CONTENTS

CHAPTER 1  Overview  1
        Evolved Programmable Network  1
        Transport Programmability  2
        Service Programmability  3
        Service Test Topology  3
        Evolved Programmable Network Services  4

CHAPTER 2  Service Orchestration Design  5
        Network Service Orchestrator Model and Mapping Configuration  5

CHAPTER 3  Orchestration Framework  7
        Network Service Orchestrator Package  7
        Key Python Functions  8
        XML Templates  10
        YANG Model  10
        Service Type and Signaling YANG Model Snippet  10
        YANG Model Snippet for Endpoint Service Topology Stitching  11
        YANG Model Snippet for Virtual Forwarding Instance  14
        YANG Model Snippet for Bridge Group or Domain  15
        YANG Model Snippet for Layer-2 Virtual Private Network XC  16
        YANG Model Snippet for Ethernet Virtual Private Network  16
        YANG Model Snippet for Provider Backbone Bridging Ethernet Virtual Private Network  17
        YANG Model Snippet for Pseudowire Interface  18
        YANG Model Snippet for Y.1564 Service Activation  19
        Python Mapping Logic  20

CHAPTER 4  Software Versions  23
        Network Service Orchestrator Software Version  23
CHAPTER 5

Service Orchestration – Sample Configurations 25

Virtual Private Wire Service with Label Distribution Protocol Signaling 25
Virtual Private Wire Service with No-Signaling 27
Ethernet Virtual Private Network Based Virtual Private Wire Service 29
Virtual Private LAN Service with Label Distribution Protocol Signaling 31
Virtual Private LAN Service with BGP Active Discovery Signaling 33
Virtual Private LAN Service Using Label Distribution Protocol with Termination on L2 Ring 36
Virtual Private LAN Service Using BGP – Active Discovery Signaling with Termination on L2 Ring 39
Layer-3 Virtual Private Network 42
Layer-3 Virtual Private Network with Termination on L2 Ring 46
Pseudowire Headend Based Layer-3 Virtual Private LAN Service 48
Hierarchical Virtual Private LAN Service with LDP Signaling 53
Hierarchical Virtual Private LAN Service with BGP Active Discovery Signaling 57
Provider Backbone Bridging Ethernet Virtual Private Network 61

CHAPTER 6

Detailed Configuration Steps for Virtual Private Wire Service with Label Distribution Protocol Signaling 65

Configuration Steps 65
Final Network Service Orchestrator Configuration CLI Set 68
Native Configuration Created by Network Service Orchestrator 68
View Service Configuration 68
Network Service Orchestrator Based Service Assurance 69
Service Removal Using Network Service Orchestrator 74
Service Augmentation 75

References 77
Overview

This chapter contains the following sections:

- Evolved Programmable Network, page 1
- Service Test Topology, page 3
- Evolved Programmable Network Services, page 4

Evolved Programmable Network

The Cisco Evolved Programmable Network (EPN) is built towards the successful Cisco EPN architecture framework to bring greater programmability and automation. The Cisco EPN system design follows a layered design to simplify the end-to-end transport and service architecture. By decoupling the transport and service infrastructure layers of the network, it allows the two distinct entities to be provisioned and managed independently. The Cisco EPN allows the programmatic interaction between the service and transport layers.

Figure 1: Evolved Programmable Network Architecture
This guide focuses on the design aspect of the service and transport programmability using the Network Service Orchestrator (NSO). The transport layer provides the framework to achieve connectivity among two or more nodes in the network, and enables all the consumer and enterprise services that the Cisco EPN system promotes.

The Cisco EPN system incorporates a network architecture designed to consolidate multiples services on a single Multiprotocol Label Switching (MPLS) transport network. This network is designed primarily based on Application Engineered Routing (AER), to optionally co-exist with the traditional MPLS technology. The service programmability is achieved with NSO, while ensuring the transport programmability using the XTC.

**Figure 2: Network Programmability—Evolution**

Transport Programmability

This programmable transport option is Software-defined networking (SDN) driven. Each domain runs IGP segment routing across the domain. Two IGP border routers in each domain uses the BGP link state (LS) to feed the topology, bandwidth, reliability, latency, Shared Risk Link Groups (SRLG) and other transport state of the IGP domain to the SDN controller. The SDN controller uses the topology data and the current state of the network, to build the best path and alternate disjoint path that satisfies a given service requirement and pushes the corresponding segment list to the service edge router.

For more details, see Programmable Transport section in the *Transport Design Guide*. 
Service Programmability

The NSO assumes that the underlying transport network is pre-configured manually and the nodes have been added and synchronized with the NSO. It is recommended to do all the service creation using NSO, to ensure that the service database is consistent and the devices are synchronized with the NSO.

The NSO to device communication is using CLI Network element driver (NED). For information on NED version, see Network Service Orchestrator Software Version, on page 23. The devices can be accessed by 'telnet' or 'ssh'. When using 'ssh', it is required to fetch the corresponding 'host-key' from the device.

**Note**

For information on the transport configuration, refer to the appropriate EPN Transport guide listed in the References section.

Service Test Topology

The development process for the Cisco EPN orchestration package ensures the validation of various functional aspects of the system design. The service validation is conducted on a test bed designed to emulate the characteristics of a converged operator's production network environment. The details of the system test bed are illustrated in the Transport Design Guides.

*Figure 3: Service Test Topology*
Evolved Programmable Network Services

The Cisco EPN supports the orchestration of following services using NSO:

- Virtual Private Wire Service (VPWS) with signaling options such as Ethernet Virtual Private Network (EVPN), static and Label Distribution Protocol (LDP) signaling.
- Virtual Private LAN Service (VPLS) with signaling options such as LDP signaling and BGP auto-discovery.
- Hierarchical Virtual Private LAN Services (H-VPLS).
- Layer-3 Virtual Private Network (L3-VPN).
- Psuedowire Headend (PWHE) based L3VPN.
- VPWS using Provider Backbone Bridging Ethernet VPN (PBB-EVPN).

In addition, the Cisco EPN supports the termination on l2_ring_endpoints on REP/G.8032 rings for the VPWS and VPLS services.
Service Orchestration Design

This chapter contains the following section:

- Network Service Orchestrator Model and Mapping Configuration, page 5

Network Service Orchestrator Model and Mapping Configuration

A service in NSO consists of the following:

- Yet Another Next Generation (YANG) service model—This defines the attributes of the service, for example, a Layer-2 VPN service might be defined with virtual circuit id, service identifiers, and interface names. This YANG model renders the NSO CLI and Web-based User Interface (UI).
• Mapping to the device configuration—This allows the required settings to be configured on the devices when a service is created.

**Figure 4: Service Orchestration Design**

To create a new service or implement the device mapping configuration, the following logic are used:

• Mapping logic—It is used to identify the service related operations applicable for the devices, among the available operations. The identified service operation is then configured on devices. For example, when you add an access link to a VPN, the mapping logic would determine the configuration settings to be done on the Customer Edge (CE) router and Provider Edge (PE) router.

• Validation logic—Most of the validation can be expressed in the YANG models. For certain cases, the data validation needs to be done using the external code. For example, to perform a look-ups in databases, the Management Agent Application Programming Interface (MAAPI) can be used.
Orchestration Framework

The Cisco EPN orchestration framework is based on NSO release 4.3.1, and uses Python for service logic and XML templates. The services are bundled in a single package, which contains the python code and the templates for demonstrating the EPN services with basic functionality. The package can be augmented to support additional parameters, additional functionality, and so on, if required.

To provide feedback about the additional functionality that you would like to have for general availability, reach out to epn-csg@cisco.com.

This chapter contains the following sections:

- Network Service Orchestrator Package, page 7
- Key Python Functions, page 8
- XML Templates, page 10
- YANG Model, page 10
- Python Mapping Logic, page 20

Network Service Orchestrator Package

The Cisco EPN NSO models and mapping logic are bundled in the package “EPN5_0”. You have to request access to the models through the account team. The package has been compiled with NSO 4.3.1 and might need to be recompiled for subsequent releases and template dependencies.

To compile a package, see NSO Beginner guide.

The tree structure of the NSO package is given below:

```
doc
  load-dir
    epn5_0_services.fxs
    package-meta-data.xml
  python
    epn5_0_services.py
  README
  src
```
# Key Python Functions

The key Python functions are given in the below table:

<table>
<thead>
<tr>
<th>Function Name and Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>config_l2_ring_service</strong> (self, root, service, l2_ring_endpoints, serv_endpoints)</td>
<td>This method is used to associate a L2 ring based service termination point with the core network.</td>
</tr>
<tr>
<td><strong>config_l2vpn_pbb_evpn_assoc_list</strong>(self, root, service)</td>
<td>This method is used to associate PBB EVPN endpoint with the core network.</td>
</tr>
<tr>
<td><strong>config_pbb_evpn_vpws</strong>(self, root, service, serv_name, pe_list)</td>
<td>This method is used to configure PBB EVPN VPWS leg to PBB core.</td>
</tr>
<tr>
<td><strong>setup_pbb_evpn</strong>(self, service, pe, l2_serv_name, as_no, l2_serv_inst_id, rem_pe_lpbk_ip, pw_id_suffix)</td>
<td>Provides the XML template configuration for L2VPN PBB EVPN.</td>
</tr>
<tr>
<td><strong>config_l3vpn_pwhe</strong>(self, root, service)</td>
<td>Provides the L3VPN PWHE basic configuration.</td>
</tr>
<tr>
<td><strong>config_l3vpn_policy</strong>(self, root, service, serv_name, pe_list, local_policy_pe, rem_policy_pe, pe_bgp_as_no, pe_lpbk_ip)</td>
<td>Provides the L3VPN policy configuration for the SR-ODN use case.</td>
</tr>
<tr>
<td><strong>config_pwhe_vpws</strong>(self, root, service, serv_name, pe_list)</td>
<td>This method is used to associate a VPWS with the PWHE terminates on the IOS-XR node.</td>
</tr>
<tr>
<td><strong>config_l3vpn_base</strong>(self, root, service, serv_endpoints)</td>
<td>This method is used to configure the basic L3VPN.</td>
</tr>
<tr>
<td>Function Name and Attributes</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>config_l2vpn_h_vpls</td>
<td>This method is used to configure the H-VPLS core.</td>
</tr>
<tr>
<td>config_l2vpn_vpls_base</td>
<td>This method is used to configure the basic VPLS. It is also invoked when the IP-SLA test is started or stopped.</td>
</tr>
<tr>
<td>config_l2vpn_vpws</td>
<td>This method is used to configure basic VPWS. It is also invoked when the IP-SLA test is started or stopped.</td>
</tr>
<tr>
<td>config_l2vpn_si</td>
<td>This method is used to configure l2vpn service instance and populate the XML template.</td>
</tr>
<tr>
<td>setup_l3vpn_vrf</td>
<td>This method is used to configure l3VPN VRF and populate the XML template.</td>
</tr>
<tr>
<td>setup_vfi</td>
<td>Provides XML template configuration for L2VPN VFI.</td>
</tr>
<tr>
<td>setup_bd</td>
<td>Provides XML template configuration for L2VPN BD.</td>
</tr>
<tr>
<td>setup_xc</td>
<td>Provides XML template configuration for L2VPN XC.</td>
</tr>
<tr>
<td>setup_pw_if</td>
<td>Provides XML template configuration for L2VPN PBB EVPN.</td>
</tr>
<tr>
<td>setup_bgp_prefix_list</td>
<td>This method is used to configure BGP prefix list on individual PE for outbound route filtering.</td>
</tr>
<tr>
<td>def setup_evpn_p2p</td>
<td>Provides XML template configuration for EVPN.</td>
</tr>
<tr>
<td>config_sa_ip_sla</td>
<td>This method is used to configure the IP SLA configuration on IOS-XE devices.</td>
</tr>
<tr>
<td>calc_sa_mac_addr</td>
<td>This method is used to calculate a dummy MAC address for IP-SLA Ethernet test for IOS-XE devices.</td>
</tr>
<tr>
<td>get_pe_ned_type</td>
<td>This method is used to create dictionary of NED types for the service end points.</td>
</tr>
<tr>
<td>get_pe_bgp_asn_no</td>
<td>This method is used to create dictionary of BGP AS Number and PE Loopback IP.</td>
</tr>
</tbody>
</table>
Function Name and Attributes | Description
---|---
get_rem_lpbk_ip(self, root, service, endpoints, pe_lpbk_ip) | This method is used to return the loopback address of remote service end point.

