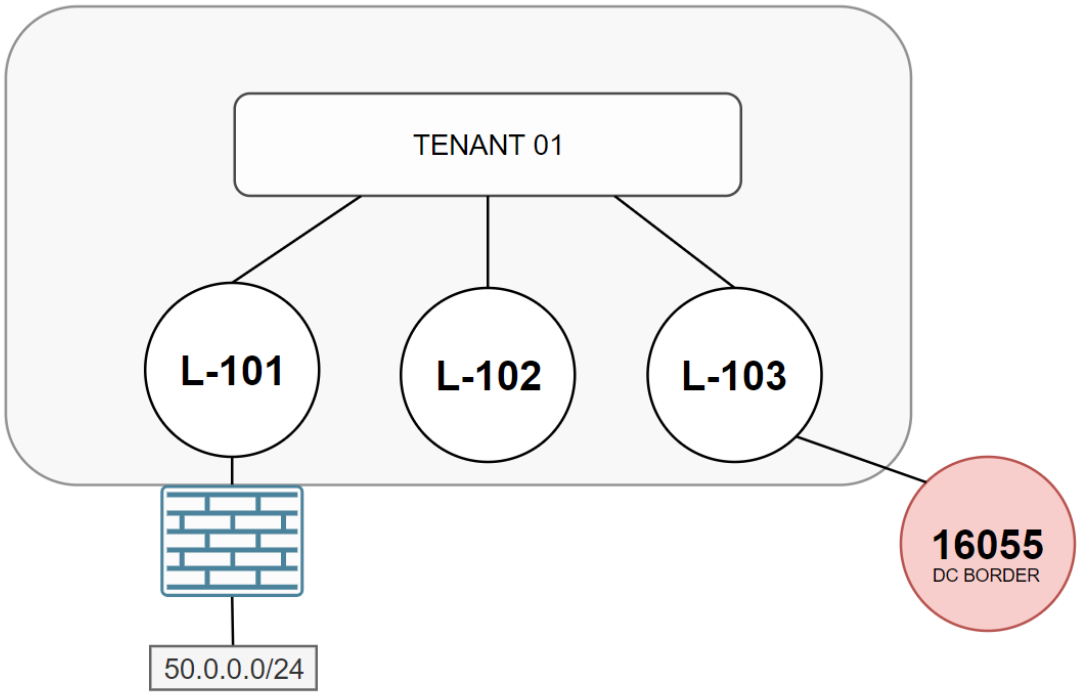

L3VPN VXLAN topology overview

After the PCE distributes the needed SRTE policy, VM-01 can ping both hosts across the domains.

```
tc@box:~$ ifconfig | grep inet
          inet addr:10.10.10.101  Bcast:10.10.10.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
tc@box:~$ ping 24.24.24.20
PING 24.24.24.20 (24.24.24.20): 56 data bytes
64 bytes from 24.24.24.20: seq=0 ttl=247 time=10.740 ms

--- 24.24.24.20 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 10.740/10.740/10.740 ms
tc@box:~$ ping 24.24.24.40
PING 24.24.24.40 (24.24.24.40): 56 data bytes
64 bytes from 24.24.24.40: seq=0 ttl=249 time=11.950 ms
64 bytes from 24.24.24.40: seq=1 ttl=249 time=8.916 ms

--- 24.24.24.40 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 8.916/10.433/11.950 ms
tc@box:~$
```

VM-01 test of external connectivity

The host routes are shared automatically from the VXLAN domain. Additional routes can be shared through traditional network advertisements under the IPv4 Unicast family in the BGP VRF configuration. For instance, if a firewall protected additional networks behind the VXLAN known networks, a BGP advertisement could be made for said networks.




Scenario regarding the need for additional advertisements from VXLAN

The common language of BGP makes it easy to integrate external connectivity to and from VXLAN fabrics. L2VPN EVPN is a powerful address family that has changed how many data centers handle layer two domains and host mobility.

## Orchestration (NSO)

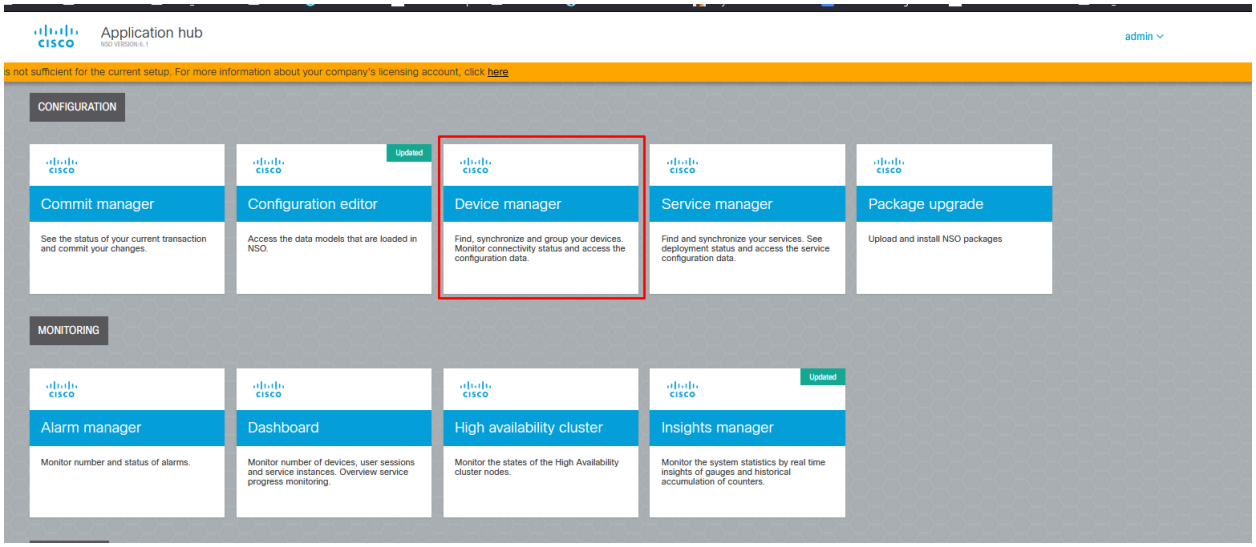
Thus far, the network has been entirely configured via CLI command input. This has worked without issue, but the configuration is tedious even at this small size. When dealing with a provider network of thousands of nodes, it is not feasible to configure everything by hand. Not only is the approach time-consuming, but it is highly prone to error. For the discussion of orchestration, a brief investigation of Cisco's Network Services Orchestrator (NSO) will be outlined in the following section.

NSO is a multi-vendor orchestration tool. The tool can be utilized to push and maintain configuration across devices. The service is relatively involved and ties together CLI alongside GUI-based operation. The goal of NSO is to deliver a model-driven network utilizing yang models. In this demonstration, device import and some simple configuration manipulation and rollback will be reviewed.

The installation of NSO is beyond the scope of this document. Cisco DevNet has provided step-by-step instructions for the entire process. This review will begin with a basic NSO instance installed on a linux host with the services started.

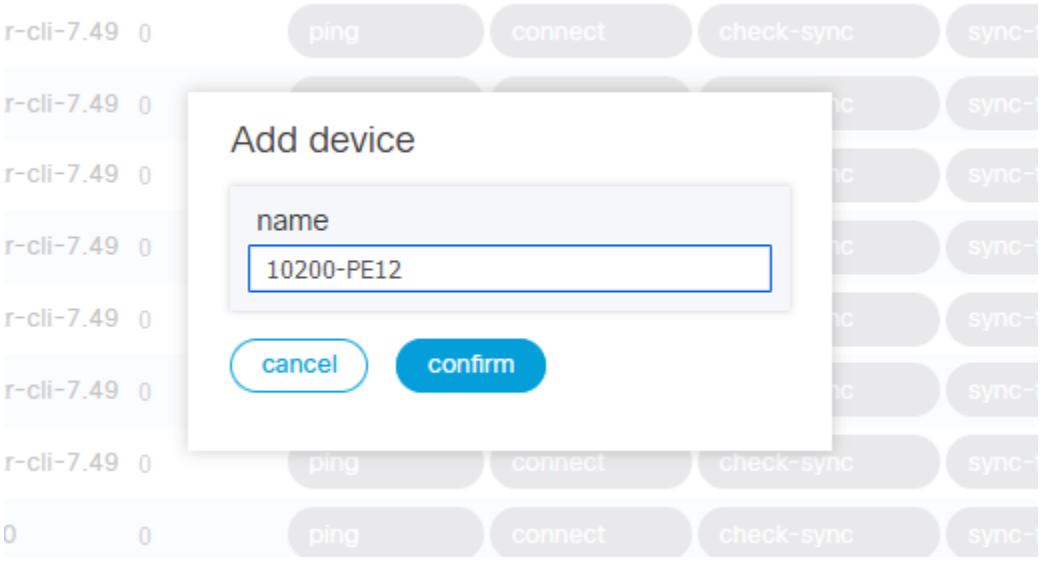
**NSO Device Import**

When accessing via the service GUI, use the tile labeled device manager to begin importing the network devices that NSO needs to monitor/control.



