Module 01: Refresher—VPNs and MPLS

Part 1: The Conceptual Lecture (The Why)

The Fundamental Problem

Imagine you're a large corporation with offices in New York, London, and Tokyo. Each office has its own local network with computers, servers, and printers. Now, you want employees in Tokyo to access a file server in New York as if it were sitting right next to them. How do you connect these geographically separated networks securely over the public Internet or a shared service provider network?

This is the fundamental problem that Virtual Private Networks (VPNs) solve.

What is a VPN?

A **Virtual Private Network (VPN)** creates a secure, private communication channel over a shared or public network infrastructure. Think of it like having a private tunnel through a public highway—only you can use your tunnel, even though the highway is shared by everyone.

```
Office A                    Service Provider Network                    Office B
[10.1.0.0/24]  <------------ Virtual Private Network ------------>  [10.2.0.0/24]
     |                           (Logical Tunnel)                          |
   LAN A                                                                LAN B
```

Traditional VPN Challenges

Early VPN implementations faced several challenges:

- **Scalability**: Point-to-point tunnels created a mesh of connections ($n * (n-1)/2$ tunnels for n sites)
- **Management Complexity**: Each tunnel required individual configuration
- **Performance**: Encryption/decryption overhead
- **Service Provider Limitations**: Difficult to offer VPN as a service

Enter MPLS: The Game Changer

Multi-Protocol Label Switching (MPLS) revolutionized how service providers could offer VPN services. Instead of creating individual tunnels, MPLS uses a clever labeling system.

How MPLS Works

Think of MPLS like a shipping company's package sorting system:

1. **Package Arrival (Ingress)**: When a package (packet) arrives at the sorting facility (Provider Edge router), it gets a shipping label
2. **Express Sorting (Transit)**: Workers (Provider routers) only look at the outer label to determine the next sorting station—they don't need to open and examine the package contents
3. **Final Delivery (Egress)**: At the destination facility, the label is removed and the package is delivered locally

```
Customer Packet Flow through MPLS Network:

CE1 ----> PE1 ----> P1 ----> P2 ----> PE2 ----> CE2
         (Push)   (Swap)   (Swap)   (Pop)
         Label    Label    Label    Label

Where:
CE = Customer Edge (your router)
PE = Provider Edge (service provider's edge router)
P = Provider (core router in service provider network)
```

MPLS Label Structure

An MPLS label is a 32-bit identifier with this structure:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Label (20 bits)        | EXP |S|   TTL (8 bits) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Label: 20-bit label value
EXP: 3-bit experimental (used for QoS)
S: 1-bit bottom of stack indicator
TTL: 8-bit time-to-live
```

Layer 3 VPNs vs Layer 2 VPNs

The key difference lies in what the service provider network understands about your traffic:

MPLS Layer 3 VPNs (L3VPN)

- Service provider participates in customer routing
- PE routers learn and store customer routes
- Customer sees the provider network as a router

```
L3VPN Operation:
CE1 announces routes --> PE1 learns routes --> PE1 shares with PE2 --> PE2 announces to CE2
     (via BGP/OSPF)        (stores in VRF)       (via MP-BGP)           (via BGP/OSPF)
```

MPLS Layer 2 VPNs (L2VPN)

- Service provider provides transparent Layer 2 connectivity
- PE routers don't learn customer routes
- Customer sees the provider network as a switch/wire

```
L2VPN Operation:
CE1 sends frame --> PE1 encapsulates --> Transport over MPLS --> PE2 decapsulates --> CE2 receives
   (Ethernet)        (adds PW label)      (label switching)       (removes label)      (Ethernet)
```

Why Choose Layer 2 VPNs?

1. **Protocol Independence**: Transport any Layer 3 protocol (IPv4, IPv6, IPX, AppleTalk)
2. **Simplified Provider Network**: No need to support customer routing protocols
3. **Customer Control**: Complete control over routing policies
4. **Legacy Application Support**: Some applications require Layer 2 adjacency
5. **Migration Simplicity**: Easier to migrate existing LANs

Part 2: The Junos CLI Masterclass (The How)

MPLS Configuration Foundation

Before configuring any L2VPN service, we need a working MPLS network. Let's build this step by step.

Step 1: Enable MPLS on Interfaces

```
[edit]
user@PE1# set interfaces ge-0/0/0 description "To P1"
user@PE1# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
user@PE1# set interfaces ge-0/0/0 unit 0 family mpls

user@PE1# set interfaces ge-0/0/1 description "To CE1"
user@PE1# set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.1/30
```

Why: The `family mpls` statement enables MPLS label processing on the interface. Without this, the interface will discard MPLS packets.

Step 2: Configure MPLS Protocol

```
[edit protocols mpls]
user@PE1# set interface ge-0/0/0.0
user@PE1# set interface lo0.0
```

Why: This tells Junos which interfaces should participate in MPLS forwarding. Always include the loopback for LSP termination.

Step 3: Configure Label Distribution

For MPLS to work, routers need a way to distribute labels. We'll use LDP (Label Distribution Protocol):

```
[edit protocols ldp]
user@PE1# set interface ge-0/0/0.0
user@PE1# set interface lo0.0
```

Or for RSVP-signaled LSPs:

```
[edit protocols rsvp]
user@PE1# set interface ge-0/0/0.0

[edit protocols mpls]
user@PE1# set label-switched-path TO-PE2 to 2.2.2.2
user@PE1# set label-switched-path TO-PE2 primary via-p1
```

Step 4: Interior Gateway Protocol (IGP)

MPLS relies on the IGP for reachability. Here's OSPF configuration:

```
[edit protocols ospf]
user@PE1# set area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p
user@PE1# set area 0.0.0.0 interface lo0.0 passive
```

Complete PE Router Base Configuration

Here's a complete, annotated base configuration for a PE router:

```
## System Basics
set system host-name PE1
set interfaces lo0 unit 0 family inet address 1.1.1.1/32

## Core-Facing Interface
set interfaces ge-0/0/0 description "To P1 - Core Network"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
set interfaces ge-0/0/0 unit 0 family mpls

## Customer-Facing Interface
set interfaces ge-0/0/1 description "To CE1 - Customer ABC"
set interfaces ge-0/0/1 unit 0 family inet address 10.0.0.1/30

## MPLS Configuration
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface lo0.0

## Label Distribution via LDP
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0

## IGP for Loopback Reachability
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

```
## BGP for L2VPN Signaling (prepare for later modules)
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 1.1.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP neighbor 2.2.2.2 description PE2
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify MPLS Interface Status

```
user@PE1> show mpls interface
Interface        State       Administrative groups (x: extended)
ge-0/0/0.0       Up          <none>
lo0.0            Up          <none>
```

What to Look For: All MPLS interfaces should show "Up". If an interface shows "Down", check:

- Is `family mpls` configured on the interface?
- Is the interface included under `[edit protocols mpls]`?

2. Verify Label Distribution

For LDP:

```
user@PE1> show ldp neighbor
Address                          Interface       Label space ID     Hold time
192.168.1.2                      ge-0/0/0.0      2.2.2.2:0          14
```

For RSVP:

```
user@PE1> show rsvp neighbor
RSVP neighbor: 1 learned
Address            Idle Up/Dn LastChange HelloInt HelloTx/Rx MsgRcvd
192.168.1.2        0    1/0   00:05:23   9        35/35      5
```

3. Verify MPLS LSPs

```
user@PE1> show mpls lsp
Ingress LSP: 1 sessions
To              From            State Rt P     ActivePath       LSPname
2.2.2.2         1.1.1.1         Up     0 *     via-p1           TO-PE2
```

Common Troubleshooting Scenarios

Scenario 1: MPLS LSP Not Establishing

Symptom: `show mpls lsp` shows State as "Dn" (Down)

Diagnostic Commands:

```
user@PE1> show mpls lsp extensive
...
  Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 20)
    192.168.1.2 S 192.168.2.2 S
  Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
  5 Oct 25 10:15:45.123 CSPF failed: no route to 2.2.2.2[6 times]
```

Cause: No IGP route to destination

Solution:

1. Verify IGP is running: `show ospf neighbor`
2. Check if destination loopback is advertised: `show route 2.2.2.2`
3. Fix OSPF configuration if needed

Scenario 2: LDP Session Not Forming

Symptom: `show ldp session` shows no neighbors

Diagnostic Commands:

```
user@PE1> show ldp session
  Neighbor                       State       Hold time  Adv. Mode
user@PE1> show ldp interface
Interface            Label space ID        Nbr count   Next hello
ge-0/0/0.0           1.1.1.1:0             0           2
```

Cause: LDP hellos not being exchanged

Solution:

1. Check interface configuration: `show configuration protocols ldp`
2. Verify no firewall filters blocking LDP (UDP 646): `show interfaces ge-0/0/0.0 extensive | match filter`
3. Ensure both sides have LDP configured

Scenario 3: Routing Loop Due to Incorrect IGP Metrics

Symptom: Traceroute shows packets looping between routers

Diagnostic Commands:

```
user@PE1> traceroute mpls ldp 2.2.2.2
  1  192.168.1.2 (192.168.1.2)  0.561 ms  0.434 ms  0.396 ms
     MPLS Label=299776 CoS=0 TTL=1 S=1
  2  192.168.1.1 (192.168.1.1)  0.543 ms  0.445 ms  0.419 ms
     MPLS Label=299776 CoS=0 TTL=1 S=1
  3  192.168.1.2 (192.168.1.2)  0.561 ms  0.434 ms  0.396 ms
```

Cause: IGP metrics incorrectly configured, causing bidirectional forwarding

Solution:

1. Check OSPF costs: `show ospf interface detail`
2. Ensure consistent metrics across the network
3. Fix with: `set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10`

Scenario 4: MPLS MTU Issues

Symptom: Small packets work, large packets fail

Diagnostic Commands:

```
user@PE1> ping 2.2.2.2 size 1500 do-not-fragment
PING 2.2.2.2: 1500 data bytes
ping: sendto: Message too long
```

Cause: MPLS overhead not accounted for in interface MTU

Solution:

```
set interfaces ge-0/0/0 mtu 1514    # Increase MTU to accommodate MPLS labels
# Or enable MPLS MTU signaling
set protocols ldp interface ge-0/0/0.0 mtu-discovery
```

Summary Verification Checklist

Before proceeding to L2VPN configuration, ensure:

- [ ] MPLS interfaces are up: `show mpls interface`
- [ ] Label distribution protocol active: `show ldp neighbor` or `show rsvp neighbor`
- [ ] LSPs established to all PE routers: `show mpls lsp`
- [ ] Full IGP reachability: `show route protocol ospf`
- [ ] No interface errors: `show interfaces ge-* extensive | match error`

This foundation is critical—all L2VPN services build upon a working MPLS core.

Module 02: The Different Flavors of Layer 2 VPN

Part 1: The Conceptual Lecture (The Why)

The Evolution of Layer 2 Connectivity

Imagine you're running a campus network in the 1990s. All your buildings are connected with Ethernet switches, creating one big, happy Layer 2 network. Life is simple—plug in a device anywhere, and it can talk to any other device as if they're on the same wire.

Now fast-forward to today. Your company has grown, and you have offices worldwide. You want that same "plug and play" simplicity, but your offices are separated by the Internet or a service provider's network. This is where Layer 2 VPNs come to the rescue.

Understanding Pseudowires: The Foundation

A **pseudowire** is like a virtual cable that extends Layer 2 connectivity across a Layer 3 network. Think of it as a tunnel that makes two distant switches believe they're connected back-to-back.

```
Traditional Local Connection:
Switch A [Port 1] <-------- Physical Cable --------> [Port 1] Switch B

Pseudowire Connection:
Switch A [Port 1] <---- Virtual Cable (Pseudowire) ----> [Port 1] Switch B
                        (Over MPLS/IP Network)
```

The Pseudowire Terminology Maze

The industry uses many terms for similar concepts. Let's clarify:

1. **Pseudowire (PW)**: The virtual connection itself
2. **L2VPN**: BGP-signaled point-to-point pseudowire (Juniper terminology)
3. **L2Circuit**: LDP-signaled point-to-point pseudowire (Juniper terminology)
4. **VPWS (Virtual Private Wire Service)**: Industry term for point-to-point service
5. **xconnect**: Cisco's term for pseudowire configuration
6. **EoMPLS (Ethernet over MPLS)**: Specific type carrying Ethernet frames

```
Terminology Mapping:
┌──────────────────┬──────────────────┬──────────────────┬──────────────────┐
│ Juniper Term     │ Cisco Term       │ Industry Term    │ What It Is       │
├──────────────────┼──────────────────┼──────────────────┼──────────────────┤
│ L2VPN            │ BGP xconnect     │ BGP VPWS         │ BGP PW           │
│ L2Circuit        │ LDP xconnect     │ LDP VPWS         │ LDP PW           │
│ VPLS             │ VPLS             │ VPLS             │ Multipoint       │
│ EVPN             │ EVPN             │ EVPN             │ Next-gen L2      │
└──────────────────┴──────────────────┴──────────────────┴──────────────────┘
```

Point-to-Point vs. Multipoint Services

Point-to-Point Pseudowires (L2VPN/L2Circuit)

These create a direct connection between exactly two sites:

```
Site A <------------ Pseudowire -------------> Site B
         One-to-One Connection Only
```

Use Cases:

- Connecting two data centers
- Extending a VLAN between two locations
- Backhauling remote site to central location

Multipoint Services: Enter VPLS

Virtual Private LAN Service (VPLS) extends the pseudowire concept to support multiple sites, creating a virtual switch in the service provider network.