XML Templates

The package has been finally released with NSO 4.3.1 and might need to be recompiled for subsequent releases and XML template dependencies. As a reference, the following errors were reported on package migration from NSO 4.2.1 to NSO 4.3.1 and were appropriately fixed.

[l3vpn-policy-template.xml:53 Unknown element: 'te']
[l2vpn-vfi-template.xml:59 Unknown element: 'name']

YANG Model

The YANG model is structured to provide a logical modeling of the network topology such as User-facing Provider Edge (U-PE) and Network Provider Edge (N-PE), and of how the service layer is overlaid on the transport network. The service configuration is done for each service endpoint and associated N-PE.

Here is the representative workflow for the service creation where one chooses:

- Service type.
- Service signaling type, for example, LDP for VPWS.
- Service endpoint types and associated endpoint parameters such as U-PE and N-PE.
- U-PE association with N-PE in case of say H-VPLS.
- Unique service instance ID.

The service endpoints can be dynamically added or removed up to their minimum number, for example, VPWS service requires minimum of two endpoints.

Some tacit assumptions on the usage of service parameters exist. The support for service access with dot1Q is available, but the configuration option is not available for QinQ. No initial check or filtering is required to know the VLANs’ in use. The enhancements can be done if required.

Service Type and Signaling YANG Model Snippet

This section models the selection of service type and the associated signaling if any.

```yml
leaf service-type {
    description "L2VPN instance type";
    type enumeration {
        enum vpws;
        enum vpls;
        enum h-vpls;
        enum l3vpn;
        enum l3vpn-pwhe;
        enum pbb-evpn-vpws;
    }
}
```
The YANG model provides an abstraction of the service instance topology (list endpoint) overlay over the transport network.
For each supported service types namely VPWS, VPLS, H-VPLS, L3 VPN, and so on, one chooses the service end points such as U-PE's or Service Vertices (N-PE's) which finally extend the service to the U-PE.

For example, for VPWS, we will have only U-PE's.

For H-VPLS, we will have core nodes (N-PE's) to provide service and service edge nodes to terminate the H-VPLS service (U-PE's)

The endpoint specific parameters for all the service end points are input as part of the endpoint list.

Here is the list of few YANG input parameters. See the EPN package for the exhaustive list.

```yaml
list endpoint {
    must "((id and access-pe and inst-id)"
        error-message "Id, access-pe, if-type, if-num/generic-if-list and inst-id must be set"
    };
    key "id";
    min-elements 2;
    leaf id {
        tailf:info "Endpoint identifier";
        type uint8 {
            range "1..10"
        }
    }
    leaf access-pe {
        type leafref {
            path "/ncs:devices/ncs:device/ncs:name"
        }
        mandatory true;
    }
    leaf pe-role {
        type enumeration {
            enum n-pe;
            enum u-pe;
        }
        mandatory true;
    }
    leaf assoc-npe {
        when "(!/pe-role = 'u-pe')"
        type leafref {
            path "/ncs:devices/ncs:device/ncs:name"
        }
    }
    leaf pe-ned-type {
        type leafref {
            path "deref(!/access-pe)/../ncs:device-type/ncs:cli/ncs:ned-id"
        }
    }
    leaf if-type {
        type enumeration {
            enum Loopback;
            enum GigabitEthernet;
            enum TenGigabitEthernet;
            enum PW-Ether;
            enum BDI;
        }
    }
    leaf if-num {
        type string;
    }
    leaf inst-id {
        type int32;
    }
}```
range "1..4000";
} mandatory true;
}

leaf encap {
  type l2-serv-encap-type;
}

container rewrite {
  container ingress {
    leaf tag {
      type Rewrite;
    }
    container dot1q {
      when "./tag = 'translate1to1'";
      leaf vlan-id {
        type int32 {
          range "1..4000";
        }
      }
    }
  }
  leaf vlan-id {
    when "./encap = 'dot1q'";
    type int32 {
      range "1..4000";
    }
  }
  leaf eth-lpbk-type {
    type enumeration {
      enum internal;
      enum external;
    }
  }
}

container l2vpn-xc {
  tailf:cli-add-mode;
  when "((/./service-type = 'vpws') or (/../service-type = 'pbb-evpn-vpws') or (/../service-type = 'l3vpn-pwhe')"
  uses l2vpn-xc-parameters-grp;
}

container evpn-vpws-instance {
  when "(/../pe-role = 'u-pe') and (/../service-type = 'vpws')";
  description "evpn-vpws";
  tailf:cli-add-mode;
  container instance {
    tailf:cli-add-mode;
    leaf id {
      type uint16 {
        range "1..65535";
      }
    }
    container bgp-auto-discovery {
      config false;
      uses bgp-auto-discovery-parameters-grp;
    }
    container vpws-context {
      tailf:cli-add-mode;
      leaf name {
        type string;
        description "EVPN context name";
      }
      container service {
        tailf:cli-add-mode;
        leaf target-instance-id {
          type uint32;
          description "Target (remote) VPWS Service Instance identifier";
        }
      }
    }
  }
}

The VPLS YANG model is organized into grouping section and endpoint section. The snippets are shown below:

**Grouping section:**

```yini
grouping l2vpn-vfi-parameters-grp {
    description "L2VPN VFI parameters grouping";
    list context {
        key "name";
        leaf name {
            type string;
        }
        leaf vpn-id {
            type string;
        }
        leaf bgp-auto-discovery {
            config false;
            type enumeration {
                enum enable;
                enum disable;
            }
            default disable;
        }
    }
    list member {
        key "ipv4-addr";
        leaf ipv4-addr {
            type inet:ipv4-address;
        }
        leaf vc-id {
            type uint32 {
                range "1..2147483647";
            }
        }
        leaf encapsulation {
            type enumeration {
                enum mpls;
            }
        }
    }
}
```

**YANG Model Snippet for Virtual Forwarding Instance**

The VPLS YANG model is organized into grouping section and endpoint section. The snippets are shown below:

**Grouping section:**

```yini
leaf source-instance-id {
    type uint32;
    description "Source (local) VPWS Service Instance identifier";
}

container local-term {
    tailf:cli-add-mode;
    description "EVPN VPWS local interface";
    leaf interface {
        type string;
    }
    leaf instance {
        type int32 {
            range "1..4000";
        }
    }
}

container pbb-evpn-instance {
    when "{./pe-role = 'n-pe')";
    description "evpn";
    uses pbb-evpn-parameters-grp;
}
```
Endpoint section:

```yml
container l2vpn-vfi-instance {
  description "L2VPN VFI Instance";
  uses l2vpn-vfi-parameters-grp;
}
```

**YANG Model Snippet for Bridge Group or Domain**

The snippet for bridge domain is given below:

**Grouping section:**

```yml
grouping l2vpn-bridge-domain-parameters-grp {
  description "L2VPN Bridge Domain parameters grouping";
  container bridge-domain {
    tailf:cli-add-mode;
    leaf name {
      type string;
    }
    container member {
      tailf:cli-add-mode;
      container vfi {
        leaf vfi-name {
          type string;
        }
      }
      container local-term {
        tailf:cli-add-mode;
        leaf interface {
          type string;
        }
        leaf inst-id {
          type int32 {
            range "1..4000";
          }
        }
      }
      container neighbor {
        tailf:cli-add-mode;
        leaf nvrp-ip {
          type string;
        }
        leaf vc-id {
          type string;
        }
      }
    }
  }
}
```
Endpoint section:

```yaml
container bridge-group-instance {
    tailf:cli-add-mode;
    leaf group-name {
        type string;
    }
    description "L2VPN Bridge-Domain Instance";
    uses l2vpn-bridge-domain-parameters-grp;
}
```

YANG Model Snippet for Layer-2 Virtual Private Network XC

The snippet for L2 VPN XC is given below:

```yaml
Endpoint section:

```yaml
container l2vpn-xc {
    tailf:cli-add-mode;
    when "((../service-type = 'vpws') or (./service-type = 'pbb-evpn-vpws') or
    (./service-type = 'l3vpn-pwhe'))";
    leaf context {
        type string;
        config false;
    }
    leaf rem-pe-lpbk {
        type string;
    }
    leaf vc-id {
        type string;
        description "Virtual Circuit id";
    }
    container member {
        leaf-list xc-member {
            type string;
        }
    }
}
```

YANG Model Snippet for Ethernet Virtual Private Network

The snippet for EVPN YANG model is given below:

```yaml
container evpn-vpws-instance {
    when "((../pe-role = 'u-pe') and (./service-type = 'vpws'))";
    description "evpn-vpws";
    tailf:cli-add-mode;
    container instance {
        tailf:cli-add-mode;
        leaf id {
            type uint16 {
                range "1..65535";
            }
        }
        container bgp-auto-discovery {
            config false;
            uses bgp-auto-discovery-parameters-grp;
        }
        container vpws-context {
```
YANG Model Snippet for Provider Backbone Bridging Ethernet Virtual Private Network

The EVPN YANG model is organized into grouping section, common section and endpoint section. The snippets are given below:

**Grouping section:**

```yang
grouping pbb-evpn-parameters-grp {
  description "PBB EVPN parameters grouping";
  leaf evi {
    type uint16 {
      range "1..65535";
    }
    description "evi";
  }
  leaf edge-i-sid {
    type uint16;
    description "PBB EDGE I-SID";
    config false;
  }
}
```

**Endpoint section:**

```yang
container pbb-evpn-instance {
  when "(../pe-role = 'n-pe')";
  description "evpn";
  uses pbb-evpn-parameters-grp;
}
```

**Common section:**

```yang
container pbb-evpn-common {
  description "PBB EVPN Common Configuration";
  leaf source-bmac {
    type yang:hex-string;
    description "source-bmac";
  }
  leaf evpn-arp-proxy {
    type boolean;
  }
```
The snippet for pseudowire interface is given below:

```yang
container pseudowire-if {
  presence "Pseudowire Interface Configured";
  tailf:cli-add-mode;
  when "../../../vpws-sig";
  leaf pw-id {
    type string;
    description "pseudowire id";
  }
  leaf vc-id {
    type string;
    description "Virtual Circuit id";
  }
  leaf peer-ip {
    type inet:ip-address;
    description "peer IP address";
  }
  leaf pw-mtu {
    type int32 {
      range "1500..9216";
    }
  }
  leaf encapsulation {
    type enumeration {
      enum mpls;
      default mpls;
    }
  }
  leaf cw-negotiation {
    type enumeration {
      enum exclude;
      enum include;
      default include;
    }
  }
  leaf signaling-protocol {
    type enumeration {
      enum none;
      enum ldp;
      default ldp;
    }
  }
  container static-pw {
    when "../../../vpws-sig/no-sig";
    must "local-pseudowire-label" {
      error-message "local-pseudowire-label must be set when configuring vpws with no-signalling";
    }
    leaf local-pseudowire-label {
      type uint16 {
        range "100..5999";
      }
    }
  }
}
```
YANG Model Snippet for Y.1564 Service Activation