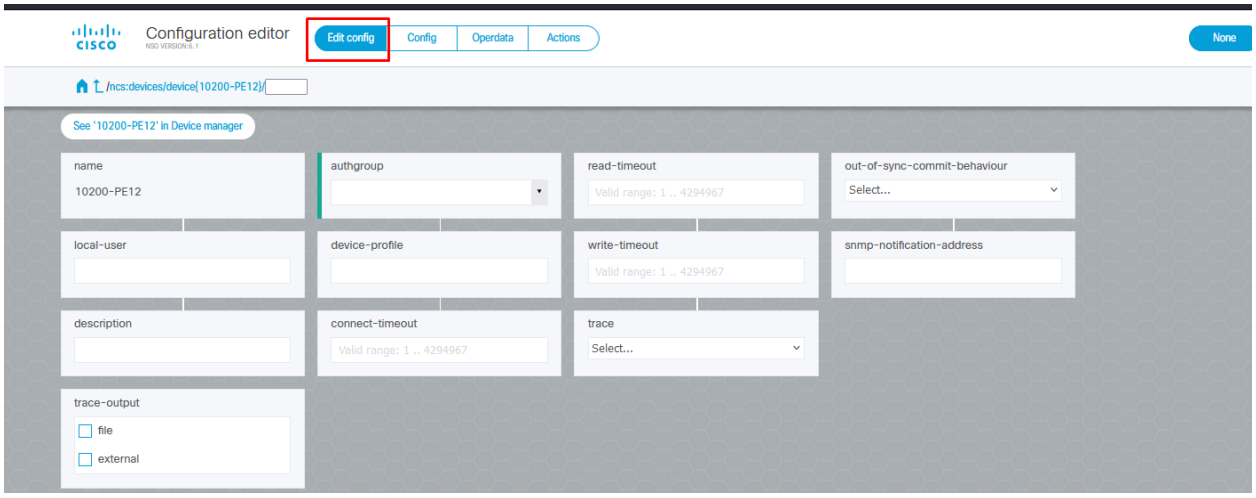

NSO home page

On the top left of the new screen, a blue icon with the “+” sign will be displayed. Clicking that will open up a new device import workflow.




NSO device import

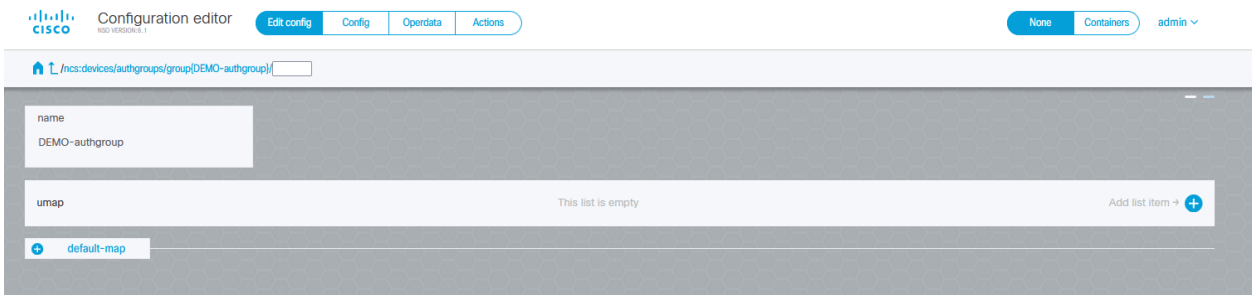
A shell of the device has now been added. Navigating to the device and switching the GUI mode to “Edit-config” will allow the input of the needed details.




NSO device configuration

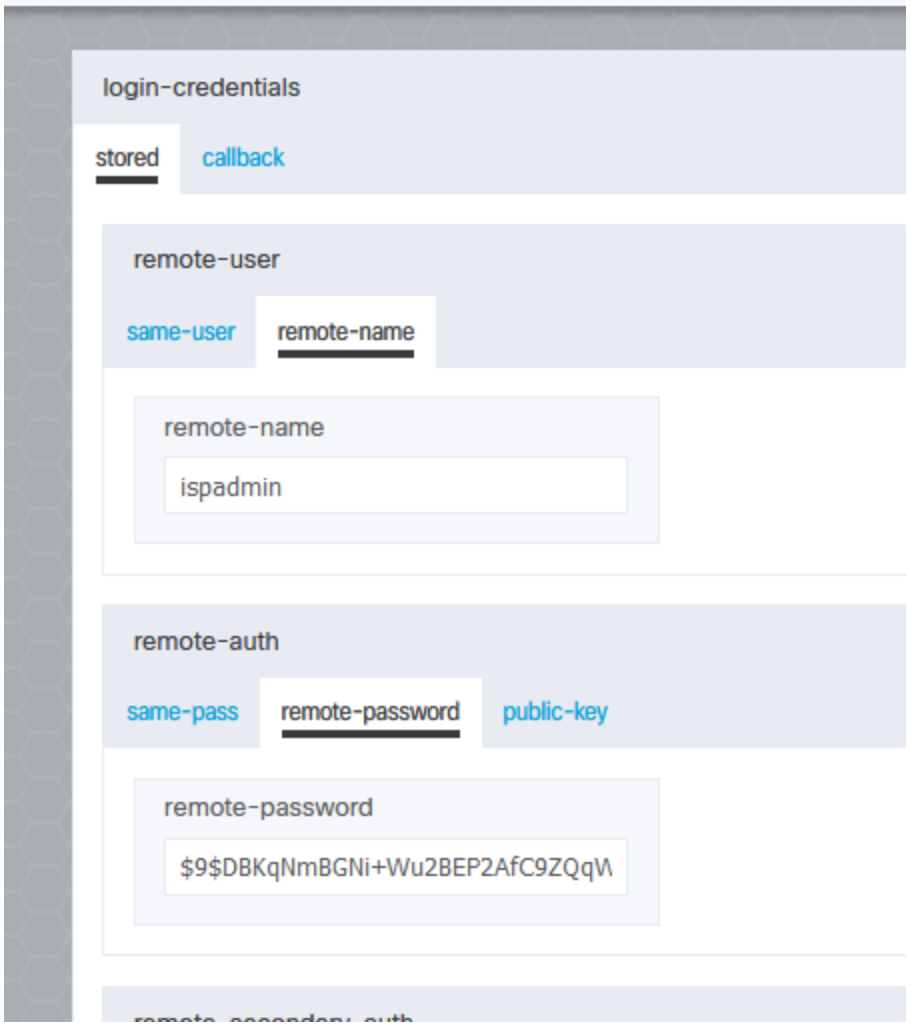
- Authgroup
  The authgroup is where operators can specify credentials to be used for a device.

Using the dropdown, navigate to the “*source”* option, which allows the creation of a new group. This will push the user into a new page where another blue “+” symbol is available to create the new object. Then again, navigate to the group shell that has been made.



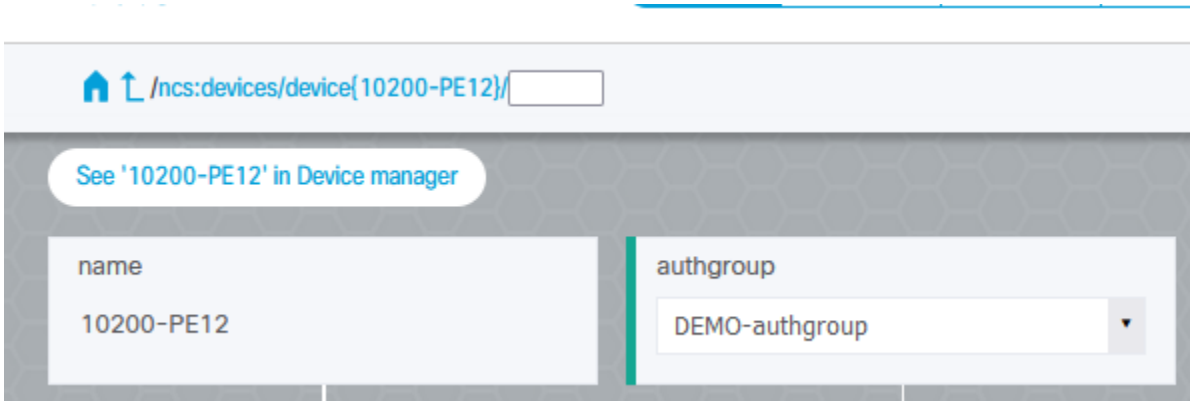

NSO authgroup mapping

Add a new value under the *“default-map”* section and specify a remote user and password that will be utilized to log in via SSH to the device(s) within the group.



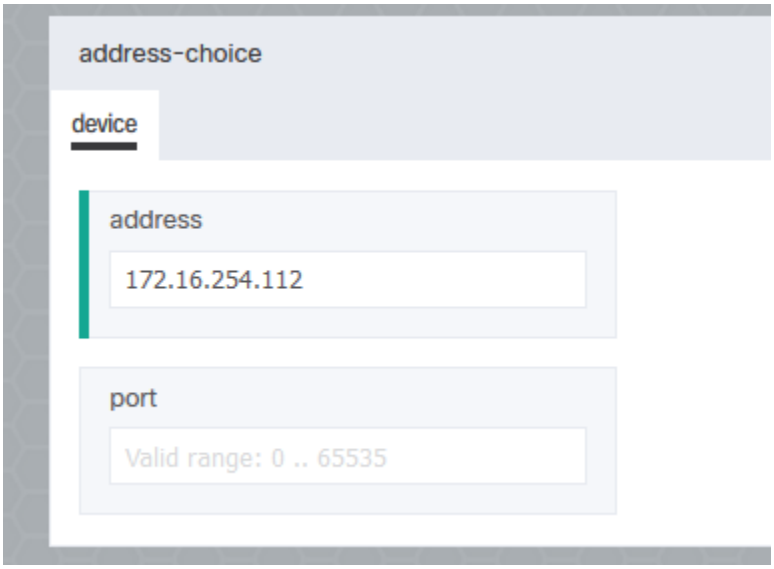

NSO device default user configuration

Use the navigation bar to return to the device configuration, and the new authgroup will be available.