```
        Site A
          |
    +-----------+
    |   VPLS    |---- Site B
    | (Virtual  |
    |  Switch)  |---- Site C
    +-----------+
          |
        Site D

All sites can communicate with each other (full mesh)
```

How VPLS Works

VPLS combines multiple pseudowires to create a multipoint service:

1. **Full Mesh of Pseudowires**: Every PE establishes pseudowires to every other PE
2. **MAC Learning**: PEs learn MAC addresses like switches
3. **Forwarding**: Unknown unicast, broadcast, and multicast are flooded
4. **Loop Prevention**: Split-horizon rule prevents loops

```
VPLS MAC Learning Example:

1. Host A (MAC: AA:AA) at Site 1 sends frame to Host B
2. PE1 learns: MAC AA:AA is behind AC from CE1
3. PE1 floods frame to all other PEs (PE2, PE3)
4. PE2 learns: MAC AA:AA is reachable via PW to PE1
5. When Host B replies, PE2 forwards directly to PE1 (no flooding)
```

The Revolution: EVPN

Ethernet VPN (EVPN) is the newest and most advanced L2VPN technology. Think of it as "VPLS done right" with these improvements:

EVPN Advantages Over VPLS

1. **Control Plane MAC Learning**: MACs are advertised via BGP, not learned in data plane
2. **Active-Active Multihoming**: Multiple PEs can actively forward traffic for same site
3. **Optimized Multicast**: No unnecessary flooding
4. **Integrated Layer 2/3**: Seamless inter-subnet routing

```
VPLS MAC Learning (Old Way):          EVPN MAC Learning (New Way):
1. Frame arrives                      1. Frame arrives
2. Learn MAC from frame               2. Learn MAC from frame
3. Flood to all PEs                  3. Advertise MAC via BGP
4. All PEs learn from flooded frame  4. Only interested PEs install MAC
   (Inefficient)                         (Efficient)
```

Choosing the Right Technology

```
Decision Tree:

Need Layer 2 extension?
    │
    ├─ Two sites only?
    │     │
    │     ├─ Yes → Use L2VPN (BGP) or L2Circuit (LDP)
    │     │        - Simpler configuration
    │     │        - Less control plane overhead
    │     │
    │     └─ No → Multiple sites?
    │              │
    │              ├─ Legacy network? → Use VPLS
    │              │   - Mature technology
    │              │   - Wide vendor support
    │              │
    │              └─ New deployment? → Use EVPN
    │                  - Better scale
    │                  - Advanced features
    │                  - Future-proof
    │
    └─ Need optimal routing? → Definitely EVPN
        - Supports L2 + L3
        - Distributed gateway
```

Part 2: The Junos CLI Masterclass (The How)

Configuration Hierarchy Overview

Understanding where different L2VPN configurations live in Junos:

```
[edit]
├── interfaces          # Physical and logical interface configuration
├── routing-instances   # VPLS and EVPN instances
├── protocols
│   ├── mpls           # MPLS and LSP configuration
│   ├── ldp            # L2Circuit signaling
│   ├── bgp            # L2VPN and EVPN signaling
│   └── l2circuit      # L2Circuit-specific configuration
└── policy-options     # Routing policies for BGP
```

Basic L2VPN Service Configuration Patterns

Pattern 1: L2VPN (BGP-Signaled Pseudowire)

```
## Step 1: Configure the Attachment Circuit (customer-facing interface)
set interfaces ge-0/0/1 description "Customer ABC - Site 1"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## Step 2: Configure BGP for L2VPN signaling
set protocols bgp group IBGP family l2vpn signaling

## Step 3: Create the L2VPN routing instance
set routing-instances CUSTOMER-ABC instance-type l2vpn
set routing-instances CUSTOMER-ABC interface ge-0/0/1.0
set routing-instances CUSTOMER-ABC route-distinguisher 1.1.1.1:100
set routing-instances CUSTOMER-ABC vrf-target target:65000:100
set routing-instances CUSTOMER-ABC protocols l2vpn encapsulation-type ethernet
```

```
set routing-instances CUSTOMER-ABC protocols l2vpn site SITE1 site-identifier 1
set routing-instances CUSTOMER-ABC protocols l2vpn site SITE1 interface ge-0/0/1.0 remote-site-id 2
```

Why each command:

- `ethernet-ccc` : Circuit Cross-Connect mode for L2 transport
- `family ccc` : Enables Circuit Cross-Connect on the logical interface
- `route-distinguisher` : Makes routes unique in BGP
- `vrf-target` : Controls which PEs import these routes
- `site-identifier` : Unique ID for this site in the L2VPN
- `remote-site-id` : The site ID we want to connect to

Pattern 2: L2Circuit (LDP-Signaled Pseudowire)

```
## Step 1: Configure the Attachment Circuit
set interfaces ge-0/0/1 description "Customer XYZ - Site 1"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## Step 2: Configure the L2Circuit
set protocols l2circuit neighbor 2.2.2.2 interface ge-0/0/1.0
set protocols l2circuit neighbor 2.2.2.2 interface ge-0/0/1.0 virtual-circuit-id 100
set protocols l2circuit neighbor 2.2.2.2 interface ge-0/0/1.0 no-control-word
```

Why each command:

- `neighbor 2.2.2.2` : The remote PE's loopback address
- `virtual-circuit-id 100` : Must match on both PEs (like a circuit ID)
- `no-control-word` : Disables control word for compatibility

Pattern 3: VPLS Configuration

```
## Step 1: Configure customer-facing interface
set interfaces ge-0/0/1 description "Customer LAN - Site 1"
set interfaces ge-0/0/1 unit 0 family bridge interface-mode trunk
set interfaces ge-0/0/1 unit 0 family bridge vlan-id-list 100-200

## Step 2: Create VPLS routing instance
set routing-instances VPLS-CUSTOMER instance-type vpls
set routing-instances VPLS-CUSTOMER interface ge-0/0/1.0
set routing-instances VPLS-CUSTOMER route-distinguisher 1.1.1.1:200
set routing-instances VPLS-CUSTOMER vrf-target target:65000:200
set routing-instances VPLS-CUSTOMER protocols vpls site-range 10
set routing-instances VPLS-CUSTOMER protocols vpls site SITE1 site-identifier 1
set routing-instances VPLS-CUSTOMER protocols vpls site SITE1 interface ge-0/0/1.0
```

Why each command:

- `family bridge` : Enables bridge (switching) functionality
- `interface-mode trunk` : Allows multiple VLANs
- `site-range 10` : Allows site IDs 1-10 (must be $\geq$ number of sites)
- `instance-type vpls` : Creates a VPLS instance

Pattern 4: EVPN Configuration

```
## Step 1: Configure customer-facing interface
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 encapsulation flexible-ethernet-services
set interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 family bridge

## Step 2: Configure BGP for EVPN
```

```
set protocols bgp group IBGP family evpn signaling

## Step 3: Create EVPN routing instance
set routing-instances EVPN-CUSTOMER instance-type evpn
set routing-instances EVPN-CUSTOMER vlan-id 100
set routing-instances EVPN-CUSTOMER interface ge-0/0/1.100
set routing-instances EVPN-CUSTOMER route-distinguisher 1.1.1.1:300
set routing-instances EVPN-CUSTOMER vrf-target target:65000:300
set routing-instances EVPN-CUSTOMER protocols evpn
```

Complete Reference Configuration

Here's a complete PE configuration supporting all L2VPN types:

```
## System Configuration
set system host-name PE1

## Interfaces
set interfaces lo0 unit 0 family inet address 1.1.1.1/32

## Core-facing
set interfaces ge-0/0/0 description "To Core"
set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/30
set interfaces ge-0/0/0 unit 0 family mpls

## Customer-facing for L2VPN/L2Circuit
set interfaces ge-0/0/1 description "L2VPN Customer"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## Customer-facing for VPLS/EVPN
set interfaces ge-0/0/2 description "VPLS/EVPN Customer"
set interfaces ge-0/0/2 flexible-vlan-tagging
set interfaces ge-0/0/2 encapsulation flexible-ethernet-services
set interfaces ge-0/0/2 unit 100 vlan-id 100
set interfaces ge-0/0/2 unit 100 family bridge

## MPLS
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface lo0.0

## LDP
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0

## BGP
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 1.1.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP family evpn signaling
set protocols bgp group IBGP neighbor 2.2.2.2

## OSPF
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0

## L2VPN Instance
set routing-instances L2VPN-A instance-type l2vpn
set routing-instances L2VPN-A interface ge-0/0/1.0
set routing-instances L2VPN-A route-distinguisher 1.1.1.1:100
set routing-instances L2VPN-A vrf-target target:65000:100
set routing-instances L2VPN-A protocols l2vpn encapsulation-type ethernet
```

```
set routing-instances L2VPN-A protocols l2vpn site SITE1 site-identifier 1
set routing-instances L2VPN-A protocols l2vpn site SITE1 interface ge-0/0/1.0 remote-site-id 2

## L2Circuit
set protocols l2circuit neighbor 2.2.2.2 interface ge-0/0/1.0 virtual-circuit-id 100

## VPLS Instance
set routing-instances VPLS-A instance-type vpls
set routing-instances VPLS-A interface ge-0/0/2.100
set routing-instances VPLS-A route-distinguisher 1.1.1.1:200
set routing-instances VPLS-A vrf-target target:65000:200
set routing-instances VPLS-A protocols vpls site-range 10
set routing-instances VPLS-A protocols vpls site SITE1 site-identifier 1

## EVPN Instance
set routing-instances EVPN-A instance-type evpn
set routing-instances EVPN-A vlan-id 100
set routing-instances EVPN-A interface ge-0/0/2.100
set routing-instances EVPN-A route-distinguisher 1.1.1.1:300
set routing-instances EVPN-A vrf-target target:65000:300
set routing-instances EVPN-A protocols evpn
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands by Service Type

Verifying Service Type Capabilities

```
user@PE1> show route table bgp.l2vpn.0
user@PE1> show route table bgp.evpn.0
user@PE1> show ldp session
```

What to Look For:

- BGP tables should exist if BGP families are configured correctly
- LDP sessions should be established for L2Circuit support

Common Troubleshooting Scenarios

Scenario 1: Wrong Encapsulation Type

Symptom: L2VPN/L2Circuit configured but no connectivity

Diagnostic Commands:

```
user@PE1> show interfaces ge-0/0/1
Physical interface: ge-0/0/1, Enabled, Physical link is Up
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514
  Logical interface ge-0/0/1.0
    Flags: Up SNMP-Trapping 0x4000000 Encapsulation: Ethernet-CCC

user@PE1> show l2circuit connections
Layer-2 Circuit Connections:
neighbor    interface    st  Time last up         # Up trans
2.2.2.2     ge-0/0/1.0   EM                       0
  Remote PE: 2.2.2.2, Negotiated control-word: No
  Incoming label: 299776, Outgoing label: 299888
  Negotiated PW status TLV: No
  Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET
  Flow Label Transmit: No, Flow Label Receive: No
  Remote PE: 2.2.2.2, Negotiated control-word: No (mtu-mismatch)
```

Cause: Encapsulation mismatch (EM status)

Solution:

```
## Check remote PE configuration
user@PE2> show configuration interfaces ge-0/0/1
encapsulation vlan-ccc;  ## Wrong! Should be ethernet-ccc

## Fix on PE2:
delete interfaces ge-0/0/1 encapsulation vlan-ccc
set interfaces ge-0/0/1 encapsulation ethernet-ccc
```

Scenario 2: VPLS MAC Learning Issues

Symptom: VPLS established but hosts can't communicate

Diagnostic Commands:

```
user@PE1> show vpls connections
Instance: VPLS-A
  Local site: SITE1 (1)
  Number of local interfaces: 1
  Number of local interfaces up: 1
  ge-0/0/2.100
  Neighbor                Type  St     Time last up          # Up trans
  2.2.2.2(vpls-id 2)      rmt   Up     Oct 25 10:30:45 2023  1
    Remote PE: 2.2.2.2, Negotiated control-word: No
    Incoming label: 262145, Outgoing label: 262146
    Local interface: vt-1/0/0.1048576, Status: Up, Encapsulation: VPLS

user@PE1> show vpls mac-table instance VPLS-A
MAC address        Type       Logical interface
aa:bb:cc:dd:ee:01  Learned    ge-0/0/2.100
aa:bb:cc:dd:ee:02  Learned    vt-1/0/0.1048576
```

Cause: If MAC table is empty, check for:

1. MAC learning disabled
2. Storm control dropping traffic
3. Interface not in forwarding state

Solution:

```
## Enable MAC learning if disabled
set routing-instances VPLS-A protocols vpls mac-table-size 5000

## Check for storm control
show routing-instances VPLS-A bridge-domains
```

Scenario 3: EVPN Type Mismatch

Symptom: EVPN neighbors established but no MAC routes exchanged

Diagnostic Commands:

```
user@PE1> show evpn instance EVPN-A extensive
Instance: EVPN-A
  Route Distinguisher: 1.1.1.1:300
  Encapsulation type: VXLAN
  MAC database status               Local  Remote
    Total MAC addresses:                1       0
    Default gateway MAC addresses:      0       0
  Number of local interfaces: 1 (1 up)

user@PE1> show route advertising-protocol bgp 2.2.2.2 table EVPN-A.evpn.0
```

```
EVPN-A.evpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  3:1.1.1.1:300::100::1.1.1.1/248
*                         Self                         100        I
```

Cause: EVPN instance type mismatch (VLAN-based vs VLAN-aware)

Solution:

```
## Ensure consistent EVPN configuration across PEs
## For VLAN-based:
set routing-instances EVPN-A vlan-id 100

## For VLAN-aware bundle:
delete routing-instances EVPN-A vlan-id
set routing-instances EVPN-A vlans v100 vlan-id 100
```