The snippet for service activation is given below:

```yang
container serv-act-y1564 {
    tailf:cli-add-mode;
    leaf ip-sla {
        type enumeration {
            enum enable {
                value "1";
            }
            enum disable {
                value "0";
            }
        }
        leaf tgen-pe {
            type leafref {
                path ".//endpoint/access-pe";
            }
        }
        leaf tgen-pe-vrf-id {
            when "../../../service-type = 'l3vpn' and ../tgen-pe";
            type int16;
        }
        leaf rem-tgen-pe {
            when "../../../service-type = 'l3vpn'";
            type leafref {
                path ".//endpoint/access-pe";
            }
            must "current() != ../tgen-pe" {
                error-message "Source and target must be different";
            }
        }
        leaf rem-tgen-pe-vrf-id {
            when "../../../service-type = 'l3vpn' and ../rem-tgen-pe";
            type int16;
        }
        leaf lpbk-pe {
            when "../../../service-type != 'l3vpn'";
            type leafref {
                path ".//endpoint/access-pe";
            }
            must "current() != ../tgen-pe" {
                error-message "Source and target must be different";
            }
        }
        leaf ip-sla-tgen {
            when "../ip-sla = 'enable'";
            type enumeration {
                enum enable {
                    value "1";
                }
                enum disable {
                    value "0";
                }
            }
        }
        leaf serv-perf-type {
            type enumeration {
                enum ip {
                    description "IP SLA for service type ip";
                }
            }
        }
    }
}
```
Python Mapping Logic

The Python mapping logic makes the following assumptions:

1. The transport network is configured and ready to support the overlay of services.
2. The node name is retrieved from the NSO configuration database (CDB).
3. Node ID is the BGP router-id (not user configurable).
4. All BGP address families such as VPNv4, VPNv6, VPLS, and EVPN have been properly configured as part of transport network setup.
5. BGP prefix list is implemented for the IOS-XE devices only:
   - The prefix-list name is BGP-Prefix-Filter.
   - New entry will be created, only if the node ID or loopback IP address of the new service endpoint is not in current list.
   - New entry will be augmented with a sequence number which is greater than the highest sequence number by ten.
   - Sample prefix list: Data ip prefix-list BGP-Prefix-Filter seq 90 permit 100.30.3.0/32.
6. Y.1564 service assurance is implemented only for IOS-XE devices with inherent capability:
   - IP SLA Type Ethernet is implemented for VPWS and VPLS services.
   - IP SLA Type IP is implemented for L3VPN services.
   - IP SLA tests are not applicable for H-VPLS / L2 ring termination scenarios.
7. PBB EVPN assumes that only IOS-XR devices are part of the PBB core. The PBB edge devices are IOS-XE devices that are connected to the core with VPWS.
8. To provide L3VPN-PWHE service, the CLI configuration for N-PE nodes require one to choose the if-type as PW-Ether. It is assumed that the PW-Ether interface is already provisioned as part of the transport configuration.
9. CE-PE routing is assumed as static route.
10. PW-ID is calculated by the mapping logic.
11. Pseudowire (PW) redundancy is not supported and will be added in subsequent releases.
12. BGP is running on all the nodes except L2 nodes. The Node ID is retrieved from the BGP configuration.

Implementation constraints:
1 Sometimes, the bridge-domain deletion on the IOS-XE devices give an error for specific devices. For the IOS-XE device issue, retry the service deletion.

2 The service activation test cannot be done for L2-ring based services.
Software Versions

This chapter contains the following sections:

- Network Service Orchestrator Software Version, page 23
- Network Element Software Version, page 23

Network Service Orchestrator Software Version

The table given below shows the minimum software (SW) version required to have a fully functional EPN5_0 package.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSO</td>
<td>4.3.1</td>
</tr>
<tr>
<td>CLI NED: cisco-ios</td>
<td>5.0.12</td>
</tr>
<tr>
<td>CLI NED: cisco-iosxr</td>
<td>5.0.8</td>
</tr>
</tbody>
</table>

Network Element Software Version

The table below shows the list of software components of the network element and their software version.

<table>
<thead>
<tr>
<th>Software Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS-XR (ASR9K)</td>
<td>6.1.3</td>
</tr>
<tr>
<td>IOS-XE (ASR920, ASR907, ASR903)</td>
<td>3.18.01.S</td>
</tr>
</tbody>
</table>
For the software maintenance updates (SMU) deployed in the EPN transport network, see the *Transport Design Guide*. 
Service Orchestration – Sample Configurations

This chapter contains the following sections:

- Virtual Private Wire Service with Label Distribution Protocol Signaling, page 25
- Virtual Private Wire Service with No-Signaling, page 27
- Ethernet Virtual Private Network Based Virtual Private Wire Service, page 29
- Virtual Private LAN Service with Label Distribution Protocol Signaling, page 31
- Virtual Private LAN Service with BGP Active Discovery Signaling, page 33
- Virtual Private LAN Service Using Label Distribution Protocol with Termination on L2 Ring, page 36
- Virtual Private LAN Service Using BGP – Active Discovery Signaling with Termination on L2 Ring, page 39
- Layer-3 Virtual Private Network, page 42
- Layer-3 Virtual Private Network with Termination on L2 Ring, page 46
- Pseudowire Headend Based Layer-3 Virtual Private LAN Service, page 48
- Hierarchical Virtual Private LAN Service with LDP Signaling, page 53
- Hierarchical Virtual Private LAN Service with BGP Active Discovery Signaling, page 57
- Provider Backbone Bridging Ethernet Virtual Private Network, page 61

Virtual Private Wire Service with Label Distribution Protocol Signaling

Here is a sample configuration for L2VPN based VPWS service with two endpoints. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

Sample NSO CLI configuration:

```
services epn5_0 vpws NCS1251
vpws-sig ldp-sig
```
endpoint 1 access-pe FAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1251
cap dot1q vlan-id 1251
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/2
inst-id 1251
cap dot1q vlan-id 1251

Configuration pushed by NSO:

native {
  device {
    name FAN-5303-ASR920
    data ip prefix-list BGP-Prefix-Filter seq 20 permit 100.56.7.0/32
    ethernet evc NCS1251
    !
    interface GigabitEthernet0/0/2
    service instance 1251 ethernet NCS1251
    encapsulation dot1q 1251
    rewrite ingress tag pop 1 symmetric
    exit
    exit
    l2vpn xconnect context NCS1251
    interworking ethernet
    member GigabitEthernet0/0/2 service-instance 1251
    member 100.56.7.0 12510 encapsulation mpls
  }
}

Service output:

show full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
vpws-sig ldp-sig
endpoint 1 FAN-5303-ASR920
pe-role u-pe
if-type GigabitEthernet
if-num 0/0/2
inst-id 1251
encap dot1q
vlan-id 1251
l2vpn-xc
context NCS1251
member
  pseudowire id 1251
  local-term
  interface GigabitEthernet0/0/2
  instance 1251
  !
  !
  !
ip-prefix-list
  prefix 100.56.7.0/32
  !
Virtual Private Wire Service with No-Signaling

Here is a sample configuration for L2VPN based VPWS service with two endpoints. This service can be terminated on IOS-XE access endpoints.

For pseudowire-if, the peer-ip is not user configurable. The mapping logic takes care of retrieving it from the topology.

Sample NSO CLI configuration:

```cli
services epn5_0 vpws NCS1251
vpws-sig no-sig
endpoint 1 access-pe FAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2 inst-id 1251
encap dot1q vlan-id 1251 pseudowire-if static-pw local-pseudowire-label 200
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/2 inst-id 1251
encap dot1q vlan-id 1251 pseudowire-if static-pw local-pseudowire-label 100
```

Configuration pushed by NSO:

```cli
native {
  device {
    name FAN-5303-ASR920
    data ip prefix-list BGP-Prefix-Filter seq 20 permit 100.56.7.0/32
    ethernet evc NCS1251
  }
  interface GigabitEthernet0/0/2
```
service instance 1251 ethernet NCS1251
  encapsulation dot1q 1251
  rewrite ingress tag pop 1 symmetric
exit
exit
interface pseudowire1251
  no shutdown
  encapsulation mpls
  signaling protocol none
  neighbor 100.56.7.0 1251
  control-word include
  label 200 100
  pseudowire type 5
exit
l2vpn xconnect context NCS1251
  interworking ethernet
  member GigabitEthernet0/0/2 service-instance 1251
  member pseudowire1251
}
)
device {
  name PAN-5607-ASR903
  data ip prefix-list BGP-Prefix-Filter seq 20 permit 100.53.3.0/32
  ethernet evc NCS1251
  !
  interface GigabitEthernet0/4/2
  service instance 1251 ethernet NCS1251
  encapsulation dot1q 1251
  rewrite ingress tag pop 1 symmetric
exit
exit
interface pseudowire1251
  no shutdown
  encapsulation mpls
  signaling protocol none
  neighbor 100.53.3.0 1251
  control-word include
  label 100 200
  pseudowire type 5
exit
l2vpn xconnect context NCS1251
  interworking ethernet
  member GigabitEthernet0/4/2 service-instance 1251
  member pseudowire1251
}
}

Service output:

how full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
  vpws-sig no-sig
endpoint 1 PAN-5303-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1251
  encaps dot1q
  vlan-id 1251
  l2vpn-xc
  context NCS1251
  member pseudowire id 1251
  local-term
  interface GigabitEthernet0/0/2
  instance 1251
  !
  !
ip-prefix-list
  prefix 100.56.7.0/32
Ethernet Virtual Private Network Based Virtual Private Wire Service

Here is a sample configuration for EVPN based VPWS with two endpoints. This service can be terminated on either IOS-XE pre-aggregation node or IOS-XE access node.

Sample configuration on NSO CLI:

```
services epn5_0 vpws NCS1251
vpps-sig evpn
endpoint 1 access-pe FAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1251
encap dot1q vlan-id 1251
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/2
inst-id 1251
encap dot1q vlan-id 1251
```

Note

The maximum transmission unit (MTU) is not configurable for EVPN based VPWS.

Configuration pushed by NSO:

```
native {
  device {
    name FAN-5303-ASR920
  }
```

```
```
data ethernet evc NCS1251
  !
  interface GigabitEthernet0/0/2
  service instance 1251 ethernet NCS1251
  encapsulation dot1q 1251
  rewrite ingress tag pop 1 symmetric
  exit

exit
l2vpn evpn instance 1251 point-to-point
vpws context NCS1251
  service target 1251 source 1251
  member GigabitEthernet0/0/2 service-instance 1251
  !
}

} device {
  name PAN-5607-ASR903
  data ethernet evc NCS1251
  !
  interface GigabitEthernet0/0/2
  service instance 1251 ethernet NCS1251
  encapsulation dot1q 1251
  rewrite ingress tag pop 1 symmetric
  exit

exit
l2vpn evpn instance 1251 point-to-point
vpws context NCS1251
  service target 1251 source 1251
  member GigabitEthernet0/0/2 service-instance 1251
  !
}

Service output:

show full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
  vpws-sig evpn
endpoint 1 PAN-5303-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1251
  encaps dot1q
  vlan-id 1251
  evpn-vpws-instance
    instance
      id 1251
      vpws-context
        name NCS1251
        service target-instance-id 1251
        service source-instance-id 1251
       !
    !
  ip-prefix-list
    prefix 100.56.7.0/32
    !
endpoint 2 PAN-5607-ASR903
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/4/2
  inst-id 1251
  encaps dot1q
  vlan-id 1251
  evpn-vpws-instance
    instance
      id 1259
      vpws-context
        name ABC

Virtual Private LAN Service with Label Distribution Protocol Signaling

Here is a sample configuration for LDB based VPLS service with two or more endpoints. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

The MTU is configurable for the VFI.