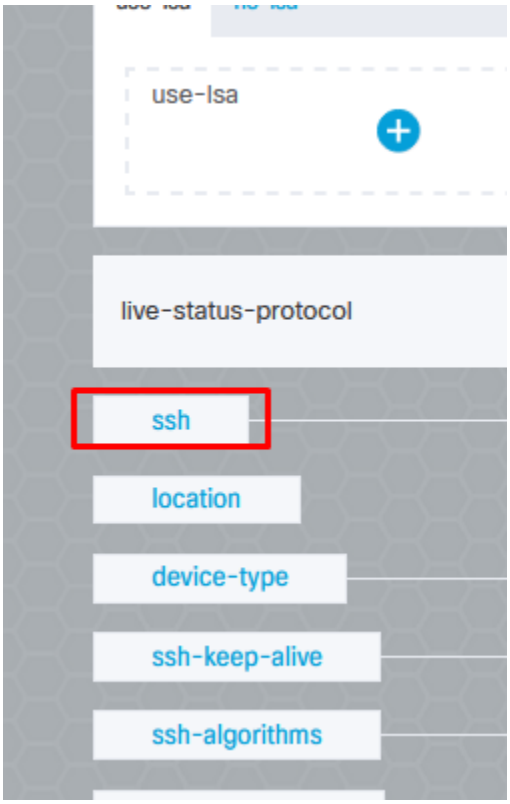



NSO authgroup selection

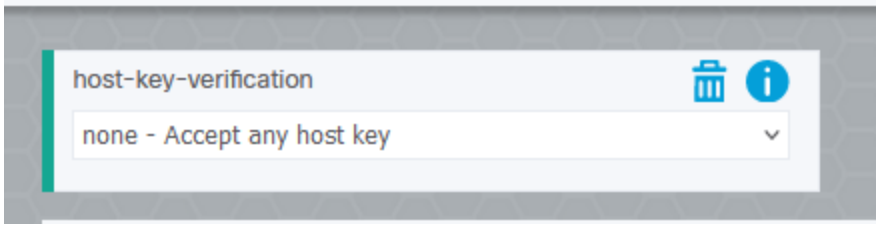
The next value needed for minimum configuration is the address and to turn off SSH key verification.




NSO device address configuration

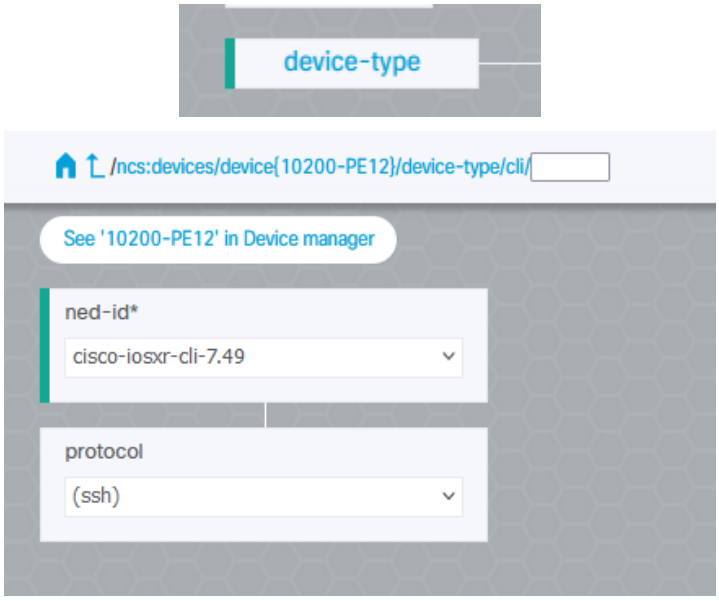



NSO ssh configuration menu



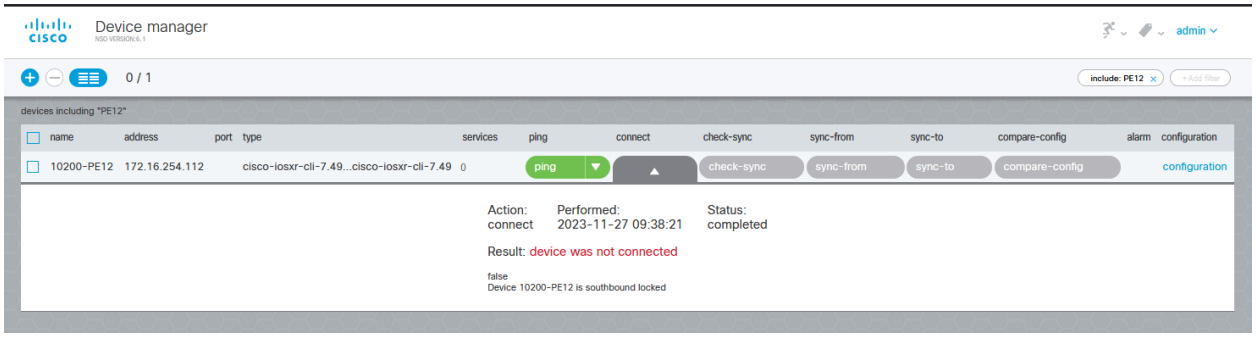

NSO SSH key acceptance

Finally, a device type must be specified. The CLI method will be used in this demonstration, and the device is an IOS-XR router.




NSO device type configuration

Navigate to the commit manager at the bottom of the page and commit the changes. After the commit, the device should ping, but an issue will prevent NSO from logging into the device.




NSO device error

This is where the CLI is needed to *“unlock”* the device. To gain access to the CLI, navigate to the NSO Linux server via SSH and run the following command in the instance.

```
ncs_cli -C -u admin
```

The CLI is similar to a Cisco device regarding behavior. The device state must be changed to resolve this issue with the following commands.

```
config
devices device 10200-PE12 state admin-state unlocked
commit
```

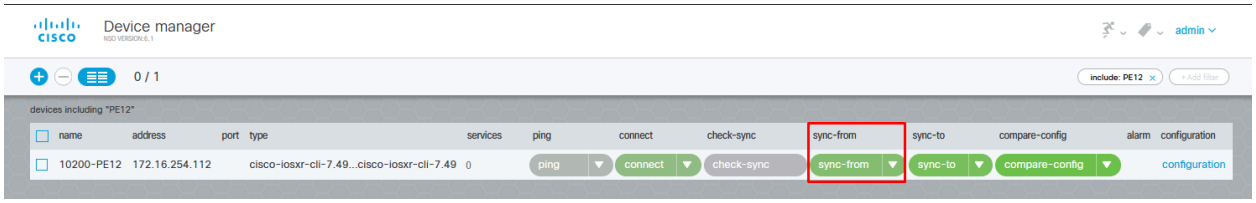
The conenction issue has now been resolved.


NSO can now successfully login

With the device imported, NSO can retrieve device configuration, which will become vital as deployments begin to originate from NSO.

**Configuration Monitoring**

NSO will act as the source of truth for device configuration. An operator can run the sync-from command in the device menu after importing a device. This will retrieve the current running configuration and treat it as the trusted iteration.




NSO sync-from

With this in place, any changes made via CLI or outside of NSO will flag the compare-config action. As a simple example, a loopback will be added via CLI.

```
interface Loopback99
 description UNAUTHORIZED LOOPBACK
```

Unauthorized configuration example

With the change implemented at PE12, a config compare can now be run from NSO, and the unauthorized change will be flagged.


Compare config within NSO identifies config drift

To remedy this issue, a simple sync-to can be run to normalize the configuration.

```
RP/0/RP0/CPU0:10200-PE12#sh run int lo 99
Mon Nov 27 08:02:07.131 UTC
% No such configuration item(s)
```

NSO removes unauthorized configuration

The reverse of this operation can also be achieved via the sync-from. If changes are made via CLI and are authorized, the sync-from command will pull the latest configuration and begin to reference it as the baseline.

## NSO Templates

Using templates grants NSO its ability to orchestrate across multiple platforms. Operators can build a personal repository that fits their needs by configuring templates. A simple loopback configuration will be demonstrated to showcase the ability of NSO templates. It is recommended

that operators be able to pull and edit files from the NSO machine via some form of text editor. In this lab, WinSCP and SublimeText4 have been utilized.

Service templates are implemented via XML and yang. The XML file will specify variables, and the yang model will classify the variable type (string, IP address). It is helpful to see the building blocks in practice rather than theory. The majority of this configuration will be handled via CLI and text editor.

The first step is to create a template shell within the NSO instance. Navigate to the directory of the active NSO instance and continue to the packages folder. A make package command within the folder will be run, and the name will be specified.

```
~/nso-directory/nso-instance/packages

ncs-make-package --service-skeleton template looptemplate
```

NSO package creation

NSO creates the necessary folder and file structure with blank values that can be modified to fit the template needs. As seen in the command above, this is known as a service skeleton.

```
ispadmin@nso:~/ncs-6.1/nso-instance/packages/looptemplate$ ls
package-meta-data.xml  src  templates  test
```

Created folder structure from service skeleton

The *templates* folder is where the XML data is housed, and the *src* folder is designated for yang model information. The next step will be retrieving the configuration structure in XML format. On most Cisco devices, the output can be returned in XML format, but NSO can also run configuration against the device model. The NSO based method will be used via the CLI services.

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# devices device 10200-PE12
admin@ncs(config-device-10200-PE12)# config
admin@ncs(config-config)# interface Loopback 99
admin@ncs(config-if)# ipv4 address 99.99.99.99 /32
admin@ncs(config-if)# top
admin@ncs(config)# show configuration
devices device 10200-PE12
 config
  interface Loopback 99
   ipv4 address 99.99.99.99 /32
   no shutdown
  exit
 !
!
```

NSO based configuration

What is happening is that NSO is allowing the configuration to be run by referencing the device model. This configuration can then be run with a *dry-run* distinction, meaning NSO will verify the command and not implement anything on the device. It is via the dry-run command an operator can retrieve the XML format.

```
admin@ncs(config)# commit dry-run outformat xml
result-xml {
        local-node {
        data <devices xmlns="http://tail-f.com/ns/ncs">
                <device>
                <name>10200-PE12</name>
                <config>
                <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
                        <Loopback>
                        <id>99</id>
                        <ipv4>
                        <address>
                        <ip>99.99.99.99</ip>
                        <mask>/32</mask>
                        </address>
                        </ipv4>
                        </Loopback>
                </interface>
                </config>
                </device>
                </devices>
        }
}
```

Dryrun xml output

With this information obtained the file housed on the NSO server can be edited accordingly. Again, this will be found in the specific template folder.