Scenario 4: BGP Route Target Mismatch

Symptom: L2VPN/VPLS/EVPN routes not imported between PEs

Diagnostic Commands:

```
user@PE1> show route table bgp.l2vpn.0 detail
1.1.1.1:100:1:1/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 1.1.1.1:100
                Next hop type: Indirect, Next hop index: 0
                Next hop: 1.1.1.1 via lo0.0
                Protocol next hop: 1.1.1.1
                Communities: target:65000:100 encapsulation:l2-3

user@PE2> show route table bgp.l2vpn.0 hidden extensive
1.1.1.1:100:1:1/96 (1 entry, 0 announced)
         BGP                    /-101
                Route Distinguisher: 1.1.1.1:100
                Communities: target:65000:100
                Import Accepted: vrf-import:NONE
                Hidden reason: rejected by import policy
```

Cause: Route target mismatch - PE2 not importing routes with target:65000:100

Solution:

```
## Check import targets on PE2
user@PE2> show configuration routing-instances L2VPN-A vrf-target
vrf-target target:65000:999;  ## Wrong target!

## Fix on PE2:
set routing-instances L2VPN-A vrf-target target:65000:100
```

Performance Monitoring Commands

Monitor the health of your L2VPN services:

```
## Monitor pseudowire statistics
user@PE1> monitor interface traffic matching "vt-*"

## Check for drops
user@PE1> show interfaces extensive | match "drops|error"

## Monitor MAC learning rate in VPLS
```

```
user@PE1> show vpls statistics instance VPLS-A

## Monitor EVPN route advertisements
user@PE1> monitor route advertising-protocol bgp 2.2.2.2 table EVPN-A.evpn.0
```

Summary Verification Matrix

Service Type	Key Tables	Key Commands	Common Issues
L2VPN	bgp.l2vpn.0	show l2vpn connections	Site ID mismatch
L2Circuit	mpls.0	show l2circuit connections	VC ID mismatch
VPLS	Instance.vpls.0	show vpls connections	Site range too small
EVPN	Instance.evpn.0	show evpn instance	VLAN ID mismatch

Module 03: L2VPN, aka BGP-Signaled Pseudowires

Part 1: The Conceptual Lecture (The Why)

The Fundamental Concepts

Imagine you're connecting two office buildings with a fiber optic cable. At each end, you have a switch port where you plug in the fiber. The fiber carries whatever you send—it doesn't care about the content. BGP-signaled L2VPNs work the same way, but instead of physical fiber, we use a virtual connection across an MPLS network.

Attachment Circuit (AC): Where Customer Meets Provider

An **Attachment Circuit** is the physical or logical connection between the customer equipment (CE) and the provider equipment (PE). Think of it as the "on-ramp" to the L2VPN highway.

```
Customer Premises          Service Provider Network         Customer Premises

   [CE Router]--------[PE Router]~~~~~~~~~~~~[PE Router]--------[CE Router]
        |                  |                      |                 |
   Attachment         Attachment            Attachment          Attachment
   Circuit (AC)       Circuit (AC)          Circuit (AC)        Circuit (AC)

   <--- Customer --->|<------ Pseudowire -------->|<--- Customer --->
```

Pseudowire Encapsulation: The Packaging

Just like shipping companies need to know if they're handling a letter, package, or pallet, MPLS networks need to know what type of Layer 2 frames they're carrying.

Common encapsulation types:

- **Ethernet**: Raw Ethernet frames (most common)
- **Ethernet-VLAN**: Ethernet frames with single VLAN tag
- **Ethernet-VPLS**: Used for VPLS services

```
Original Customer Frame:
[Ethernet Header][IP Packet][Ethernet FCS]

After Pseudowire Encapsulation:
[MPLS Label][PW Label][Control Word (optional)][Original Frame]
```

Route Distinguishers (RD): Making Routes Unique

Here's a problem: What if two customers both use the same IP subnet (e.g., 10.1.1.0/24)? In BGP, we need unique routes. The Route Distinguisher solves this by adding a unique prefix.

```
Without RD:
Customer A: 10.1.1.0/24  ←─┐
                           ├─→ BGP can't distinguish these!
Customer B: 10.1.1.0/24  ←─┘

With RD:
Customer A: 65000:100:10.1.1.0/24  ←─→ Now unique!
Customer B: 65000:200:10.1.1.0/24  ←─→ Now unique!
```

For L2VPN, the RD makes each site's NLRI (Network Layer Reachability Information) unique:

```
L2VPN NLRI Format:
[Route Distinguisher]:[Site ID]:[Label Block Offset]:[Label Base]:[Label Block Size]

Example:
1.1.1.1:100:1:800000:8
│       │   │ │      └─→ Can allocate 8 labels
│       │   │ └────────→ Starting label value
│       │   └──────────→ Site ID 1
│       └──────────────→ RD for this L2VPN
└──────────────────────→ PE loopback (common practice)
```

Route Targets (RT): The Membership Card

While RD makes routes unique, Route Target determines which PEs should import these routes. Think of RT as a membership card—only PEs with matching membership can join the VPN.

```
PE1 Configuration:                    PE2 Configuration:
L2VPN-A:                             L2VPN-A:
  RD: 1.1.1.1:100                     RD: 2.2.2.2:100
  RT: target:65000:100                 RT: target:65000:100
        ↓                                    ↓
    Exports with                         Imports routes with
    target:65000:100                     target:65000:100

                They can communicate!
```

Site IDs: The Address Within the VPN

Within an L2VPN, each site needs a unique identifier. Site IDs serve as "addresses" that sites use to reach each other.

```
L2VPN "Customer-ABC":
┌──────────────────────────────────────────┐
│  Site 1 (HQ)          Site 2 (Branch-1)  │
│  ID: 1                ID: 2              │
│  NYC                  London             │
│                                          │
│  Site 3 (Branch-2)    Site 4 (Branch-3)  │
│  ID: 3                ID: 4              │
│  Tokyo                Sydney             │
└──────────────────────────────────────────┘

Configuration specifies: Site 1 wants to connect to Site 2
```

The Control Plane: How L2VPNs Are Built

The control plane is like the planning phase of building a road. Before any traffic flows, PEs must:

1. **Advertise Sites**: Each PE tells others about its local sites via BGP
2. **Exchange Labels**: PEs allocate MPLS labels for the pseudowire
3. **Build Forwarding State**: PEs program their hardware for forwarding

```
Control Plane Sequence:

1. PE1 (Site 1) → BGP UPDATE → All PEs
   "I have Site 1, use label 100000 to reach it"

2. PE2 receives update, checks:
   - Do I have a site that wants to connect to Site 1?
   - Yes! Site 2 wants remote-site-id 1

3. PE2 (Site 2) → BGP UPDATE → All PEs
   "I have Site 2, use label 200000 to reach it"

4. PE1 receives update:
   - Site 1 wants remote-site-id 2
   - Programs forwarding: "To reach Site 2, use label 200000"
```

The Data Plane: How Traffic Flows

Once the control plane establishes the pseudowire, the data plane handles actual traffic:

```
Traffic Flow from Site 1 to Site 2:

CE1 → Frame → PE1
              ↓
      PE1 Processing:
      1. Receive on AC (ge-0/0/1)
      2. Lookup destination site
      3. Push PW label (200000)
      4. Push transport label
              ↓
      [Transport Label][200000][Frame]
              ↓
      MPLS Network
              ↓
      PE2 Processing:
      5. Pop transport label
      6. See PW label 200000
      7. Remove PW label
      8. Forward to AC
              ↓
              PE2 → Frame → CE2
```