Sample NSO CLI Configuration:

```plaintext
services epn5_0 vpls NCS1251
vpls-sig ldp-sig
endpoint 1 access-pe FAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1251
encap dot1q vlan-id 1251
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/2
inst-id 1251
encap dot1q vlan-id 1251
endpoint 3 access-pe CSG-3103-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1251
encap dot1q vlan-id 1251

Configuration pushed by NSO:

native {
  device {
    name CSG-3103-ASR920
    data {
      ip prefix-list BGP-Prefix-Filter seq 90 permit 100.53.3.0/32
      ip prefix-list BGP-Prefix-Filter seq 100 permit 100.56.7.0/32
    }
    bridge-domain 1251
    l2vpn vfi context NCS1251
    vpn-id 1251
    member 100.53.3.0 10012510 encapsulation mpls
    member 100.56.7.0 10012510 encapsulation mpls
    bridge-domain 1251
    member vfi NCS1251
    ethernet evc NCS1251
    interface GigabitEthernet0/0/2
    service instance 1251 ethernet NCS1251
    encapsulation dot1q 1251
    rewrite ingress tag pop 1 symmetric
    exit
    exit
    bridge-domain 1251
    member GigabitEthernet0/0/2 service-instance 1251
  }
}
```

Note: Sample NSO CLI Configuration:
Service output:

show full-configuration services epn5_0 vpls NCS1251
services epn5_0 vpls NCS1251
vpls-sig ldp-sig
endpoint 1 FAN-5303-ASR920
pe-role u-pe
if-type GigabitEthernet
if-num 0/0/2
inst-id 1251
encap dot1q
vlan-id 1251
l2vpn-vfi-instance context NCS1251
vpn-id 1251
member 100.31.3.0
vc-id 12510


Virtual Private LAN Service with BGP Active Discovery Signaling

Here is a sample configuration for L2VPN based VPLS service with two or more endpoints. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

Sample NSO CLI Configuration:

```
services epn5_0 vpls NCS1211
vpls-sig bgp-auto-discovery enable
endpoint 1 access-pe AGG-3003-ASR9K pe-role u-pe if-type GigabitEthernet if-num 0/1/1/2
inst-id 1211
encap dot1q vlan-id 1211
endpoint 2 access-pe CSG-3102-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1211
encap dot1q vlan-id 1211
endpoint 3 access-pe CSG-1107-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1211
encap dot1q vlan-id 1211
Configuration pushed by NSO:

native {
```
device {
  name AGG-3003-ASR9K
  data interface GigabitEthernet 0/1/2.1211 12transport
     mtu 1522
     exit
  l2vpn
     pw-class NCS1211
     encapsulation mpls
     control-word
     exit
  exit
  bridge group NCS1211
  bridge-domain NCS1211
     interface GigabitEthernet0/1/2.1211
     exit
  vfi NCS1211
     vpn-id 1211
     autodiscovery bgp
     rd 100:1211
     route-target 100:1211
     route-target export 100:1211
     route-target import 100:1211
     signaling-protocol bgp
     ve-id 115
     ve-range 11
     exit
     exit
     exit
     exit
     exit
}

device {
  name CSG-1103-ASR920
  data ip prefix-list BGP-Prefix-Filter seq 90 permit 100.30.3.0/32
  bridge-domain 1211
  !
  l2vpn vfi context NCS1211
  vpn id 1211
  autodiscovery bgp signaling ldp
  !

  bridge-domain 1211
  member vfi NCS1211
  !
  ethernet evc NCS1211
  !
  interface GigabitEthernet0/0/2
    service instance 1211 ethernet NCS1211
    encapsulation dot1q 1211
    rewrite ingress tag pop 1 symmetric
    exit
    exit
    bridge-domain 1211
    member GigabitEthernet0/0/2 service-instance 1211
    !
}

device {
  name CSG-3102-ASR920
  data bridge-domain 1211
  !
  l2vpn vfi context NCS1211
  vpn id 1211
  autodiscovery bgp signaling ldp
  !

  bridge-domain 1211
  member vfi NCS1211
  !
  ethernet evc NCS1211
  !
  interface GigabitEthernet0/0/2
    service instance 1211 ethernet NCS1211
}
show full-configuration services epn5_0 vpls NCS1211
services epn5_0 vpls NCS1211
vpls-sig bgp-auto-discovery enable
endpoint 1 AGG-3003-ASR9K
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/1/1/2
  inst-id 1211
  encap dot1q
  vlan-id 1211
  l2vpn-vfi-instance context NCS1211
  vpn-id 1211
  !
  bridge-group-instance
  bridge-domain
    member
      vfi vfi-name NCS1211
      local-term
        interface GigabitEthernet0/1/1/2
        inst-id 1211
        !
        !
        !
        !
        !
endpoint 2 CSG-3102-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1211
  encap dot1q
  vlan-id 1211
  l2vpn-vfi-instance context NCS1211
  vpn-id 1211
  !
  bridge-group-instance
  bridge-domain
    member
      vfi vfi-name NCS1211
      local-term
        interface GigabitEthernet0/0/2
        inst-id 1211
        !
        !
        !
        !
        !
endpoint 3 CSG-1107-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1211
  encap dot1q
  vlan-id 1211
  l2vpn-vfi-instance context NCS1211
  vpn-id 1211
  !
  bridge-group-instance
  bridge-domain
    member
Virtual Private LAN Service Using Label Distribution Protocol with Termination on L2 Ring

Here is a sample configuration to extend the VPLS from the endpoint ‘x’ to the associated L2 ring (REP or G.8032). It is required to use the REP interface on the endpoint ‘x’ and to add an L2 leg using L2-ring-endpoint sub-command. The L2-ring-endpoint needs to be associated to the corresponding homing endpoint ‘x’.

This service can be terminated on IOS-XR pre-aggregation node, IOS-XE access node, or IOS-XE ring node.

Sample Configuration on NSO CLI:

```plaintext
services epn5_0 vpls NCS_UC8_VPLS_3640
vpls-sig 1dp-sig
endpoint 1 access-pe AGG-1003-ASR903 pe-role l2-ring-n-pe inst-id 3640
endpoint 2 access-pe AGG-4006-ASR903 pe-role l2-ring-n-pe inst-id 3640
l2-ring-endpoint CSG-1203-ASR920 assoc-l2-n-pe AGG-1003-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3640 inst-id 3640
  rewrite ingress tag translate1to1 dot1q vlan-id 3641
l2-ring-endpoint CSG-3202-ASR920 assoc-l2-n-pe AGG-4006-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3640 inst-id 3640
  rewrite ingress tag translate1to1 dot1q vlan-id 3641
endpoint 3 access-pe CSG-1106-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
  inst-id 3640
  encap dot1q vlan-id 3640
```

The above configuration is using VLAN-ID 3640 as the customer VLAN (C-VLAN) and VLAN-ID 3641 as service-provider VLAN (S-VLAN). The VLAN translation is done on the L2 ring endpoint with CLI:

```plaintext
rewrite ingress tag translate 1-to-1 dot1q 3641 symmetric
```

Configuration pushed by NSO:

```plaintext
native {
  device {
    name AGG-1003-ASR903
    data bridge-domain 3640
    !
    l2vpn vfi context NCS_UC8_VPLS_3640
    vpn id 3640
    member 100.11.6.0 10036400 encapsulation mpls
    member 100.40.6.0 10036400 encapsulation mpls
    !
    bridge-domain 3640
    member vfi NCS_UC8_VPLS_3640
    !
  }
}
```
device {  
  name AGG-4006-ASR903  
data bridge-domain 3640  
  !  
l2vpn vfi context NCS_UC8_VPLS_3640  
vpn id 3640  
  member 100.10.3.0 10036400 encapsulation mpls  
  member 100.11.6.0 10036400 encapsulation mpls  
  !  
bridge-domain 3640  
  member vfi NCS_UC8_VPLS_3640  
  !  
}  

device {  
  name CSG-1106-ASR920  
data bridge-domain 3640  
  !  
l2vpn vfi context NCS_UC8_VPLS_3640  
vpn id 3640  
  member 100.10.3.0 10036400 encapsulation mpls  
  member 100.40.6.0 10036400 encapsulation mpls  
  !  
bridge-domain 3640  
  member vfi NCS_UC8_VPLS_3640  
  !  
eternet evc NCS_UC8_VPLS_3640  
  !  
interface GigabitEthernet0/0/2  
  service instance 3640 ethernet NCS_UC8_VPLS_3640  
  encapsulation dot1q 3640  
  rewrite ingress tag pop 1 symmetric  
  exit  
  exit  
  bridge-domain 3640  
  member GigabitEthernet0/0/2 service-instance 3640  
  !  
}  

device {  
  name CSG-1203-ASR920  
data bridge-domain 3640  
  !  
eternet evc NCS_UC8_VPLS_3640  
  !  
interface GigabitEthernet0/0/2  
  service instance 3640 ethernet NCS_UC8_VPLS_3640  
  encapsulation dot1q 3640  
  rewrite ingress tag translate 1-to-1 dot1q 3641 symmetric  
  exit  
  exit  
  bridge-domain 3640  
  member GigabitEthernet0/0/2 service-instance 3640  
  !  
}  

device {  
  name CSG-3202-ASR920  
data bridge-domain 3640  
  !  
eternet evc NCS_UC8_VPLS_3640  
  !  
interface GigabitEthernet0/0/2  
  service instance 3640 ethernet NCS_UC8_VPLS_3640  
  encapsulation dot1q 3640  
  rewrite ingress tag translate 1-to-1 dot1q 3641 symmetric  
  exit  
  exit  
  bridge-domain 3640  
  member GigabitEthernet0/0/2 service-instance 3640  
  !  
}  
}
Service output:

```
show full-configuration services epn5_0 vpls NCS_UC8_VPLS_3640
services epn5_0 vpls NCS_UC8_VPLS_3640
vpls-sig ldp-sig
l2-ring-endpoint CSG-1203-ASR920
assoc-l2-n-pe AGG-1003-ASR903
if-type GigabitEthernet
if-num 0/0/2
inst-id 3640
encap dot1q
rewrite ingress tag translate1to1
rewrite ingress dot1q vlan-id 3641
vlan-id 3640
bridge-group-instance
bridge-domain
member
  local-term
    interface GigabitEthernet0/0/2
    inst-id 3640

l2-ring-endpoint CSG-3202-ASR920
assoc-l2-n-pe AGG-4006-ASR903
if-type GigabitEthernet
if-num 0/0/2
inst-id 3640
encap dot1q
rewrite ingress tag translate1to1
rewrite ingress dot1q vlan-id 3641
vlan-id 3640
bridge-group-instance
bridge-domain
member
  local-term
    interface GigabitEthernet0/0/2
    inst-id 3640

endpoint 1 AGG-1003-ASR903
pe-role l2-ring-n-pe
inst-id 3640
l2vpn-vfi-instance context NCS_UC8_VPLS_3640
vpn-id 3640
member 100.11.6.0
  vc-id 36400
member 100.40.6.0
  vc-id 36400

bridge-group-instance
bridge-domain
member
  vfi vfi-name NCS_UC8_VPLS_3640

endpoint 2 AGG-4006-ASR903
pe-role l2-ring-n-pe
inst-id 3640
l2vpn-vfi-instance context NCS_UC8_VPLS_3640
vpn-id 3640
member 100.10.3.0
  vc-id 36400
```
Here is a sample configuration to extend the VPLS service from the endpoint 'x' to the associated L2 ring (REP or G.8032). It is required to use the REP interface on the endpoint 'x' and to add an L2 leg using l2-ring-endpoint sub-command. The L2-ring endpoint needs to be associated to corresponding homing endpoint 'x'. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

**Sample NSO CLI Configuration:**

```plaintext
services epn5_0 vpls NCS_UC8_VPLS_3740
vpls-sig bgp-auto-discovery enable
endpoint 1 access-pe AGG-1003-ASR903 pe-role l2-ring-n-pe inst-id 1000
endpoint 2 access-pe AGG-4006-ASR903 pe-role l2-ring-n-pe inst-id 1000
l2-ring-endpoint CSG-1203-ASR920 assoc-l2-n-pe AGG-1003-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3740 inst-id 1000
  rewrite ingress tag translate1to1 dot1q vlan-id 3741
l2-ring-endpoint CSG-3202-ASR920 assoc-l2-n-pe AGG-4006-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3740 inst-id 1000
```

---

Virtual Private LAN Service Using BGP – Active Discovery Signaling with Termination on L2 Ring

Here is a sample configuration to extend the VPLS service from the endpoint 'x' to the associated L2 ring (REP or G.8032). It is required to use the REP interface on the endpoint 'x' and to add an L2 leg using l2-ring-endpoint sub-command. The L2-ring endpoint needs to be associated to corresponding homing endpoint 'x'. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