```
ispadmin@nso:~/ncs-6.1/nso-instance/packages/looptemplate/templates$ ls
looptemplate-template.xml
```

XML template folder path

Opening the looptemplate-template.xml file in the text editor reveals a blank file with some descriptions in place to guide users.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="looptemplate">
  <devices xmlns="http://tail-f.com/ns/ncs">
      <device>
      <!--
            Select the devices from some data structure in the service
            model. In this skeleton the devices are specified in a
leaf-list.
            Select all devices in that leaf-list:
      -->
      <name>{/device}</name>
      <config>
      <!--
            Add device-specific parameters here.
            In this skeleton the service has a leaf "dummy"; use that
            to set something on the device e.g.:
            <ip-address-on-device>{/dummy}</ip-address-on-device>
      -->
      </config>
      </device>
  </devices>
</config-template>
```

Blank NSO XML template

The device section will remain blank as the device is specified upon running the action in NSO. Placing the XML configuration structure within the respective section needs to be done. Below is the configuration used with added variable identifiers (/LOOPNUM,/ipadd)

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="looptemplate">
  <devices xmlns="http://tail-f.com/ns/ncs">
      <device>
      <name>{/device}</name>
      <config>
      <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
            <Loopback>
            <id>{/LOOPNUM}</id>
            <ipv4>
            <address>
            <ip>{/ipadd}</ip>
            <mask>/32</mask>
            </address>
            </ipv4>
            </Loopback>
      </interface>
      </config>
      </device>
  </devices>
</config-template>
```

Loopback XML configuration with added variables

The variables will allow the specification of values at the time of service instantiation. The XML file can then be saved with its original name and file location on the NSO server.  The following configuration is needed to specify the variable value types in the Yang model.

```
module looptemplate {
  namespace "http://com/example/looptemplate";
  prefix looptemplate;

  import ietf-inet-types {
      prefix inet;
  }
  import tailf-ncs {
      prefix ncs;
  }

  list looptemplate {
      key name;

      uses ncs:service-data;
      ncs:servicepoint "looptemplate";

      leaf name {
      type string;
      }

      // may replace this with other ways of refering to the devices.
      leaf-list device {
      type leafref {
      path "/ncs:devices/ncs:device/ncs:name";
      }
      }

      // replace with your own stuff here
      leaf ipadd {
      type inet:ipv4-address;
      }

      leaf LOOPNUM {
      type string;
      }
  }
}
```

NSO yang model to be used for XML template

The yang file handles the variable data type and instructs on how that information should be interpreted. The yang file will also be saved under the same file name in the same location. Once the files are saved on the NSO server, operators must navigate to the *src* folder under the template and run the following command, which activates the yang file.

```
ispadmin@nso:~/ncs-6.1/nso-instance/packages/looptemplate/src$ make
/home/ispadmin/ncs-6.1/bin/ncsc `ls looptemplate-ann.yang  > /dev/null 2>&1
&& echo "-a looptemplate-ann.yang"` \
      --fail-on-warnings \
      \
      -c -o ../load-dir/looptemplate.fxs yang/looptemplate.yang
```

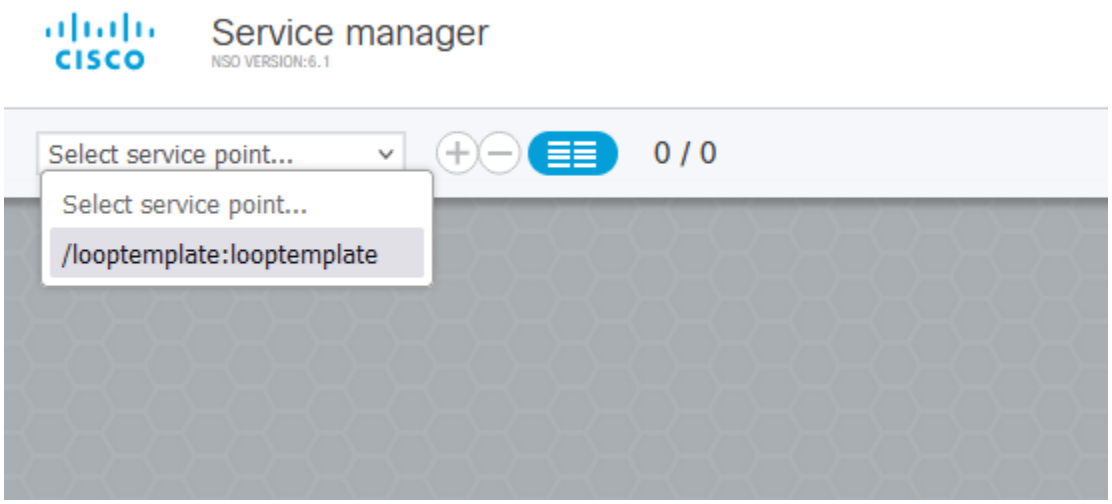
NSO make command

If a fail condition is met, more information will be displayed on the failure in the yang file format. After the make command is completed, a package reload must occur in the NSO CLI.

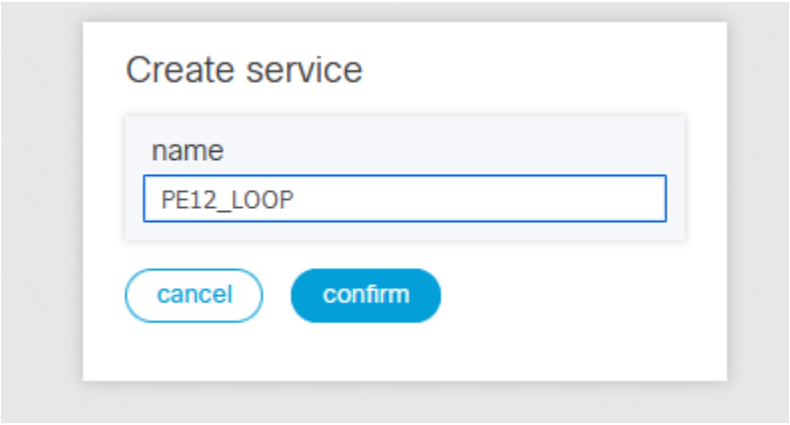
```
ispadmin@nso:~/ncs-6.1/nso-instance/packages/looptemplate/src$ ncs_cli -C
-u admin

User admin last logged in 2023-11-27T17:40:30.62573+00:00, to nso, from
10.101.0.11 using cli-ssh
admin connected from 10.101.0.11 using ssh on nso
admin@ncs# packages reload force
reload-result {
      package cisco-iosxr-cli-3.5
      result true
}
reload-result {
      package cisco-iosxr-cli-7.49
      result true
}
reload-result {
      package cisco-nx-cli-3.0
      result true
}
reload-result {
      package looptemplate
      result true
}
```

NSO package reload command and confirmation

Navigating back to the NSO GUI, under the Service Manager menu the new template is available to deploy.

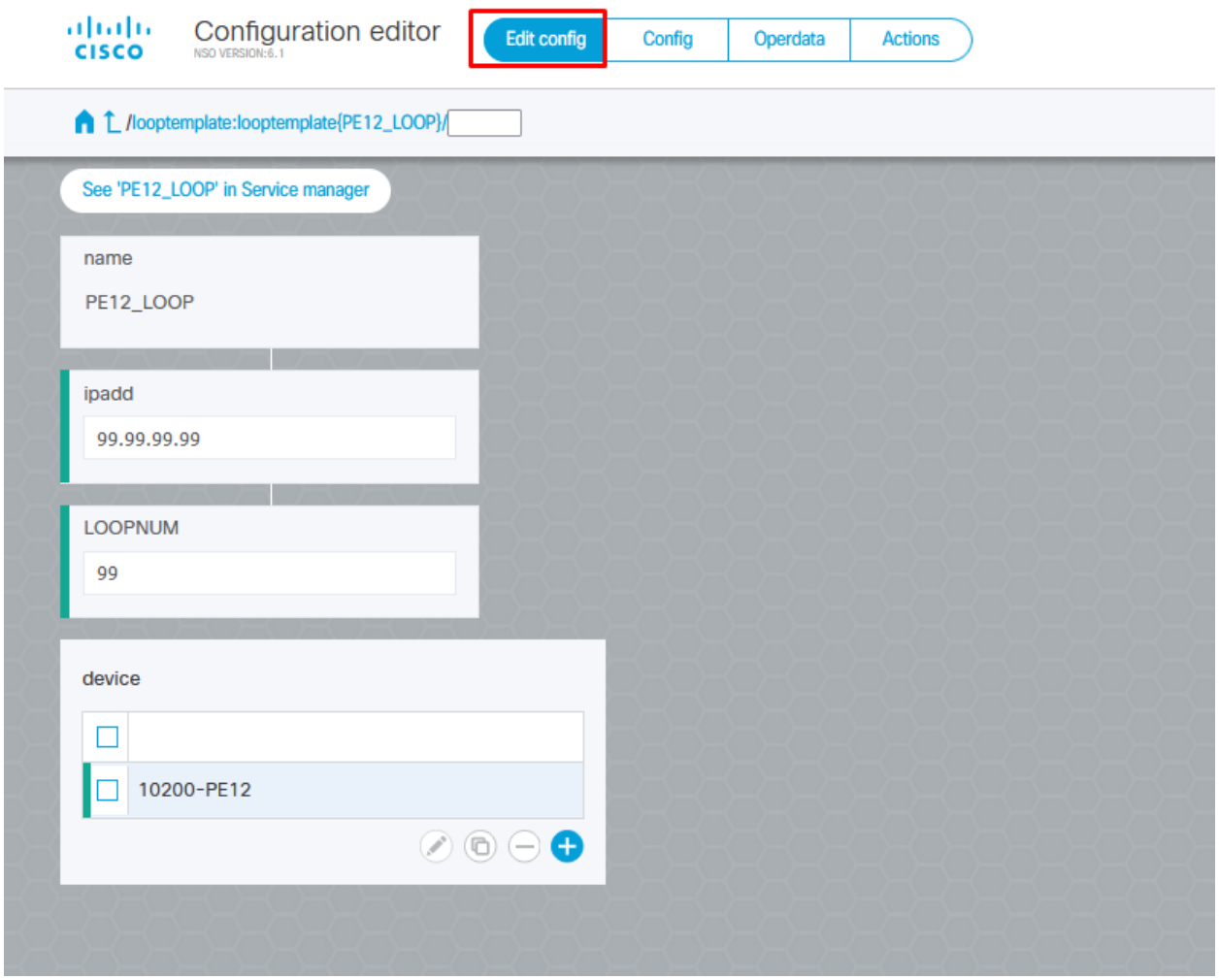



Service manager templates




Template naming nd creation

Provide a user-defined name for the service and confirm the value. This will again create a shell object that can be navigated and configuration values specified.




Service configuration values

A commit will then deploy the service to PE12. Any failures will be displayed within NSO. At PE12, the deployment was successful as the device now has loopback 99 configured.