BGP L2VPN NLRI Details

Let's decode an actual BGP UPDATE message for L2VPN:

```
BGP UPDATE Message Components:

1. AFI/SAFI: 25/65 (L2VPN)
2. NLRI: 1.1.1.1:100:1:800000:8
3. Next-Hop: 1.1.1.1
4. Route Target: target:65000:100
5. Layer 2 Info Extended Community:
   - Encapsulation: Ethernet (5)
   - Control Word: Not Required
   - MTU: 1500

Decoded NLRI:
1.1.1.1:100 = Route Distinguisher
1           = Site ID
```

```
800000      = Label base (starting label)
8           = Label block size
```

Label Block Allocation

L2VPN uses a clever label block system for efficiency:

```
Label Block Concept:

Instead of signaling one label per remote site:
Site 1 → Site 2: Use label 100001
Site 1 → Site 3: Use label 100002
Site 1 → Site 4: Use label 100003

L2VPN signals a block:
Site 1: Starting label 100000, block size 8
- To reach Site 1 from Site 2: Use 100000 + (2-1) = 100001
- To reach Site 1 from Site 3: Use 100000 + (3-1) = 100002
- To reach Site 1 from Site 4: Use 100000 + (4-1) = 100003
```

Part 2: The Junos CLI Masterclass (The How)

L2VPN Configuration Hierarchy

```
[edit routing-instances <instance-name>]
├── instance-type l2vpn          # Defines this as L2VPN
├── interface                    # Attachment circuit(s)
├── route-distinguisher          # Makes BGP routes unique
├── vrf-target                   # Controls route import/export
└── protocols
    └── l2vpn
        ├── encapsulation-type   # Ethernet, vlan, etc.
        ├── control-word         # Optional 4-byte header
        ├── mtu                  # Maximum transmission unit
        └── site <name>
            ├── site-identifier  # Unique ID within L2VPN
            └── interface
                └── remote-site-id  # Which site to connect to
```

Step-by-Step L2VPN Configuration

Step 1: Prepare the Attachment Circuit

```
## For Ethernet encapsulation (all VLANs)
set interfaces ge-0/0/1 description "Customer ABC - Site 1"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## For VLAN encapsulation (specific VLAN)
set interfaces ge-0/0/2 description "Customer XYZ - VLAN 100"
set interfaces ge-0/0/2 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 100 vlan-id 100
set interfaces ge-0/0/2 unit 100 family ccc
```

Why CCC: Circuit Cross-Connect (CCC) family tells Junos this interface is for Layer 2 transport, not Layer 3 routing.

Step 2: Configure BGP for L2VPN Signaling

```
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 1.1.1.1
set protocols bgp group IBGP family l2vpn signaling
```

```
set protocols bgp group IBGP neighbor 2.2.2.2 description "PE2"
set protocols bgp group IBGP neighbor 3.3.3.3 description "PE3"
```

Key Point: All PEs participating in L2VPN must have BGP sessions with `family l2vpn signaling`.

Step 3: Create the L2VPN Instance

```
## Basic L2VPN instance
set routing-instances L2VPN-CUSTOMER-ABC instance-type l2vpn
set routing-instances L2VPN-CUSTOMER-ABC interface ge-0/0/1.0
set routing-instances L2VPN-CUSTOMER-ABC route-distinguisher 1.1.1.1:100
set routing-instances L2VPN-CUSTOMER-ABC vrf-target target:65000:100

## L2VPN protocol configuration
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn mtu 1500
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn no-control-word
```

Step 4: Configure Sites

```
## Local site configuration
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn site HQ site-identifier 1

## Specify which interface belongs to this site
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn site HQ interface ge-0/0/1.0

## Simple point-to-point connection
set routing-instances L2VPN-CUSTOMER-ABC protocols l2vpn site HQ interface ge-0/0/1.0 remote-site-id 2
```

Advanced Configuration Patterns

Pattern 1: Hub-and-Spoke Topology

```
## Hub site (can connect to multiple spokes)
set routing-instances L2VPN-HUB protocols l2vpn site HUB site-identifier 1
set routing-instances L2VPN-HUB protocols l2vpn site HUB interface ge-0/0/1.0

## Spoke sites (each connects only to hub)
set routing-instances L2VPN-SPOKE1 protocols l2vpn site SPOKE1 site-identifier 2
set routing-instances L2VPN-SPOKE1 protocols l2vpn site SPOKE1 interface ge-0/0/2.0 remote-site-id 1

set routing-instances L2VPN-SPOKE2 protocols l2vpn site SPOKE2 site-identifier 3
set routing-instances L2VPN-SPOKE2 protocols l2vpn site SPOKE2 interface ge-0/0/3.0 remote-site-id 1
```

Pattern 2: VLAN Normalization

```
## PE1: Customer uses VLAN 100
set interfaces ge-0/0/1 encapsulation vlan-ccc
set interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 family ccc

## PE2: Same customer uses VLAN 200
set interfaces ge-0/0/1 encapsulation vlan-ccc
set interfaces ge-0/0/1 unit 200 vlan-id 200
set interfaces ge-0/0/1 unit 200 family ccc

## Both PEs: Configure VLAN normalization
set routing-instances L2VPN-NORMALIZE protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-NORMALIZE protocols l2vpn normalize vlan-id-any
```

Pattern 3: Multi-Site with Selective Connectivity

```
## Site can connect to multiple specific remote sites
set routing-instances L2VPN-MULTI protocols l2vpn site LOCAL site-identifier 1
set routing-instances L2VPN-MULTI protocols l2vpn site LOCAL interface ge-0/0/1.0 remote-site-id 2
set routing-instances L2VPN-MULTI protocols l2vpn site LOCAL interface ge-0/0/1.0 remote-site-id 3
set routing-instances L2VPN-MULTI protocols l2vpn site LOCAL interface ge-0/0/1.0 remote-site-id 4
```