**Sample NSO CLI Configuration:**

```plaintext
services epn5_0 vpls NCS_UC8_VPLS_3740
vpls-sig bgp-auto-discovery enable
endpoint 1 access-pe AGG-1003-ASR903 pe-role l2-ring-n-pe inst-id 1000
endpoint 2 access-pe AGG-4006-ASR903 pe-role l2-ring-n-pe inst-id 1000
l2-ring-endpoint CSG-1203-ASR920 assoc-l2-n-pe AGG-1003-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3740 inst-id 1000
  rewrite ingress tag translate1to1 dot1q vlan-id 3741
l2-ring-endpoint CSG-3202-ASR920 assoc-l2-n-pe AGG-4006-ASR903 if-type GigabitEthernet
  if-num 0/0/2
  encap dot1q vlan-id 3740 inst-id 1000
```
Configuration pushed by NSO:

```
native {
  device {
    name AGG-1003-ASR903
    data bridge-domain 1000
      !
      l2vpn vfi context NCS_UC8_VPLS_3740
      vpn id 1000
      autodiscovery bgp signaling ldp
      
      !
      bridge-domain 1000
      member vfi NCS_UC8_VPLS_3740
      
  }
  device {
    name AGG-4006-ASR903
    data bridge-domain 1000
      !
      l2vpn vfi context NCS_UC8_VPLS_3740
      vpn id 1000
      autodiscovery bgp signaling ldp
      
      !
      bridge-domain 1000
      member vfi NCS_UC8_VPLS_3740
      
  }
  device {
    name CSG-1106-ASR920
    data bridge-domain 3640
      !
      l2vpn vfi context NCS_UC8_VPLS_3740
      vpn id 3640
      autodiscovery bgp signaling ldp
      
      !
      bridge-domain 3640
      member vfi NCS_UC8_VPLS_3740
      
      ethernet evc NCS_UC8_VPLS_3740
      
      interface GigabitEthernet0/0/2
      service instance 3640 ethernet NCS_UC8_VPLS_3740
      encapsulation dot1q 3640
      rewrite ingress tag pop 1 symmetric
      exit
      exit
      bridge-domain 3640
      member GigabitEthernet0/0/2 service-instance 3640
      
  }
  device {
    name CSG-1203-ASR920
    data bridge-domain 3740
      !
      ethernet evc NCS_UC8_VPLS_3740
      
      interface GigabitEthernet0/0/2
      service instance 1000 ethernet NCS_UC8_VPLS_3740
      encapsulation dot1q 3740
      rewrite ingress tag translate 1-to-1 dot1q 3741 symmetric
      exit
      exit
      bridge-domain 3740
      member GigabitEthernet0/0/2 service-instance 1000
      
  }
}
```
Service output:

show full-configuration services epn5 0 vpls NCS_UC8_VLAN_3740
services epn5_0 vpls NCS_UC8_VLAN_3740
vpls-sig bgp-auto-discovery enable
l2-ring-endpoint CSG-1203-ASR920
assoc-l2-n-pe AGG-1003-ASR903
if-type GigabitEthernet
if-num 0/0/2
inst-id 1000
encap dot1q
rewrite ingress tag translate 1-to-1 dot1q 3741 symmetric
exit

bridge-domain
member
local-term
interface GigabitEthernet0/0/2
inst-id 1000

l2-ring-endpoint CSG-3202-ASR920
assoc-l2-n-pe AGG-4006-ASR903
if-type GigabitEthernet
if-num 0/0/2
inst-id 1000
encap dot1q
rewrite ingress tag translate 1-to-1 dot1q 3741 symmetric
exit

bridge-domain
member
local-term
interface GigabitEthernet0/0/2
inst-id 1000

endpoint 1 AGG-1003-ASR903
pe-role l2-ring-n-pe
inst-id 1000
l2vpn-vf1-instance context NCS_UC8_VLAN_3740
vpn-id 1000

bridge-group-instance
bridge-domain

Layer-3 Virtual Private Network

Here is a sample configuration for L3VPN service with two or more endpoints. This service can be terminated on either IOS-XR pre-aggregation node or IOS-XE access node.

Ensure that the pe-ip-addr is on the same subnet as that of the corresponding customer edge (CE).

Sample NSO CLI configuration:

```
services epn5_0 l3vpn NCS140
endpoint 1 access-pe PAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/0
  inst-id 140
  pe-ip-addr 192.168.4.1 pe-mask 255.255.255.0
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/0
  inst-id 140
  pe-ip-addr 192.168.9.1 pe-mask 255.255.255.0
```
Optionally, you can configure the VRF name using the CLI: `vrf forwarding <vrf-name>`.

Configuration pushed by NSO:

```plaintext
native {
  device {
    name FAN-5303-ASR920
    data vrf definition NCS140
    rd 200:140
    address-family ipv4
    route-target export 200:140
    route-target import 200:140
    exit-address-family
    !
    ip prefix-list BGP-Prefix-Filter seq 20 permit 100.56.7.0/32
    interface GigabitEthernet0/0/0
description Link for vrf NCS140
    vrf forwarding NCS140
    ip address 192.168.4.1 255.255.255.0
    exit
    router bgp 200
    address-family ipv4 unicast vrf NCS140
    redistribute connected
    exit-address-family
    !
  }!
}
device {
  name FAN-5607-ASR903
  data vrf definition NCS140
  rd 200:140
  address-family ipv4
  route-target export 200:140
  route-target import 200:140
  exit-address-family
  !
  ip prefix-list BGP-Prefix-Filter seq 30 permit 100.53.3.0/32
  interface GigabitEthernet0/4/0
description Link for vrf NCS140
  vrf forwarding NCS140
  ip address 192.168.9.1 255.255.255.0
  exit
  router bgp 200
  address-family ipv4 unicast vrf NCS140
  redistribute connected
  exit-address-family
  !
}
}

Service output:

```
show full-configuration services epn5_0 l3vpn NCS140
services epn5_0 l3vpn NCS140
  endpoint 1 FAN-5303-ASR920
    pe-role u-pe
    if-type GigabitEthernet
    if-num 0/0/0
    pe-ip-addr 192.168.4.1
    pe-mask 255.255.255.0
    vrf forwarding NCS140
    inst-id 140
    ip-prefix-list
    prefix 100.56.7.0/32
```
Enabling SR-ODN and imposing policy (on IOS-XR terminated endpoints only):

Setup L3VPN service:

```plaintext
services epn5_0 13vpn NCS740
endpoint 1 AGG-3003-ASR9K pe-role u-pe if-type TenGigabitEthernet if-num 0/0/0/7 inst-id 740
encap dot1q vlan-id 740
pe-ip-addr 192.168.4.1 pe-mask 255.255.255.0
endpoint 2 AGG-0908-ASR9K pe-role u-pe if-type TenGigabitEthernet if-num 0/0/0/0 inst-id 740
encap dot1q vlan-id 740
pe-ip-addr 192.168.9.1 pe-mask 255.255.255.0
```

Configuration pushed by NSO:

```plaintext
native {
  device {
    name AGG-0908-ASR9K
    data vrf NCS740
    address-family ipv4 unicast
    import route-target
    100:740
    exit
    export route-target
    100:740
    exit
    exit
  }
  device {
    name AGG-3003-ASR9K
    data vrf NCS740
    address-family ipv4 unicast
    import route-target
    100:740
    exit
    export route-target
    100:740
    exit
    exit
  }
}
```
exit
interface TenGigE 0/0/0/7.740
description Link to AGG-3003-ASR9K
encapsulation dot1q 740
vrf NCS740
ipv4 address 192.168.4.1 255.255.255.0
exit
router bgp 100
vrf NCS740
rd 100:740
address-family ipv4 unicast
redistribute connected
exit
exit
}

A sample policy has been hard-coded for demonstration purpose, on the assumption that a path computation element (PCE) is present and accessible in the network.

**Sample NSO CLI configuration:**

This configuration step is subsequent to the L3VPN service setup.

```
l3vpn-sr-odn loc-policy-pe AGG-0908-ASR9K rem-policy-pe AGG-3003-ASR9K sr-odn enable community 999:999
```

**Configuration pushed by NSO:**

native {
    device {
        name AGG-0908-ASR9K
        data prefix-set L3VPN_PREFIXES_740
        192.168.9.0/24
        end-set
        route-policy SET_Community_740
        if destination in L3VPN_PREFIXES_740 then
            set community (999:999) additive
        else
            pass
        endif
        end-policy
        router bgp 100
        vrf NCS740
        address-family ipv4 unicast
        redistribute connected route-policy SET_Community_740
        exit
        exit
    }
}

device {
    name AGG-3003-ASR9K
    data prefix-set L3VPN_PREFIXES_740
    192.168.4.0/24
    end-set
    route-policy MPLS_TE_ATTR_740
    if community matches-any (999:999) then
        set mpls traffic-eng attributeset Set_MPLS_TE_ATTR_740
    else
        pass
    endif
    end-policy
    router bgp 100
    neighbor-group SvRR
```
Layer-3 Virtual Private Network with Termination on L2 Ring

Here is a sample configuration for L3VPN service with termination on L2 ring. This service can have two or more endpoints. This service can be terminated on IOS-XE L2 access ring.

Ensure that the pe-ip-addr is in same subnet as the corresponding CE. For termination on the ring, you need to choose a ring origination endpoint.

Sample configuration on NSO CLI:

```
services epn5_0 l3vpn NCS_UC8_L3VPN_3640
endpoint 1 access-pe AGG-1003-ASR903 pe-role 12-ring-n-pe inst-id 3640 if-type BDI
pe-ip-addr 192.168.4.1 pe-mask 255.255.255.0
endpoint 2 access-pe AGG-4006-ASR903 pe-role 12-ring-n-pe inst-id 3640 if-type BDI
pe-ip-addr 192.168.9.1 pe-mask 255.255.255.0
12-ring-endpoint CSG-1203-ASR920 assoc-12-n-pe AGG-1003-ASR903 if-type GigabitEthernet
if-num 0/0/2
encap dot1q vlan-id 3640 inst-id 3640
12-ring-endpoint CSG-3202-ASR920 assoc-12-n-pe AGG-4006-ASR903 if-type GigabitEthernet
if-num 0/0/2
encap dot1q vlan-id 3640 inst-id 3640
```

Configuration pushed by NSO:

```
native {
  device {
    name AGG-1003-ASR903
    data vrf definition NCS_UC8_L3VPN_3640
    rd 100:3640
    address-family ipv4
    route-target export 100:3640
    route-target import 100:3640
    exit-address-family
  }
  !
  ip prefix-list BGP-Prefix-Filter seq 12 permit 100.40.6.0/32
  interface BDI3640
  vrf forwarding NCS_UC8_L3VPN_3640
  ip address 192.168.4.1 255.255.255.0
  no shutdown
```
Service output:

show full-configuration services epn5_0 l3vpn NCS_UC8_L3VPN_3640
services epn5_0 l3vpn NCS_UC8_L3VPN_3640
  12-ring-endPoint CSG-1203-ASR920
  assoc-12-n-pe AGG-1003-ASR903
  if-type GigabitEthernet
Pseudowire Headend Based Layer-3 Virtual Private LAN Service

Here is a sample of the PWHE based L3VPN service configuration. It is required to ensure that the pe-ip-addr is on the same subnet as that of the corresponding CE. This service can be terminated on the IOS-XE access node. The PWHE nodes are IOS-XR nodes.