```
RP/0/RP0/CPU0:10200-PE12#sh run int lo 99
Mon Nov 27 09:57:32.873 UTC
interface Loopback99
 ipv4 address 99.99.99.99 255.255.255.255
!
```

PE12 loopack configuration via NSO

Within the service manager menu for the specific template and device, the Actions tab can be utilized to “un-deploy” the change.


NSO service actions

Scrolling to the bottom of the following page and selecting the run un-deploy action will remove the loopback interface.

```
RP/0/RP0/CPU0:10200-PE12#sh int lo 99
Mon Nov 27 10:03:26.753 UTC
Interface not found (Loopback99)
```

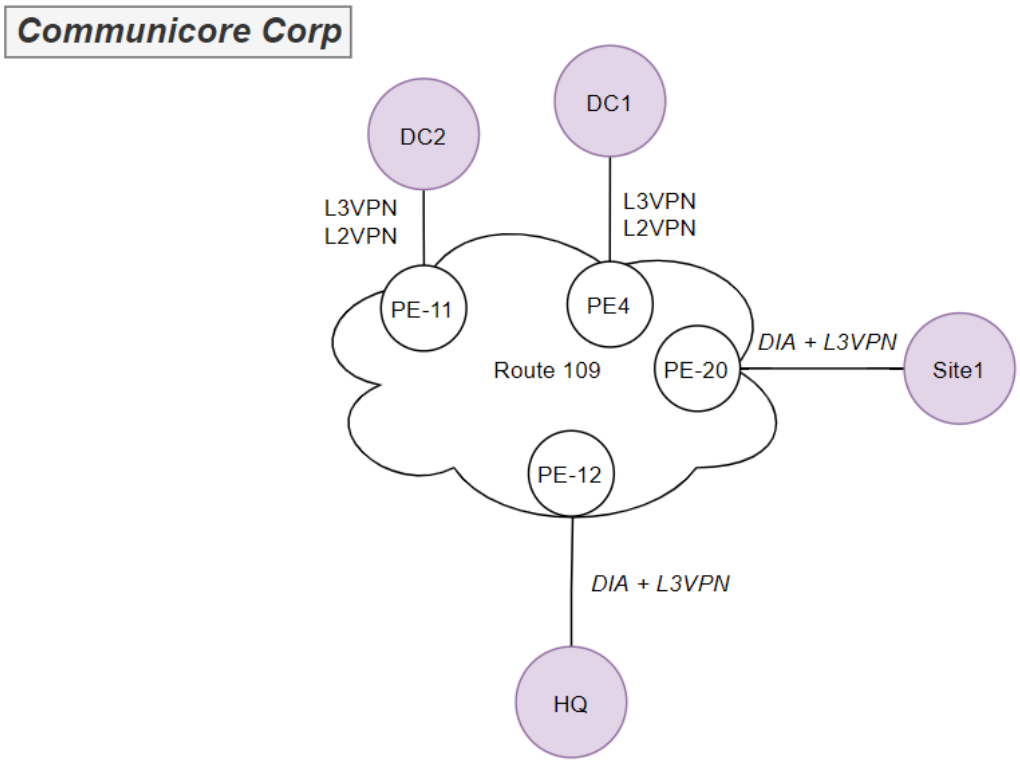
Loopback removed via NSO un-deploy action

NSO templates provide the ability to model a network change for various devices and programmatically implement them. Although this example was simple, the principle can be applied to any configuration.

## Case Studies

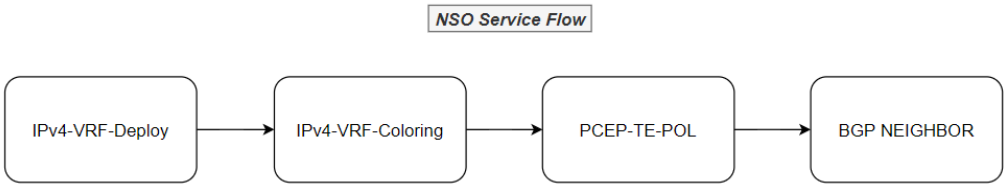
Multiple technologies and methods have been explored throughout this document. A combined demonstration will be examined in the form of simple case studies.

**Enterprise Customer**
In this study, the connectivity requirements of Communicore Corp will be examined. The focus will be on the provider technologies. The nodes used to emulate Communicore Corp will be generic IOS-XE routers.

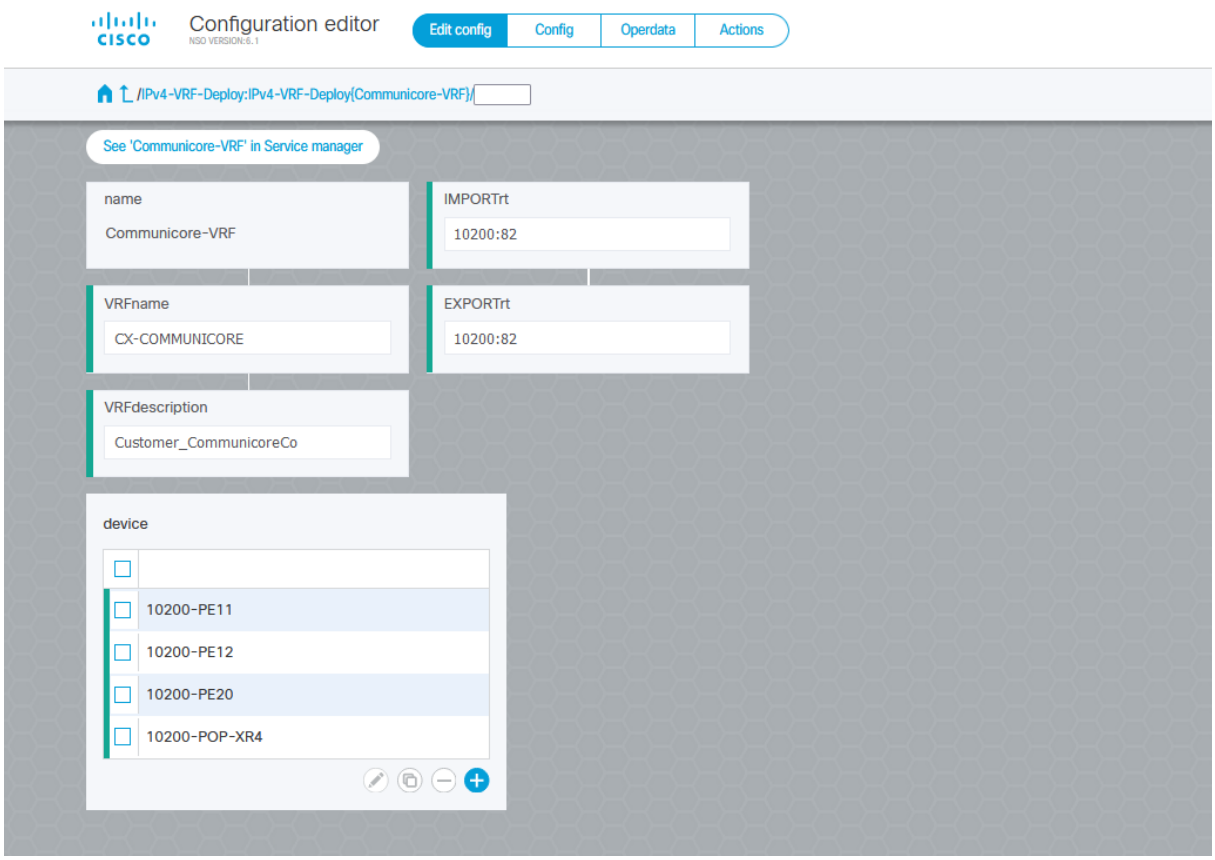



The diagram above illustrates the high-level design as seen by Communicore Corp. The following discussion will follow how the provider will implement the customer requirements.

**HQ**
The HQ office will require direct internet access (DIA) and L3VPN service. This will be implemented via dot1.q subinterface peerings with one interface participating in the global table and the other confined to a VRF. For all sites, the VRF, extended community coloring,on-demand SRTE policy, and BGP peer configuration will be handled by modular NSO services.




Please reference the [GitHub] page for the full XML/Yang files for each service. With NSO, operators can quickly deploy the needed configuration across all included provider edge routers. After VRF creation, the VRF can be called, and a route-policy is assigned at the target export level. Then, an On-demand SR-TE policy is defined to match the color and rely on the PCE to calculate the inter-domain path. Finally, the BGP VPNv4 and IPv4 Unicast details are pushed via NSO.




VRF Deploy to applicable provider edge

Current transaction (14 - webui-one) is VALID  Revert  Load/Save

changes  errors  warnings  config  native config  commit queue

10200-PE11

```
vrf CX-COMMUNICORE
 description Customer_CommunicoreCo
 address-family ipv4 unicast
  import route-target
   10200:82
  exit
  export route-target
   10200:82
  exit
 exit
exit
```

10200-PE12

```
vrf CX-COMMUNICORE
 description Customer_CommunicoreCo
 address-family ipv4 unicast
  import route-target
   10200:82
  exit
  export route-target
   10200:82
  exit
 exit
exit
```

10200-PE20

```
vrf CX-COMMUNICORE
 description Customer_CommunicoreCo
 address-family ipv4 unicast
  import route-target
   10200:82
  exit
  export route-target
   10200:82
  exit
 exit
exit
```

Native configuration displayed within NSO

What would have previously required four separate logins and 10+ lines of configuration is now done from a single screen, error-free. With the VRF in place, the route coloring workflow can be applied to the same PEs.

changes | errors | warnings | config | native config | commit queue

10200-PE11