Complete L2VPN Configuration Example

Here's a comprehensive configuration for PE1:

```
## System and Interfaces
set system host-name PE1
set interfaces lo0 unit 0 family inet address 1.1.1.1/32

## Core interface
set interfaces ge-0/0/0 description "To P-Router"
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces ge-0/0/0 unit 0 family mpls

## Customer attachment circuits
set interfaces ge-0/0/1 description "Customer-A Site-1 All-VLANs"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

set interfaces ge-0/0/2 description "Customer-B Site-1 VLAN-100"
set interfaces ge-0/0/2 encapsulation vlan-ccc
set interfaces ge-0/0/2 unit 100 vlan-id 100
set interfaces ge-0/0/2 unit 100 family ccc

## MPLS and LDP
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface lo0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0

## OSPF
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0

## BGP
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 1.1.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP neighbor 2.2.2.2
set protocols bgp group IBGP neighbor 3.3.3.3

## L2VPN Instance for Customer A
set routing-instances L2VPN-CUST-A instance-type l2vpn
set routing-instances L2VPN-CUST-A interface ge-0/0/1.0
set routing-instances L2VPN-CUST-A route-distinguisher 1.1.1.1:100
set routing-instances L2VPN-CUST-A vrf-target target:65000:100
set routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-CUST-A protocols l2vpn control-word
set routing-instances L2VPN-CUST-A protocols l2vpn mtu 1500
set routing-instances L2VPN-CUST-A protocols l2vpn site SITE1 site-identifier 1
set routing-instances L2VPN-CUST-A protocols l2vpn site SITE1 interface ge-0/0/1.0 remote-site-id 2

## L2VPN Instance for Customer B
set routing-instances L2VPN-CUST-B instance-type l2vpn
set routing-instances L2VPN-CUST-B interface ge-0/0/2.100
```

```
set routing-instances L2VPN-CUST-B route-distinguisher 1.1.1.1:200
set routing-instances L2VPN-CUST-B vrf-target target:65000:200
set routing-instances L2VPN-CUST-B protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-CUST-B protocols l2vpn no-control-word
set routing-instances L2VPN-CUST-B protocols l2vpn site SITE1 site-identifier 1
set routing-instances L2VPN-CUST-B protocols l2vpn site SITE1 interface ge-0/0/2.100 remote-site-id 2
```

Part 3: Verification & Troubleshooting (The What-If)

Essential L2VPN Verification Commands

1. Verify L2VPN Instance Status

```
user@PE1> show l2vpn connections
Layer-2 VPN connections:

Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby        SN -- Static Neighbor
LB -- Local site not best-site   RB -- Remote site not best-site
VM -- VLAN ID mismatch

Instance: L2VPN-CUST-A
  Local site: SITE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 26 12:00:01 2023  1
      Remote PE: 2.2.2.2, Negotiated control-word: Yes (Null)
      Incoming label: 800001, Outgoing label: 800002
      Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET
```

What to Look For:

- Status should be "Up"
- Control-word negotiation should match configuration
- Labels should be allocated (not 0)
- Encapsulation should match on both sides

2. Verify BGP L2VPN Routes

```
user@PE1> show route table bgp.l2vpn.0

bgp.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.1:100:1:800000:8/96
                   *[L2VPN/170] 00:05:23, metric2 1
                     Indirect
2.2.2.2:100:2:800008:8/96
```

```
                  *[BGP/170] 00:04:15, localpref 100, from 2.2.2.2
                     AS path: I, validation-state: unverified
                   > to 10.0.0.2 via ge-0/0/0.0, Push 299776

user@PE1> show route table bgp.l2vpn.0 detail 2.2.2.2:100:2

bgp.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
2.2.2.2:100:2:800008:8/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 2.2.2.2:100
                Next hop type: Indirect, Next hop index: 0
                Address: 0x9c8f3f0
                Next-hop reference count: 2
                Source: 2.2.2.2
                Protocol next hop: 2.2.2.2
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                Local AS: 65000 Peer AS: 65000
                Age: 4:23       Metric2: 1
                Validation State: unverified
                Task: BGP_65000.2.2.2.2
                Announcement bits (1): 0-L2VPN
                AS path: I
                Communities: target:65000:100 encapsulation:l2-3
                Import Accepted
                Layer2-info: encaps: ETHERNET, control-word: No, site-id: 2
                Label-block Offset: 8, Size: 8
                Localpref: 100
                Router ID: 2.2.2.2
                Primary Routing Table bgp.l2vpn.0
```

3. Verify Label Allocation

```
user@PE1> show l2vpn connections extensive
Layer-2 VPN connections:

Instance: L2VPN-CUST-A
  Local site: SITE1 (1)
    Number of local interfaces: 1
    Number of local interfaces up: 1
    interface-name: ge-0/0/1.0
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 26 12:00:01 2023  1
      Remote PE: 2.2.2.2, Negotiated control-word: Yes (Null)
      Incoming label: 800001, Outgoing label: 800009
      Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET
      Flow Label Transmit: No, Flow Label Receive: No
    Connection History:
        Oct 26 12:00:01 2023  status update timer
        Oct 26 12:00:01 2023  PE route changed
        Oct 26 12:00:01 2023  Out lbl Update                    800009
        Oct 26 12:00:01 2023  In lbl Update                     800001
        Oct 26 12:00:01 2023  loc intf up                ge-0/0/1.0
```

Common L2VPN Troubleshooting Scenarios

Scenario 1: Site ID Mismatch

Symptom: L2VPN shows as "Down" with "CN" (circuit not provisioned) status

Diagnostic Commands:

```
user@PE1> show l2vpn connections
Instance: L2VPN-CUST-A
  Local site: SITE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   CN     -                     0

user@PE1> show route table bgp.l2vpn.0 | match "L2VPN|site-id"
1.1.1.1:100:1:800000:8/96
                  *[L2VPN/170] 00:10:00, metric2 1
2.2.2.2:100:3:800008:8/96  ## Note: remote has site-id 3, not 2!
                  *[BGP/170] 00:05:00, localpref 100, from 2.2.2.2
```

Cause: Local configuration expects remote-site-id 2, but remote PE advertises site-id 3

Solution:

```
## Option 1: Fix remote PE configuration
user@PE2# set routing-instances L2VPN-CUST-A protocols l2vpn site SITE2 site-identifier 2

## Option 2: Fix local configuration
user@PE1# set routing-instances L2VPN-CUST-A protocols l2vpn site SITE1 interface ge-0/0/1.0 remote-site-id 3
```

Scenario 2: Encapsulation Mismatch

Symptom: L2VPN shows "EM" (encapsulation mismatch) status

Diagnostic Commands:

```
user@PE1> show l2vpn connections extensive | match "Encap|encap"
      Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET
                Layer2-info: encaps: ETHERNET-VLAN, control-word: No, site-id: 2

user@PE1> show configuration routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type
encapsulation-type ethernet;

user@PE2> show configuration routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type
encapsulation-type ethernet-vlan;  ## Mismatch!
```

Cause: PE1 uses ethernet encapsulation, PE2 uses ethernet-vlan

Solution:

```
## Make encapsulation consistent
user@PE2# delete routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type ethernet-vlan
user@PE2# set routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type ethernet
user@PE2# commit
```

Scenario 3: MTU Mismatch

Symptom: L2VPN up but large frames dropped, shows "MM" status

Diagnostic Commands:

```
user@PE1> show l2vpn connections | match "MM|MTU"
    2                         rmt   MM     Oct 26 12:00:01 2023  1

user@PE1> show interfaces ge-0/0/1 | match MTU
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps

user@PE1> show configuration routing-instances L2VPN-CUST-A protocols l2vpn mtu
mtu 1500;
```

```
## Check remote side
user@PE2> show configuration routing-instances L2VPN-CUST-A protocols l2vpn mtu
mtu 9000;  ## Mismatch!
```