Note

The PE-IP address is configured on the U-PE, but it is used on the PW-Ether interface on the associated N-PE.
Sample NSO CLI configuration:

```
services epn5_0 13vpn-pwhe NCS601
endpoint 1 access-pe CSG-3102-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
  inst-id 601
  encap dot1q vlan-id 601
  pe-ip-addr 192.168.4.1 pe-mask 255.255.255.0
  assoc-npe AGG-3004-ASR9K
endpoint 2 access-pe AGG-3004-ASR9K pe-role n-pe if-type PW-Ether generic-if-list Ring31-IL
  inst-id 601
  encap dot1q vlan-id 601
  13vpn-pe-prefix 192.168.4.0/24
endpoint 3 access-pe AGG-0909-ASR9K pe-role n-pe if-type PW-Ether generic-if-list Ring10-IL
  inst-id 601
  encap dot1q vlan-id 601
  13vpn-pe-prefix 192.168.9.0/24
endpoint 4 access-pe CSG-11062-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
  inst-id 601
  encap dot1q vlan-id 601
  pe-ip-addr 192.168.9.1 pe-mask 255.255.255.0
  assoc-npe AGG-0909-ASR9K
```

Configuration pushed by NSO:

```
native {
  device {
    name AGG-0909-ASR9K
    data vrf NCS601
    address-family ipv4 unicast
      import route-target 100:601
      exit
      export route-target 100:601
      exit
      exit
      exit
  }
  interface PW-Ether 6014
  vrf NCS601
  ipv4 address 192.168.9.1 255.255.255.0
  attach generic-interface-list Ring10-IL
  exit
  l2vpn
    xconnect group PWHE
    p2p NCS601-4
      interface PW-Ether6014
      neighbor ipv4 100.11.6.0 pw-id 6014
      pw-class PWHE
      exit
      exit
  }
  device {
    name AGG-3004-ASR9K
    data vrf NCS601
    address-family ipv4 unicast
      import route-target 100:601
      exit
      export route-target 100:601
      exit
      exit
  }
```
Service Orchestration – Sample Configurations


Service output:

show full-configuration services epn5_0 l3vpn-pwhe NCS601
services epn5_0 l3vpn-pwhe NCS601
l3vpn-pwhe sr-odn sr-odn enable
policy-pe AGG-3004-ASR9K
endpoint 1 CSG-3102-ASR920
pe-role u-pe
assoc-npe AGG-3004-ASR9K
if-type GigabitEthernet

Pseudowire Headend Based Layer-3 Virtual Private LAN Service
if-num 0/0/2
pe-ip-addr 192.168.4.1
pe-mask 255.255.255.0
inst-id 601
encap dot1q
vlan-id 601
l2vpn-xc
context NCS601
member
  local-term
    interface GigabitEthernet0/0/2
    instance 601
  neighbor
    nbr-ip 100.30.4.0
    vc-id 6011

endpoint 2 AGG-3004-ASR9K
pe-role n-pe
if-type PW-Ether
l3vpn-pe-prefix [ 192.168.4.0/24 ]
vrf forwarding NCS601
generic-if-list Ring31-IL
inst-id 601
encap dot1q
vlan-id 601
l2vpn-xc
context NCS601
member
  local-term
    interface PW-Ether601
    instance 601
  neighbor
    nbr-ip 100.31.2.0
    vc-id 6011

endpoint 3 AGG-0909-ASR9K
pe-role n-pe
if-type PW-Ether
l3vpn-pe-prefix [ 192.168.9.0/24 ]
vrf forwarding NCS601
generic-if-list Ring10-IL
inst-id 601
encap dot1q
vlan-id 601
l2vpn-xc
context NCS601
member
  local-term
    interface PW-Ether601
    instance 601
  neighbor
    nbr-ip 100.11.6.0
    vc-id 6014

endpoint 4 CSG-1106-ASR920
pe-role u-pe
assoc-npe AGG-0909-ASR9K
if-type GigabitEthernet
if-num 0/0/2
pe-ip-addr 192.168.9.1
pe-mask 255.255.255.0
Enabling SR-ODN and imposing policy

The policy has been hard-coded for demonstration purpose, on the assumption that the PCE is present and accessible in the network.

Sample NSO CLI configuration:

This configuration step is subsequent to the L3VPN-PWHE service setup.

l3vpn-pwhe-sr-odn policy-pe AGG-3004-ASR9K community 999:999 sr-odn enable

Configuration pushed by NSO:

```
native {
  device {
    name AGG-0909-ASR9K
    data prefix-set L3VPN_PREFIXES_601
      192.168.9.0/24
      end-set
    route-policy MPLS_TE_ATTR_601
      if community matches-any (999:999) then
        set mpls traffic-eng attributeset Set_MPLS_TE_ATTR_601
        pass
      else
        pass
      endif
      end-policy
    router bgp 100
    neighbor-group SvRR
    address-family vpnv4 unicast
      route-policy MPLS_TE_ATTR_601 in
      exit
    exit
    mpls traffic-eng
    attribute-set p2p-te Set_MPLS_TE_ATTR_601
    pce
    exit
    path-selection
    metric te
    exit
    exit
  }
  device {
    name AGG-3004-ASR9K
    data route-policy SET_Community_601
      if destination in L3VPN_PREFIXES_601 then
        set community (999:999) additive
      end-if
      exit
  }
}
```
Hierarchical Virtual Private LAN Service with LDP Signaling

Here is a sample configuration for the creation of a H-VPLS with core nodes or N-PE nodes such as AGG-0909-ASR9K, AGG-3004-ASR9K, and AGG-3003-ASR9K. The associated H-VPLS service edge points are:

- u-pe node: CSG-1102-ASR920 associated with AGG-0909-ASR9K.
- u-pe node: CSG-3103-ASR920 associated with AGG-3003-ASR9K.

This service can be terminated on the IOS-XE access node. The H-VPLS core consists of IOS-XR devices.

Sample NSO CLI configuration:

```plaintext
services epn5_0 h-vpls NCS1440
vpls-sig ldp-sig
endpoint 1 access-pe CSG-1106-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2 inst-id 1440
   encap dot1q vlan-id 1440
   assoc-npe AGG-0909-ASR9K
endpoint 2 access-pe AGG-0909-ASR9K pe-role n-pe inst-id 1440
endpoint 3 access-pe AGG-3003-ASR9K pe-role n-pe inst-id 1440
endpoint 4 access-pe CSG-3103-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2 inst-id 1440
   encap dot1q vlan-id 1440
   assoc-npe AGG-3003-ASR9K
endpoint 5 access-pe AGG-3004-ASR9K pe-role n-pe if-type GigabitEthernet inst-id 1440

Configuration pushed by NSO:

```plaintext
native {
  device {
    name AGG-0909-ASR9K
    data l2vpn
    pw-class NCS1440
    encapsulation mpls
    control-word
    exit
  }
  bridge group NCS1440
  bridge-domain NCS1440
  neighbor 100.11.6.0 pw-id 144012
  exit
  vfi NCS1440
  vpn-id 1440
  neighbor 100.30.3.0 pw-id 144023
  pw-class NCS1440
  exit
  neighbor 100.30.4.0 pw-id 144025
}
pw-class NCS1440
exit
exit
exit
exit
exit
}
device {
  name AGG-3003-ASR9K
  data l2vpn
  pw-class NCS1440
  encapsulation mpls
  control-word
  exit
  exit
  bridge group NCS1440
  bridge-domain NCS1440
  neighbor 100.31.3.0 pw-id 144034
  exit
  vfi NCS1440
  vpn-id 1440
  neighbor 100.30.4.0 pw-id 144035
  pw-class NCS1440
  exit
  neighbor 100.9.9.0 pw-id 144023
  pw-class NCS1440
  exit
  exit
  exit
  exit
}
device {
  name AGG-3004-ASR9K
  data l2vpn
  pw-class NCS1440
  encapsulation mpls
  control-word
  exit
  exit
  bridge group NCS1440
  bridge-domain NCS1440
  vfi NCS1440
  vpn-id 1440
  neighbor 100.30.4.0 pw-id 144035
  pw-class NCS1440
  exit
  neighbor 100.9.9.0 pw-id 144025
  pw-class NCS1440
  exit
  exit
  exit
  exit
}
device {
  name CSG-1106-ASR920
  data bridge-domain 1440
    !
    l2vpn vfi context NCS1440
    vpn id 1440
    member 100.9.9.0 144012 encapsulation mpls
    mtu 1508
    !
    bridge-domain 1440
    member vfi NCS1440
    !
    ethernet evc NCS1440
    !
    interface GigabitEthernet0/0/2
    service instance 1440 ethernet NCS1440
    encapsulation dot1q 1440
    rewrite ingress tag pop 1 symmetric
Service output:

show full-configuration services epn5_0 h-vpls NCS1440
services epn5_0 h-vpls NCS1440
vpls-sig ldp-sig
derived 1 CSG-1106-ASR920
pe-role n-pe
assoc-npe AGG-0909-ASR9K
if-type GigabitEthernet
if-num 0/0/2
inst-id 1440
cap cap dot1q
vni 1440
v2v-fni-instance context NCS1440
vpn-id 1440
member 100.9.9.0
vc-id 1440

bridge-group-instance
bridge-domain
member
vni vni-name NCS1440
local-term
interface GigabitEthernet0/0/2
inst-id 1440

ip-pref-list
prefix 100.9.9.0/32

endpoint 2 AGG-0909-ASR9K
pe-role d-pe
inst-id 1440
v2v-fni-instance context NCS1440
vpn-id 1440
member 100.30.3.0
vc-id 144023
! member 100.30.4.0
vc-id 144025
!

bridge-group-instance
bridge-domain
member
vfi vfi-name NCS1440
neighbor
nbr-ip 100.11.6.0
vc-id 144012
!
!
!
endpoint 3 AGG-3003-ASR9K
pe-role n-pe
inst-id 1440
l2vpn-vfi-instance context NCS1440
vpn-id 1440
member 100.9.9.0
vc-id 144023
!
member 100.30.4.0
vc-id 144035
!

bridge-group-instance
bridge-domain
member
vfi vfi-name NCS1440
neighbor
nbr-ip 100.31.3.0
vc-id 144034
!
!
!
!
endpoint 4 CSG-3103-ASR920
pe-role u-pe
assoc-npe AGG-3003-ASR9K
if-type GigabitEthernet
if-num 0/0/2
inst-id 1440
encap dot1q
vlan-id 1440
l2vpn-vfi-instance context NCS1440
vpn-id 1440
member 100.30.3.0
vc-id 144034
!
!
!
!
bridge-group-instance
bridge-domain
member
vfi vfi-name NCS1440
local-term
interface GigabitEthernet0/0/2
inst-id 1440
!
!
!
ip-prefix-list
prefix 100.30.3.0/32
!
!
endpoint 5 AGG-3004-ASR9K
pe-role n-pe
Hierarchical Virtual Private LAN Service with BGP Active Discovery Signaling

Here is the sample configuration for the creation of the H-VPLS service with core nodes or N-PE nodes such as AGG-0909-ASR9K, AGG-3004-ASR9K, and AGG-3003-ASR9K. The associated H-VPLS service edge points are:

- u-pe node: CSG-1102-ASR920 associated with AGG-0909-ASR9K.
- u-pe node: CSG-3103-ASR920 associated with AGG-3003-ASR9K.

This service can be terminated on IOS-XE access node. The H-VPLS core consists of IOS-XR devices.