```
vrf CX-COMMUNICORE
 address-family ipv4 unicast
  export route-policy RP-COL-820
 exit
exit
extcommunity-set opaque COLOR-820
 820
end-set
!
route-policy RP-COL-820
set extcommunity color COLOR-820
 end-policy
!
```

Native VLAN that will be placed at the router level from NSO for route color policy

An On-demand color policy is then pushed referencing the previously configured color applied to the route map.

changes | errors | warnings | config | native config | commit queue

10200-PE11

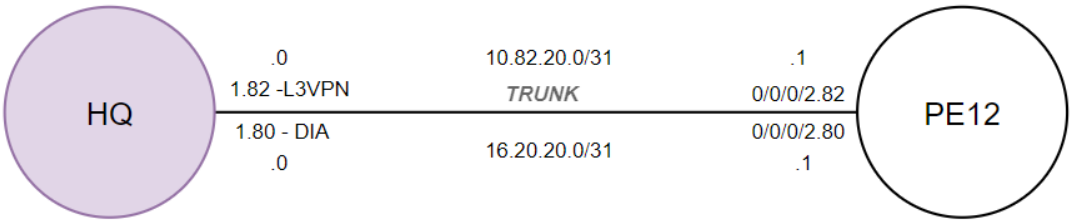
```
segment-routing
 traffic-eng
  on-demand color 820
   dynamic
    pce
    exit
    metric
     type igp
    exit
   exit
  exit
 !
!
```

PCE driven on-demand policy pushed via NSO

Currently, all the VRF and SR-TE configuration is in place. The last step within NSO will be configuring the BGP peerings for the IPv4 and VPNv4 address families.


Native config for IPv4 unicast displayed in NSO

Currently, the only configuration remaining is the local interface configuration at the respective PE. All the BGP signaling and SRTE configurations have been placed via NSO service templates.



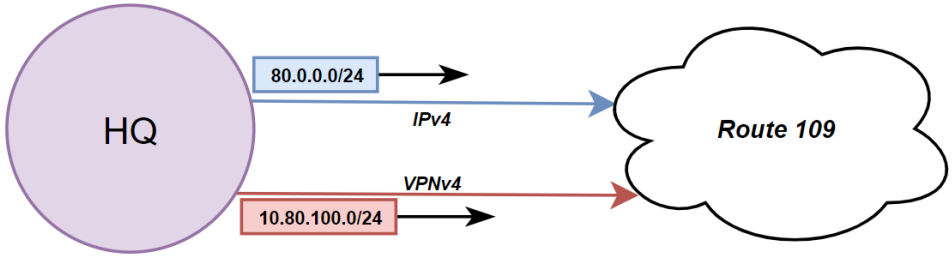

HQ physical topology

Upon configuration of the PE and the HQ router interfaces, both peering sessions are up, and IPv4 routes are being received.

```
COMMUNICORE-HQ#sh bgp summary  | b Neigh
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.82.20.1      4        10200     100     111       12    0    0 01:37:17        0
16.20.20.1      4        10200     107     110       12    0    0 01:36:09       11
COMMUNICORE-HQ#
```

HQ Router IPv4 received prefix

Communicore HQ will advertise the 80.0.0.0/24 and 10.80.100.0/24. Route maps will be applied to both neighbors to control route advertisement toward the correct service type.




HQ route advertisement

```
#COMMUNICORE-HQ Router

ip prefix-list BGPNET seq 5 permit 80.0.0.0/24
!
ip prefix-list MPLS seq 5 permit 10.82.100.0/24
!
route-map mNET permit 10
 match ip address prefix-list MPLS
!
route-map iNET permit 10
 match ip address prefix-list BGPNET

address-family ipv4
  network 80.0.0.0 mask 255.255.255.0
  neighbor 10.82.20.1 activate
  neighbor 10.82.20.1 allowas-in
  neighbor 10.82.20.1 route-map mNET out
  neighbor 16.20.20.1 activate.
  neighbor 16.20.20.1 allowas-in
  neighbor 16.20.20.1 route-map iNET out
 exit-address-family
```

HQ Router BGP configuration

PE12 has now received all the advertised routes from the HQ router.

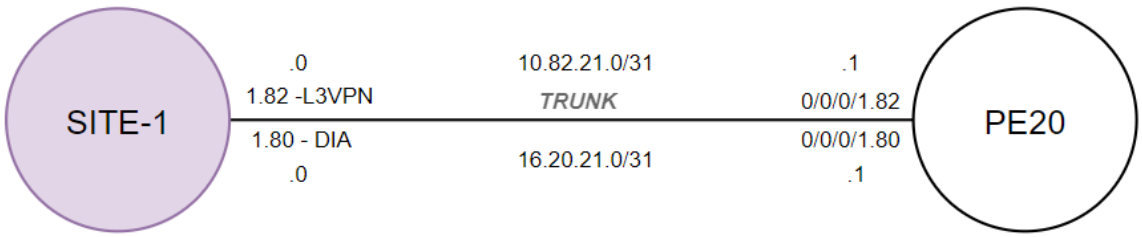
```
RP/0/RP0/CPU0:10200-PE12#sh route vrf CX-COMMUNICORE bgp

B     10.82.100.0/24 [20/0] via 10.82.20.0, 00:00:47


RP/0/RP0/CPU0:10200-PE12#sh ip route | i 80.0.0.0

B     80.0.0.0/24 [20/0] via 16.20.20.0, 02:08:04
```

PE12 VPNv4 and IPv4 routes received from Communicore HQ

At this point the HQ office is configured and focus can move onto the SITE-1 router.

**Remote Site 1**

Communicore's remote site will be another simple implementation of L3VPN and DIA. All the VRF and SRTE policy configurations are already in place from the push made by NSO. The one remaining configuration that will be needed from NSO is the IPv4/VPNv4 peering.



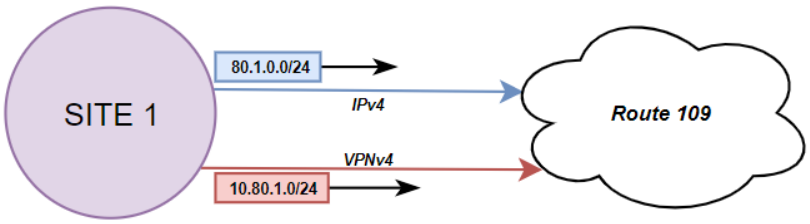

SITE-1 physical topology

Due to an IOS-XR behavior, a special knob is needed to allow Communicore's AS (1082) to receive routes from itself being sent from the provider network. In traditional IOS, the *allowas-in* would permit the routes from 1082 to be imported. With IOS-XR, a check is performed before the advertisement that also checks for loops. This behavior is documented here. The allow-as command is still needed on the remote peer. This additional configuration occurs under the IPv4 address family with the command *as-path-loopcheck out disable*. The command as-override is another option and simplifies the process as routes will be seen as originating from the Route-109 network.  Either way will suffice for a lab environment.

```
COMMUNICORE-SITE1#sh bgp regexp _1082$
% Command accepted but obsolete, unreleased or unsupported; see documentation.

BGP table version is 18, local router ID is 16.20.21.0
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,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>   10.82.100.0/24   10.82.21.1                             0 10200 1082 i
 *>   80.0.0.0/24      16.20.21.1                             0 10200 1082 i
```

Routes received by Remote-Site1

SITE-1 will make two route advertisements. Similar to the HQ router, these advertisements will be scoped down via route-maps to enforce proper separation of the IPv4 and VPNv4 routes.




Site-1 route advertisements

```
ip prefix-list BGPNET seq 5 permit 80.1.0.0/24
!
ip prefix-list MPLS seq 5 permit 10.80.1.0/24
!
route-map mNET permit 10
 match ip address prefix-list MPLS
!
route-map iNET permit 10
 match ip address prefix-list BGPNET
!
address-family ipv4
  network 10.80.1.0 mask 255.255.255.0
  network 80.1.0.0 mask 255.255.255.0
  neighbor 10.82.21.1 route-map mNET out
  neighbor 16.20.21.1 route-map iNET out
```

Site-1 BGP configuration

At this point the SRTE policy is in an active state and SITE-1 and HQ can communicate over both the VPNv4 and IPv4 routes.

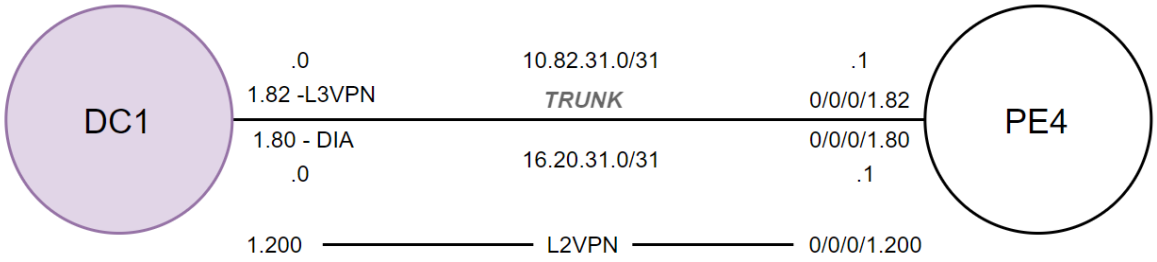
```
RP/0/RP0/CPU0:10200-PE12#sh segment-routing traffic-eng policy color 820
Tue Nov 28 12:56:36.001 UTC