Sample NSO CLI Configuration:

```
services epn5_0 h-vpls NCS1340
vpls-sig bgp-auto-discovery enable
endpoint 1 access-pe CSG-1106-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1340
encap dot1q vlan-id 1340
assoc-npe AGG-0909-ASR9K
endpoint 2 access-pe AGG-0909-ASR9K pe-role n-pe inst-id 1340
endpoint 3 access-pe AGG-3003-ASR9K pe-role n-pe inst-id 1340
endpoint 4 access-pe CSG-3103-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
inst-id 1340
encap dot1q vlan-id 1340
assoc-npe AGG-3003-ASR9K
endpoint 5 access-pe AGG-3004-ASR9K pe-role n-pe inst-id 1340
```

Configuration pushed by NSO:
neighbor 100.11.6.0 pw-id 100134012
exit
vfi NCS1340
vpn-id 1340
autodiscovery bgp
rd 100:1340
route-target 100:1340
route-target import 100:1340
route-target export 100:1340
signaling-protocol bgp
ve-id 115
ve-range 11
exit
exit
exit
exit
exit

} device {
   name AGG-3003-ASR9K
   data l2vpn
   pw-class NCS1340
   encapsulation mpls
   control-word
   exit
exit
bridge group NCS1340
bridge-domain NCS1340
neighbor 100.31.3.0 pw-id 100134034
exit
vfi NCS1340
vpn-id 1340
autodiscovery bgp
rd 100:1340
route-target 100:1340
route-target import 100:1340
route-target export 100:1340
signaling-protocol bgp
ve-id 115
ve-range 11
exit
exit
exit
exit
exit

} device {
   name CSG-1106-ASR920
   data bridge-domain 1340
   !
   l2vpn vfi context NCS1340
   vpn id 1340
   autodiscovery bgp signaling ldp
   !
   mtu 1508
   !
   bridge-domain 1340
   member vfi NCS1340
   !
   ethernet evc NCS1340
   !
   interface GigabitEthernet0/0/2
   service instance 1340 ethernet NCS1340
   encapsulation dot1q 1340
   rewrite ingress tag pop 1 symmetric
   exit
exit
bridge-domain 1340
   member GigabitEthernet0/0/2 service-instance 1340
   !
}
Service output:

show full-configuration services epn5_0 h-vpls NCS1340
services epn5_0 h-vpls NCS1340
vpls-sig bgp-auto-discovery enable
endpoint 1 CSG-1106-ASR920
  pe-role u-pe
  assoc-npe AGG-0909-ASR9K
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1340
  encap dot1q
  vlan-id 1340
  l2vpn-vfi-instance context NCS1340
  vpn-id 1340
  bridge-group-instance
    bridge-domain
      member
        vfi vfi-name NCS1340
        local-term
        interface GigabitEthernet0/0/2
        inst-id 1340
        !
        !
        !
        !
        ip-prefix-list
        prefix 100.9.9.0/32
        !
endpoint 2 AGG-0909-ASR9K
  pe-role n-pe
  inst-id 1340
  l2vpn-vfi-instance context NCS1340
  vpn-id 1340
  bridge-group-instance
    bridge-domain
      member
        vfi vfi-name NCS1340
        neighbor
        nbr-ip 100.11.0
        vc-id 134012
  !
Hierarchical Virtual Private LAN Service with BGP Active Discovery Signaling

endpoint 3 AGG-3003-ASR9K
pe-role n-pe
inst-id 1340
l2vpn-vfi-instance context NCS1340
  vpn-id 1340
  bridge-group-instance
  bridge-domain
    member
      vfi vfi-name NCS1340
      neighbor
        nbr-ip 100.31.3.0
        vc-id 134034
  }
}
endpoint 4 CSG-3103-ASR920
pe-role u-pe
assoc-npe AGG-3003-ASR9K
if-type GigabitEthernet
if-num 0/0/2
inst-id 1340
encap dot1q
vlan-id 1340
l2vpn-vfi-instance context NCS1340
  vpn-id 1340
  bridge-group-instance
  bridge-domain
    member
      vfi vfi-name NCS1340
      local-term
        interface GigabitEthernet0/0/2
        inst-id 1340
  }
}
ip-prefix-list
  prefix 100.30.3.0/32
  }
endpoint 5 AGG-3004-ASR9K
pe-role n-pe
inst-id 1340
l2vpn-vfi-instance context NCS1340
  vpn-id 1340
  bridge-group-instance
  bridge-domain
    member
      vfi vfi-name NCS1340
    }

Provider Backbone Bridging Ethernet Virtual Private Network

Here is the sample configuration for the PBB-EVPN with core nodes or N-PE nodes such as AGG-0909-ASR9K, AGG-3004-ASR9K, and AGG-3003-ASR9K. The associated PBB-EVPN service edge points are:

- u-pe node: CSG-1106-ASR920 associated with AGG-0909-ASR9K.
- u-pe node: CSG-3103-ASR920 associated with AGG-3003-ASR9K.

This service can be terminated on the IOS-XE access devices (uses VPWS for the last leg). The PBB core is built using the IOS-XR devices.

Sample NSO CLI configuration:

```
services epn5_0 pbb-evpn-vpws NCS1251
endpoint 1 access-pe CSG-1106-ASR920 pe-role u-pe
if-type GigabitEthernet if-num 0/0/2
inst-id 1251
encap dot1q vlan-id 1251
assoc-npe AGG-0909-ASR9K
endpoint 2 access-pe AGG-0909-ASR9K pe-role n-pe
inst-id 1251
pbb-evpn-instance evi 1251
endpoint 3 access-pe AGG-3003-ASR9K pe-role n-pe
inst-id 1251
pbb-evpn-instance evi 1251
endpoint 4 access-pe CSG-3103-ASR920 pe-role u-pe
if-type GigabitEthernet if-num 0/0/2
inst-id 1251
encap dot1q vlan-id 1251
assoc-npe AGG-3003-ASR9K
endpoint 5 access-pe AGG-3004-ASR9K pe-role n-pe
if-type GigabitEthernet inst-id 1251
```

Configuration pushed by NSO:
Service output:

show full-configuration services epn5_0 pbb-evpn-vpws
services epn5_0 pbb-evpn-vpws NCS1251
endpoint 1 CSG-1106-ASR920
  pe-role u-pe
  assoc-npe AGG-0909-ASR9K
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1251
  encap dot1q
  vlan-id 1251
  l2vpn-xc
  rem-pe-lpbk 100.9.9.0
  vc-id 12511
  !
  ip-prefix-list
  prefix 100.9.9.0/32
  !
endpoint 2 AGG-0909-ASR9K
  pe-role n-pe
  inst-id 1251
  pbb-evpn-instance evi 1251
!
endpoint 3 AGG-3003-ASR9K
  pe-role n-pe
  inst-id 1251
  pbb-evpn-instance evi 1251
!
endpoint 4 CSG-3103-ASR920
  pe-role u-pe
  assoc-npe AGG-3003-ASR9K
  if-type GigabitEthernet
if-num 0/0/2
inst-id 1251
encap dot1q
vlan-id 1251
l2vpn-xc
rem-pe-lpbk 100.30.3.0
vc-id 12514
!
ip-prefix-list
prefix 100.30.3.0/32
!
!

show full-configuration services epn5_0 pbb-evpn-common source-bmac
CHAPTER 6

Detailed Configuration Steps for Virtual Private Wire Service with Label Distribution Protocol Signaling

The VPWS allows the two L2VPN Provider Edge nodes to tunnel the L2VPN traffic over Multiprotocol Label Switching (MPLS) cloud. The two attachment circuits connecting at each L2VPN PE are linked by Pseudowire over the MPLS cloud. Each PE needs to have a MPLS label, to reach the loopback of the remote PE. The label can be learned by segment routing or LDP.

This chapter contains the following sections:

- Configuration Steps, page 65
- Final Network Service Orchestrator Configuration CLI Set, page 68
- Native Configuration Created by Network Service Orchestrator, page 68
- View Service Configuration, page 68
- Network Service Orchestrator Based Service Assurance, page 69
- Service Removal Using Network Service Orchestrator, page 74
- Service Augmentation, page 75

Configuration Steps

Before You Begin

Add the network elements (NEs) to the NSO and perform the configuration synchronization with the NSO.

Step 1

Choose service type.

```
admin@ncs(config)# services epn5_0 ?
Possible completions:
```
Step 2  Set the unique service name.

```
admin@ncs(config)# services epn5_0 vpws ?
% No entries found
Possible completions:
  Unique service Name
admin@ncs(config)# services epn5_0 vpws NCS1251
```

Step 3  Set the service specific generic parameters, for example, VPWS signaling type.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# vpws-sig ?
Possible completions:
  evpn  ldp-sig  no-sig
admin@ncs(config-epn5_0-vpws/NCS1251)# vpws-sig ldp-sig
```

Step 4  Configure service endpoint

**Note**  The VPWS requires two endpoints.

a)  Provide unique endpoint numeric identifier.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint ?
Possible completions:
  Endpoint identifier range
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1
```

b)  Choose the NE from the service endpoint list.

The list is obtained from the NE's set for NSO configuration.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 ?
Possible completions:
  AGG-0909-ASR9K  AGG-1003-ASR903  AGG-1004-ASR903  AGG-3003-ASR9K  AGG-3004-ASR9K
  AGG-4003-ASR903  AGG-4004-ASR903  AGG-4007-ASR903  CN-ASBR-0008-ASR9K  CN-P-0001-ASR9K
  CSG-1102-ASR920  CSG-1103-ASR920  CSG-1104-ASR920  CSG-3102-ASR920  CSG-3103-ASR920
  FAN-5303-ASR920  FAN-5606-ASR903  FAN-5607-ASR903
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920
```

c)  Choose the relevant PE role for the service endpoint.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role ?
Possible completions:
  n-pe  u-pe
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe
```
d) Choose the interface type and provide the interface-number to terminate the service.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
Possible completions:
GigabitEthernet  Loopback  FW-Ether  TenGigabitEthernet
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num ?
Possible completions:
<string>
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2
```

e) Provide instance-id for the service.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2 inst-id ?
Possible completions:
<int, 1 .. 4000>
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2 inst-id 1251
```

f) Choose the service end interface parameters such as L2 interface encapsulation type and vlan-id.

```
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2 inst-id 1251 encap ?
Possible completions:
dot1q untagged
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2 inst-id 1251 encap dot1q vlan-id ?
Possible completions:
<int, 1 .. 4000>
admin@ncs(config-epn5_0-vpws/NCS1251)# endpoint 1 FAN-5303-ASR920 pe-role u-pe if-type
GigabitEthernet
if-num 0/0/2 inst-id 1251 encap dot1q vlan-id 1251
```

**Note** No check is done for the VLAN number mapping on the node.

**Step 5** Repeat the sub-steps of Step 4 for additional endpoints.

**Note** For VPWS, you need exactly two endpoints. For VPLS or L3VPN, you can have more than two endpoints.

**Step 6** Commit the configuration.
Final Network Service Orchestrator Configuration CLI Set

Here is the summary of all the configuration steps.

```
services epn5_0 vpws NCS1251
vpws-sig ldp-sig
endpoint 1 access-pe FAN-5303-ASR920 pe-role u-pe if-type GigabitEthernet if-num 0/0/2
   inst-id 1251
   encap dot1q vlan-id 1251
endpoint 2 access-pe PAN-5607-ASR903 pe-role u-pe if-type GigabitEthernet if-num 0/4/2
   inst-id 1251
   encap dot1q vlan-id 1251
```

Native Configuration Created by Network Service Orchestrator

The configuration pushed to the node in its native format, for example, IOS-XR or IOS-XE, is given below:

```
native {
  device {
    name FAN-5303-ASR920
    data ethernet evc NCS1251
    
    interface GigabitEthernet0/0/2
      service instance 1251 ethernet NCS1251
      encapsulation dot1q 1251
      rewrite ingress tag pop 1 symmetric
      exit
    exit
    l2vpn xconnect context NCS1251
    interworking ethernet
      member GigabitEthernet0/0/2 service-instance 1251
      member 100.56.7.0 12510 encapsulation mpls
    }
  }
  device {
    name FAN-5607-ASR903
    data ethernet evc NCS1251
    
    interface GigabitEthernet0/4/2
      service instance 1251 ethernet NCS1251
      encapsulation dot1q 1251
      rewrite ingress tag pop 1 symmetric
      exit
    exit
    l2vpn xconnect context NCS1251
    interworking ethernet
      member GigabitEthernet0/4/2 service-instance 1251
      member 100.53.3.0 12510 encapsulation mpls
    }
  }
}
```

View Service Configuration

```
admin@ncs(config)# show full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
vpws-sig ldp-sig
endpoint 1 FAN-5303-ASR920
   pe-role u-pe
```
Network Service Orchestrator Based Service Assurance

The Cisco EPN services terminated on IOS-XE devices can be validated out-of-service to assess the proper configuration and performance prior to customer notification and delivery. The Cisco EPN infrastructure leverages the inherent traffic generator and loopback capability on the IOS-XE devices.