SR-TE policy database
---------------------

Color: 820, End-point: 20.20.20.20
  Name: srte_c_820_ep_20.20.20.20
  Status:
      Admin: up  Operational: up for 00:06:27 (since Nov 28 12:50:08.706)
  Candidate-paths:
      Preference: 200 (BGP ODN) (inactive) (shutdown)
      Requested BSID: dynamic
      Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 10
      Dynamic (inactive)
      Metric Type: IGP,   Path Accumulated Metric: 0
      Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
      PCC info:
      Symbolic name: bgp_c_820_ep_20.20.20.20_discr_100
      PLSP-ID: 1
      Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 10
      Dynamic (pce 111.111.111.111) (valid)
      Metric Type: IGP,   Path Accumulated Metric: 22
            16002 [Prefix-SID, 2.2.2.2]
            16020 [Prefix-SID, 20.20.20.20]
```

SRTE policy active at connected PE to Site 1

**Data Center Interconnect**

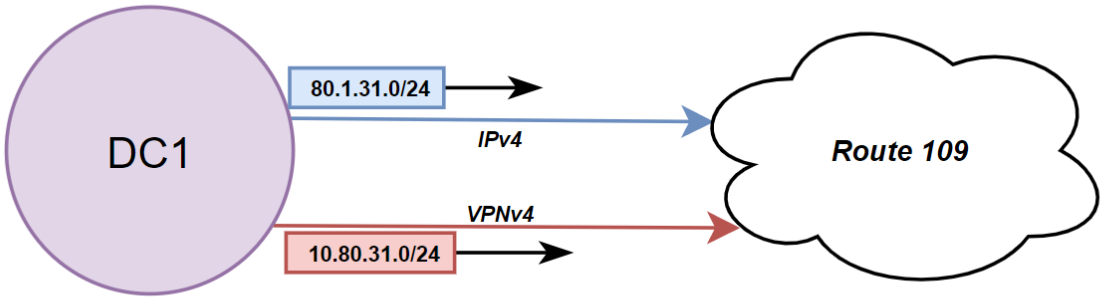
The focus will now shift to providing a data center interconnect to Communicore. A layer 2 interconnect can be used in various ways. The connection demonstrated here will be a point-to-point L2VPN due to the operational limitation of the virtual IOS-XR platform. Over the layer 2 link, Communicore will establish an OSPF peering. This test demonstrates the ability to

pass any traffic through as OSPF will utilize multicast to form a neighbor relationship. Direct peering information is displayed in the diagram below.




DC1 physical connection diagram

Again, NSO will be responsible for deploying IPv4 and VPNv4 peering information on the provider side. DC1 will also share its IPv4 and VPNv4 Unicast routes toward the provider network.

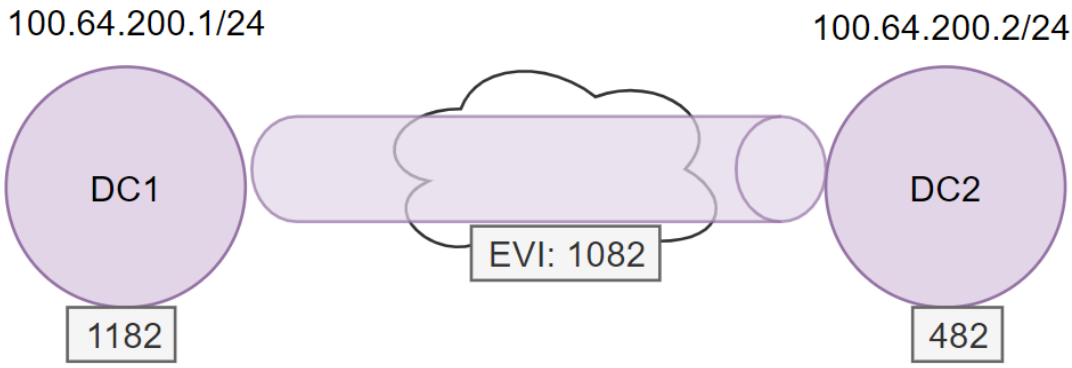



DC1 route advertisement

The BGP configuration in this example is a repeat of what has already been demonstrated. Where the DC connections differ is the use of L2VPN. Another NSO template will deploy the necessary L2VPN information to the involved PE nodes.

```
l2vpn
 pw-class MP
  encapsulation mpls
  !
 !
 xconnect group 4_2_11
  p2p 4211
   interface GigabitEthernet0/0/0/1.200
   neighbor evpn evi 1082 target 1182 source 482
      pw-class MP
```

Provider L2VPN configuration




DC1 - DC2 L2VPN topology

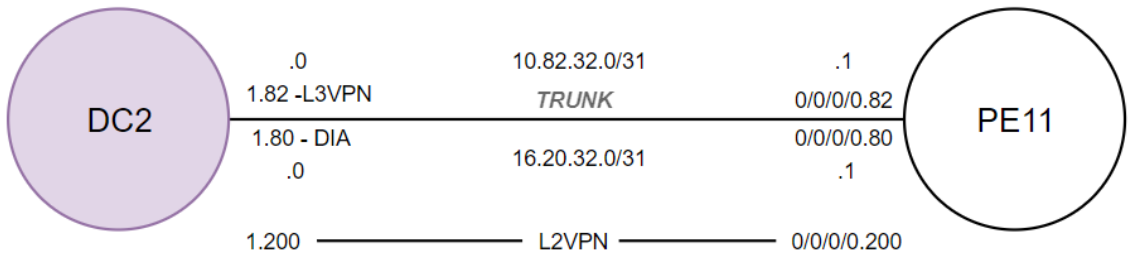
With the basic L2VPN configuration up on both PE nodes (PE11 and PE4) the cross-connect is up. Both DC1 and DC2 can pass traffic as expected.

```
RP/0/RP0/CPU0:POP40-XR4#sh l2vpn xconnect group 4_2_11
Tue Nov 28 21:23:20.400 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
        LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                   Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description            ST
------------------------   ----------------------------   ----------------------------
4_2_11     4211       UP   Gi0/0/0/1.200          UP       EVPN 1082,1182,11.11.11.11
                                                                                  UP
------------------------------------------------------------------------------------------
```

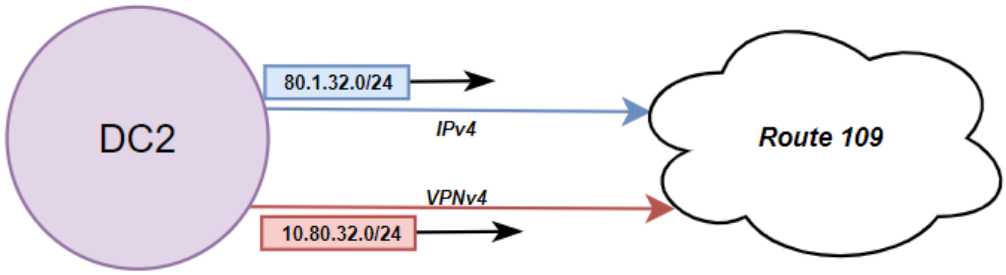
DC1 and DC2 now have a layer 2 adjacency with OSPF layered over the top.

```
COMMUNICORE-DC1#sh ip ospf neighbor
Neighbor ID       Pri   State            Dead Time   Address
Interface
100.64.200.2      1   FULL/DR       00:00:35    100.64.200.2
GigabitEthernet1.200