Before You Begin

The traffic generation steps have been pre-defined as 100 Kbps, 200 Kbps, 300 Kbps, 400 Kbps, and 500 Kbps, with a test duration of 30 seconds.

Step 1

Choose the traffic generator PE (IOS-XE node resident), remote loopback PE, and the service performance type (Ethernet for L2 service). Also, enable the ip-sla and commit the configuration change.

Note

The node selection is filtered for the specific service instance.

admin@ncs(config-epn5_0-vpws/NCS1251)# serv-act-y1564
admin@ncs(config-serv-act-y1564)# tgen-pe ?
Possible completions:
FAN-5303-ASR920  PAN-5607-ASR903
admin@ncs(config-serv-act-y1564)# tgen-pe FAN-5303-ASR920 ?
Possible completions:
ip-sla  lpbk-pe serv-perf-type <cr>
admin@ncs(config-serv-act-y1564)# tgen-pe FAN-5303-ASR920 lpbk-pe ?
Possible completions:
FAN-5607-ASR903
admin@ncs(config-serv-act-y1564)# tgen-pe FAN-5303-ASR920 lpbk-pe FAN-5607-ASR903
admin@ncs(config-serv-act-y1564)# serv-perf-type ?
Possible completions:
ethernet  ip
admin@ncs(config-serv-act-y1564)# serv-perf-type ethernet
admin@ncs(config-serv-act-y1564)# ip-sla enable
admin@ncs(config-serv-act-y1564)# commit
Commit complete.
Step 2  Enable the traffic internal loopback from the destination PE.

**Note** The loopback is active for 300 seconds. No commit is required to enable the internal loopback.

```bash
admin@ncs(config-epn5_0-vpws/NCS1251)# ip-sla-remote-lpbk
Value for 'internal' [disable,enable]: enable
result
This is an intrusive loopback and the packets matched with the service will not be able to pass through.
Continue? (yes/[no]): yes
PAN-5607-ASR903#
```

Step 3  Start the traffic generator and commit the configuration.

```bash
admin@ncs(config-epn5_0-vpws/NCS1251)# serv-act-y1564 ip-sla-tgen enable
admin@ncs(config-serv-act-y1564)# commit
```

**Note** The ip-sla enable state shows that the service activation configuration is done.

```bash
admin@ncs(config)# show full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
  vpws-sig ldp-sig
  serv-act-y1564
    ip-sla enable
    tgen-pe FAN-5303-ASR920
    lpbk-pe PAN-5607-ASR903
    serv-perf-type ethernet
  !
endpoint 1 FAN-5303-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1251
  encap dot1q
  vlan-id 1251
  l2vpn-xc
    rem-pe-lpbk 100.56.7.0
  !
endpoint 2 PAN-5607-ASR903
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/4/2
  inst-id 1251
  encap dot1q
  vlan-id 1251
  l2vpn-xc
    rem-pe-lpbk 100.53.3.0
  !
```

Step 4  Check the ip-sla statistics using the EPN NSO CLI, for example, ios-cmd "<actual command>".  
Note  This command is executed on the traffic generator PE. Use the specific service instance id, for example, 1251 in above configuration to get the details.

Here is sample output for the test in progress. This test sends the traffic in five stages from 100Kbps to 500Kbps for 30 seconds each, and measures the frame loss, round trip duration, and frame receive deviation for the Tgen Tx packets that are received back on the traffic generator (TGEN) PE.

**Interim result when the test is still on:**

```
admin@ncs(config-epn5_0-vpws/NCS1251)# any ios-cmd "show ip sla statistics 1251"
result
IPSLAs Latest Operation Statistics

IPSLA operation id: 1251
Type of operation: Ethernet Service Performance
Test mode: Two-way Measurement
Steps Tested (kbps): 100 200 300 400 500
Test duration: 30 seconds

Latest measurement: *13:45:42.034 UTC Wed Jan 25 2017
Latest return code: OK
Overall throughput: In Progress

Stage 1(100 kbps):

Stats:
IR(kbps)  FL   FLR  Avail  FTD Min/Avg/Max  FDV Min/Avg/Max
95      0     0.00% 100.00% 2498.36ms/2417.01ms/2426.27ms 2.72us/9.26ms/65.86ms
Tx Packets: 1392  Tx Bytes: 356352
Rx Packets: 1392  Rx Bytes: 381408
Step Duration: 30 seconds

Stage 2 (200 kbps):

Stats:
IR(kbps)  FL   FLR  Avail  FTD Min/Avg/Max  FDV Min/Avg/Max
190     0     0.00% 100.00%  57.02ms/4275.33ms/4279.98ms 2.64us/4.66ms/33.72ms
Tx Packets: 2781  Tx Bytes: 711936
Rx Packets: 2781  Rx Bytes: 761994
Step Duration: 30 seconds

Stage 3 (300 kbps):

Stats:
IR(kbps)  FL   FLR  Avail  FTD Min/Avg/Max  FDV Min/Avg/Max
285     0     0.00% 100.00% 1046.32ms/2843.73ms/2846.82ms 2.72us/3.09ms/21.86ms
Tx Packets: 2640  Tx Bytes: 675840
Rx Packets: 2640  Rx Bytes: 723360
```
Step Duration: 18 seconds

Stage 4 (400 kbps):
Stage 5 (500 kbps):

FAN-5303-ASR920#
Final result after the test execution:

admin@ncs(config-epn5_0-vpws/NCS1251)# any ios-cmd "show ip sla statistics 1251"
result
IPSLAs Latest Operation Statistics

IPSLA operation id: 1251
Type of operation: Ethernet Service Performance
Test mode: Two-way Measurement
Steps Tested (kbps): 100 200 300 400 500
Test duration: 30 seconds

Latest return code: OK

Overall Throughput: 490 kbps

Stage 1 (100 kbps):

Stats:
IR(kbps) FL FLR Avail FTD Min/Avg/Max FDV Min/Avg/Max
95 0 0.00% 100.00% 2498.36ms/2417.01ms/2426.27ms 2.72us/9.26ms/65.86ms
Tx Packets: 1392 Tx Bytes: 356352
Rx Packets: 1392 Rx Bytes: 381408
Step Duration: 30 seconds

Stage 2 (200 kbps):

Stats:
IR(kbps) FL FLR Avail FTD Min/Avg/Max FDV Min/Avg/Max
190 0 0.00% 100.00% 57.02ms/4275.33ms/4279.98ms 2.64us/4.66ms/33.72ms
Tx Packets: 2781 Tx Bytes: 711936
Rx Packets: 2781 Rx Bytes: 761994
Step Duration: 30 seconds

Stage 3 (300 kbps):

Stats:
IR(kbps) FL FLR Avail FTD Min/Avg/Max FDV Min/Avg/Max
285 0 0.00% 100.00% 1046.32ms/971.33ms/974.41ms 2.72us/3.09ms/21.93ms
Tx Packets: 4170 Tx Bytes: 1067520
Rx Packets: 4170 Rx Bytes: 1142580
Step Duration: 30 seconds
Stage 4 (400 kbps):

Stats:
IR(kbps) FL FLR Avail FTD Min/Avg/Max FDV Min/Avg/Max
395 0 0.00% 100.00% 2020.95ms/1943.99ms/1946.22ms 2.72us/2.23ms/15.57ms
Tx Packets: 5790 Tx Bytes: 1482240
Rx Packets: 5790 Rx Bytes: 1586460
Step Duration: 30 seconds

Stage 5 (500 kbps):

Stats:
IR(kbps) FL FLR Avail FTD Min/Avg/Max FDV Min/Avg/Max
490 0 0.00% 100.00% 3991.43ms/3915.34ms/3917.14ms 2.64us/1.79ms/12.55ms
Tx Packets: 7179 Tx Bytes: 1837824
Rx Packets: 7179 Rx Bytes: 1967046
Step Duration: 30 seconds

FAN-5303-ASR920#

Step 5
Disable the internal loopback, if the test duration is less than 300 secs.

admin@ncs(config-epn5_0-vpws/NCS1251)# ip-sla-remote-lpbk internal disable

Note If the traffic internal loopback function is timed out, the result would be:

admin@ncs(config-epn5_0-vpws/NCS1251)# ip-sla-remote-lpbk internal disable
result
Cannot find the corresponding session 1 on GigabitEthernet0/4/2
PAN-5607-ASR903#

Step 6
Disable the traffic generation.

admin@ncs(config-epn5_0-vpws/NCS1251)# serv-act-y1564 ip-sla-tgen disable
admin@ncs(config-serv-act-y1564)# commit
Commit complete.

Step 7
Disable the ip-sla to revert the configuration to the service active state.

admin@ncs(config-epn5_0-vpws/NCS1251)# serv-act-y1564 ip-sla disable
admin@ncs(config-serv-act-y1564)# commit
Commit complete.

Confirm that the ip-sla has been disabled with the below show command.

admin@ncs(config)# show full-configuration services epn5_0 vpws NCS1251
services epn5_0 vpws NCS1251
Service Removal Using Network Service Orchestrator

The configuration given below is applicable only for the services created using NSO.

```
vpws-sig ldp-sig
serv-act-y1564
ip-sla disable
tgen-pe FAN-5303-ASR920
lpbk-pe PAN-5607-ASR903
serv-perf-type ethernet
!
endpoint 1 FAN-5303-ASR920
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/0/2
  inst-id 1251
  encap dot1q
  vlan-id 1251
  l2vpn-xc
    rem-pe-lpbk 100.56.7.0
    !
!
endpoint 2 PAN-5607-ASR903
  pe-role u-pe
  if-type GigabitEthernet
  if-num 0/4/2
  inst-id 1251
  encap dot1q
  vlan-id 1251
  l2vpn-xc
    rem-pe-lpbk 100.53.3.0
    !
!
```

admin@ncs(config)# no services epn5_0 vpws NCS1251
admin@ncs(config)# commit
Commit complete
Service Augmentation

It is possible to augment the service with new functionality or change the existing functionality. Here is an example to augment the Layer-3 VPN to configure VRF forwarding instance name.

**Step 1**
Augment the YANG model to include the new attribute for the endpoint list.

```yaml
container vrf {
    when "(../.service-type = 'l3vpn') or (../.service-type = 'l3vpn-pwhe')";
    leaf forwarding {
        type string;
    }
}
```

*Note* This configuration option is available only for L3 VPN service.

**Step 2**
Augment the Python mapping logic to use this new configuration option, and traverse the epn5_o_services.py for the service l3vpn. Augment the function `setup_l3vpn_vrf` with the following:

```python
# VRF YANG Parameters
if str(point.vrf.forwarding) != 'None':
    serv_vrf_name = point.vrf.forwarding
else:
    serv_vrf_name = l3_serv_name
    point.vrf.forwarding = l3_serv_name
```

*Note* The default VRF forwarding instance name is retained as l3_serv_name.

**Step 3**
Pass the attribute to the l3vpn-v2-template.xml template.

```python
l3_tv.add('VRF_NAME', serv_vrf_name)
```

**Step 4**
Update the XML template to use the passed parameter "VRF NAME" appropriately.

```xml
<definition>
    $VRF_NAME</name>
</vrf>
</config-template>
</device>
</config-template>
```

---

Detailed Configuration Steps for Virtual Private Wire Service with Label Distribution Protocol Signaling

Service Augmentation
The Cisco Evolved Programmable Network solution has the following guides:

- Cisco Evolved Programmable Network Transport Design Guide, Release 5.0
- Cisco Evolved Programmable Network Services Design Guide, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Large Network with End to End Segment Routing, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Small Network with End-to-End Segment Routing, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Large Network with End to End Programmable Segment Routing, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Inter-AS Large Network with End to End Segment Routing, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Large Network with Segment Routing and LDP Interworking, Release 5.0
- Cisco Evolved Programmable Network Implementation Guide for Large Network with Layer2 Access into Segment Routing Transport, Release 5.0