COMMUNICORE-DC1#traceroute 100.64.200.2
Type escape sequence to abort.
Tracing the route to 100.64.200.2
VRF info: (vrf in name/id, vrf out name/id)
  1 100.64.200.2 3 msec *  5 msec
```

The remaining DC2 peering details have been displayed below.



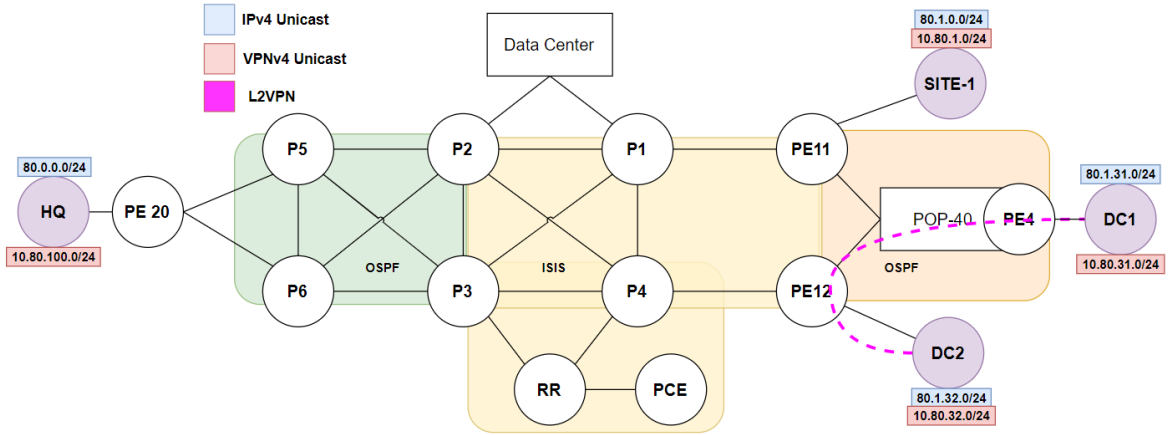

DC2 physical topology




DC2 route advertisement

## Summary

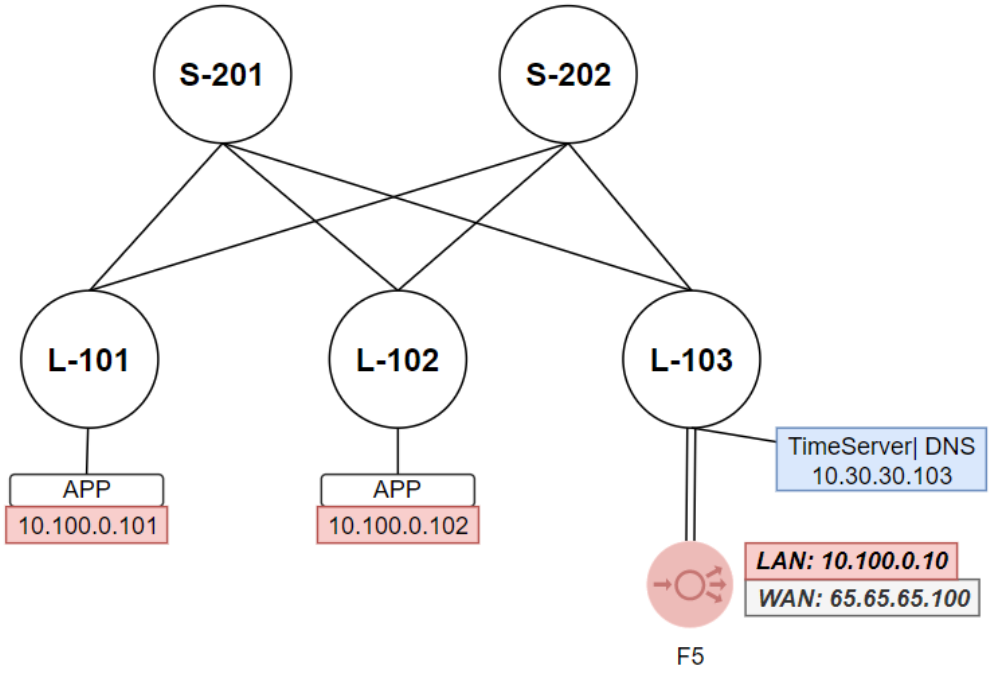
The enterprise customer is now fully connected across all sites via the provider network. This demonstration touched on three major “global”  and private connectivity options. Segment routing was also featured and proven crucial in the inter-domain nature of the traffic flows. NSO increased configuration efficiency and provided standardized deployment methods.




Enterprise Connectivity Overview

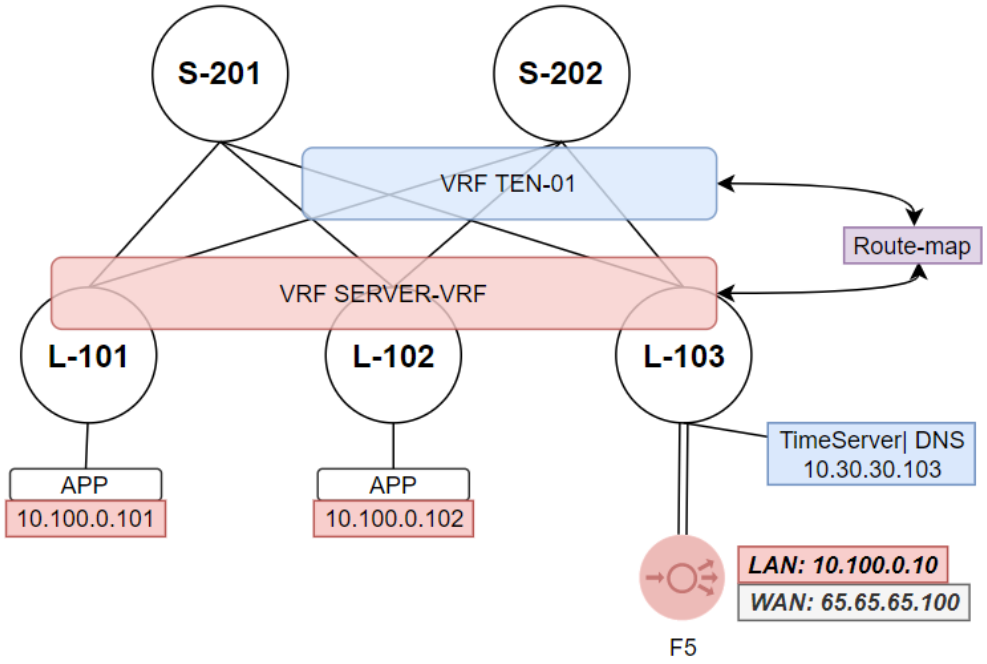
## Data Center Routing

For the Data Center case study, the basic operation of VXLAN will be demonstrated. In the topology, an F5 load balancer will act as the front door into the data center, providing traffic management to an HTTP application.




High-level data center topology

The app servers will allow inbound connections from the F5 and use a *route leak* process to utilize DNS and NTP from a shared resource in another VRF. The 65.65.65.x/24 Route will be advertised via BGP and will allow reachability from external peers via IPv4 Unicast.




VRF route leaking components

The route leaking method is similar to the premise of L3VPN route sharing. The process will utilize the extended community of route targets to import/export routes selectively.
A route map will further define the import/export action. Before the VRF manipulation, both tables only contain their local routes. The further defined objective is to extract 10.30.30.103/32 from TEN-01 and import it into the SERVER-VRF while allowing two-way communication between the VRFs. To control the import process, a prefix list will be applied to VRF SERVER-VRF.

```
ip prefix-list ImportDNS seq 10 permit 10.30.30.103/32
ip prefix-list ImportDNS seq 20 permit 10.100.0.0/24 le 32
```

Import prefix-list

The correct extended communities need to be identified to import the proper routes. Like MPLS, these values will be explicitly called out to trigger the import/export process.

```
10200-DC-L101# sh ip bgp 10.30.30.103 vrf TEN-01 | i Ext
      Extcommunity: RT:65500:30 RT:65500:2000 ENCAP:8 Router
MAC:0001.2441.1b08
```

Viewing the correct target values

```
vrf context SERVER-VRF
  vni 2100
  rd auto
  address-family ipv4 unicast
      route-target both auto
      route-target both auto evpn
      route-target import 65500:2000
      route-target import 65500:2000 evpn
      import map IMPORT-DNS-SERVER
```

VRF configuration needed for import

With the import specified the route for 10.30.30.103/32 is now present in the SERVER-VRF table. This route is identified as an asymmetric entry due to the L3VNI the route possesses differing from the value specified for the SERVER-VRF (2100 vs 2000).

```
10200-DC-L101# sh ip route 10.30.30.103 vrf SERVER-VRF

10.30.30.103/32, ubest/mbest: 1/0
      *via 192.168.1.103%default, [200/0], 00:06:15, bgp-65500, internal,
tag 65500, segid: 2000 (Asymmetric) tunnelid: 0xc0a80167 encap: VXLAN
```

Imported route to SERVER-VRF

A mirrored operation is now needed within the TEN-01 VRF to allow bidirection communication.

```
vrf context TEN-01
  vni 2000
  rd auto
  address-family ipv4 unicast
      route-target both auto
      route-target both auto evpn
      route-target import 65500:2100
      route-target import 65500:2100 evpn
```

Opposite VRF configuration

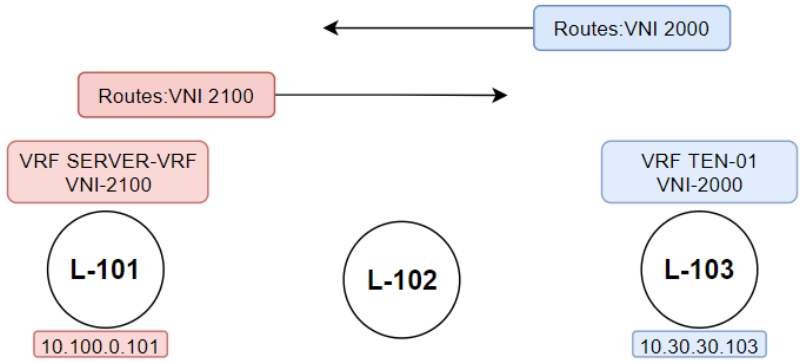
With the configuration in place for both VRFs, specifically at L-101 and L-103 pings can now traverse VRF boundaries.

```
tc@box:~$ ifconfig | grep 10.100
          inet addr:10.100.0.101  Bcast:10.100.0.255  Mask:255.255.255.0
tc@box:~$
tc@box:~$ ping 10.30.30.103
PING 10.30.30.103 (10.30.30.103): 56 data bytes
64 bytes from 10.30.30.103: seq=1 ttl=62 time=17.214 ms
64 bytes from 10.30.30.103: seq=2 ttl=62 time=6.666 ms
```

Pings across VRF boundaries

## Downstream VNI

The previously demonstrated behavior is known as Downstream VNI. Downstream VNI is similar to MPLS in that the egress VTEP advertises the route with a defined label. In this case, the label is the VNI value associated with the VRF.




When traffic is sent from either node destined for the remote VRF, the label used is the received VNI. The remote VTEP receives the traffic and sends it toward the correct instance. So, for this example, traffic from the SERVER-VRF destined toward TEN-01 will use VNI 2000, and the process is the same for return traffic. A wireshark capture confirms this behavior.

| Source | Destination | VXLAN Network Identifier (VNI) | Length | Protocol | Info |
|---|---|---|---|---|---|
| 10.100.0.101 | 10.30.30.103 | 2000 | 148 | ICMP | Echo (ping) request |
| 10.30.30.103 | 10.100.0.101 | 2100 | 148 | ICMP | Echo (ping) reply |
| 10.100.0.101 | 10.30.30.103 | 2000 | 148 | ICMP | Echo (ping) request |
| 10.30.30.103 | 10.100.0.101 | 2100 | 148 | ICMP | Echo (ping) reply |
| 10.100.0.101 | 10.30.30.103 | 2000 | 148 | ICMP | Echo (ping) request |
| 10.30.30.103 | 10.100.0.101 | 2100 | 148 | ICMP | Echo (ping) reply |

```
> Frame 28: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface \Device\NPF_{5A5AEA2D-E51A-4
> Ethernet II, Src: MicroTec_52:1b:08 (00:00:43:52:1b:08), Dst: 00:8c:ba:0f:1b:08 (00:8c:ba:0f:1b:08)
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 192.168.1.103
> User Datagram Protocol, Src Port: 57488, Dst Port: 4789
v Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 2000
    Reserved: 0
> Ethernet II, Src: MicroTec_52:1b:08 (00:00:43:52:1b:08), Dst: Acer_41:1b:08 (00:01:24:41:1b:08)
> Internet Protocol Version 4, Src: 10.100.0.101, Dst: 10.30.30.103
> Internet Control Message Protocol
```

Wireshark capture of VNI behavior

Downstream VNI provides flexibility and eases the operational burden of deploying VRFs. The leaking of routes between VRFs allows tenants to reach shared resources while maintaining separation from the other Tenants. As demonstrated, the separation can be further tuned via route-maps.

## Project Summary

The project aimed to further my understanding of some service provision technologies. The project was also a test of myself to see if I am insane enough to go after a CCIE in this discipline, which I am still unsure about. If you have made it this far, thank you for reading until the end, and I hope this adds some value to your learning experience.

Thank you!

## Appendix Information

All NSO and Device config files can be found on the GitHub page. The configuration files are raw and may contain additional configurations unnecessary to the demonstrations reviewed throughout the document.