Cause: MTU settings don't match between PEs

Solution:

```
## Option 1: Set consistent MTU in L2VPN
user@PE1# set routing-instances L2VPN-CUST-A protocols l2vpn mtu 9000

## Option 2: Let L2VPN discover MTU automatically
user@PE1# delete routing-instances L2VPN-CUST-A protocols l2vpn mtu
user@PE2# delete routing-instances L2VPN-CUST-A protocols l2vpn mtu
```

Scenario 4: Control Word Mismatch

Symptom: L2VPN shows "CM" (control-word mismatch) status

Diagnostic Commands:

```
user@PE1> show l2vpn connections
    2                       rmt   CM     -                    0
      Remote PE: 2.2.2.2, Negotiated control-word: No

user@PE1> show configuration routing-instances L2VPN-CUST-A protocols l2vpn | match control
control-word;

user@PE2> show configuration routing-instances L2VPN-CUST-A protocols l2vpn | match control
no-control-word;
```

Cause: Control word configuration mismatch

Solution:

```
## Make control word configuration consistent
user@PE2# delete routing-instances L2VPN-CUST-A protocols l2vpn no-control-word
user@PE2# set routing-instances L2VPN-CUST-A protocols l2vpn control-word
```

Performance Monitoring

Monitor L2VPN health and performance:

```
## Check interface statistics
user@PE1> monitor interface ge-0/0/1

## Monitor L2VPN-specific counters
user@PE1> show l2vpn connections statistics
Instance: L2VPN-CUST-A
  Local site: SITE1 (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 26 12:00:01 2023  1
      Local interface: ge-0/0/1.0
        Input bytes  :            123456789
        Output bytes :            987654321
        Input packets:               100000
        Output packets:              200000

## Check for L2VPN flapping
user@PE1> show l2vpn connections history
```

Advanced Troubleshooting

For complex issues, check BGP update messages:

```
## Enable BGP tracing
set protocols bgp group IBGP traceoptions file bgp-trace
set protocols bgp group IBGP traceoptions file size 10m
set protocols bgp group IBGP traceoptions flag update detail

## Monitor BGP updates
user@PE1> monitor start bgp-trace
user@PE1> clear bgp neighbor 2.2.2.2 soft

## Check trace file
user@PE1> show log bgp-trace | match L2VPN
```

Summary Checklist for L2VPN Verification

- [ ] BGP session up with `family l2vpn signaling`: `show bgp summary`
- [ ] L2VPN routes in BGP table: `show route table bgp.l2vpn.0`
- [ ] L2VPN connection status "Up": `show l2vpn connections`
- [ ] Correct label allocation: `show l2vpn connections extensive`
- [ ] No interface errors: `show interfaces extensive | match error`
- [ ] Matching encapsulation on both PEs
- [ ] Matching control word settings
- [ ] Consistent MTU configuration
- [ ] Correct site IDs configured

Module 04: L2VPN—Configuration

Part 1: The Conceptual Lecture (The Why)

Understanding Ethernet Transport Options

When you connect two locations with L2VPN, you need to decide what traffic to transport. It's like choosing between shipping an entire warehouse (all VLANs) or just specific boxes (specific VLANs).

All Ethernet Traffic vs. Specific VLANs

Consider these two scenarios:

Scenario 1: Data Center Interconnect

```
Data Center A                    MPLS Network                    Data Center B
┌──────────────┐                                                ┌──────────────┐
│ VLAN 10-999 │ ←──────── Transport ALL VLANs Transparently ────→ │ VLAN 10-999 │
└──────────────┘                                                └──────────────┘

Customer wants: Complete Layer 2 transparency
Solution: L2VPN accepting all Ethernet traffic
```

Scenario 2: Branch Office Extension

```
Main Office                      MPLS Network                    Branch Office
┌──────────────┐                                                ┌──────────────┐
│ VLAN 100     │ ←──────── Transport ONLY VLAN 100 ──────────────→ │ VLAN 100     │
│ (Finance)    │                                                │ (Finance)    │
└──────────────┘                                                └──────────────┘

Customer wants: Extend only Finance VLAN
Solution: L2VPN accepting specific VLAN tag
```

Port Mode vs. VLAN Mode

Think of these as two different service models:

Port Mode (All Ethernet Traffic):

- PE doesn't look at or modify VLAN tags
- Customer has full control over VLANs
- Like having a private fiber between sites

VLAN Mode (Specific VLAN):

- PE inspects and potentially modifies VLAN tags
- Service provider can offer per-VLAN services
- Like having a VLAN-specific connection

```
Port Mode Frame Processing:
Customer Frame → [VLAN 100][Data] → PE (doesn't inspect) → [MPLS][VLAN 100][Data]

VLAN Mode Frame Processing:
Customer Frame → [VLAN 100][Data] → PE (inspects VLAN) → [MPLS][Data]
                                          ↓
                                    May remove/add/swap VLAN
```

Encapsulation Types and Their Purposes

The encapsulation type tells the PE routers how to handle the frames:

```
Encapsulation Decision Tree:

Does customer need all VLANs transported?
├─ Yes → Use 'ethernet' encapsulation
│        └─ Interface: ethernet-ccc
└─ No → Specific VLAN(s)?
         ├─ Single VLAN → Use 'ethernet-vlan' encapsulation
         │                └─ Interface: vlan-ccc
         └─ Multiple specific VLANs → Use 'ethernet-vlan' + multiple units
                                      └─ Interface: extended-vlan-ccc
```

Service Delimiting: How PEs Identify Services

When multiple services share a physical interface, PEs need to identify which frames belong to which service:

```
Single Physical Interface, Multiple Services:

                          PE Router
Customer ─── ge-0/0/1 ──┤
                          │ Unit 0 (all traffic) → L2VPN-A
                          │ Unit 100 (VLAN 100) → L2VPN-B
                          │ Unit 200 (VLAN 200) → L2VPN-C
                          └──────────────────────────
```

The Importance of Proper Configuration

Incorrect configuration can lead to:

1. **Traffic Black-holing**: Frames enter but never exit
2. **VLAN Leaking**: Traffic appears in wrong VLANs
3. **Broadcast Storms**: Without proper loop prevention
4. **Service Disruption**: During maintenance windows

Part 2: The Junos CLI Masterclass (The How)

Configuration Pattern 1: L2VPN Accepting All Ethernet Traffic

This configuration transports all Ethernet frames transparently.

Step 1: Configure the Physical Interface

```
set interfaces ge-0/0/1 description "Customer ABC - All VLANs"
set interfaces ge-0/0/1 mtu 9192
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 description "All Ethernet traffic"
set interfaces ge-0/0/1 unit 0 family ccc
```

Why each command:

- `mtu 9192` : Accommodates jumbo frames plus MPLS overhead
- `encapsulation ethernet-ccc` : Treats entire port as circuit
- `unit 0` : Logical unit 0 captures all traffic on the interface
- `family ccc` : Circuit Cross-Connect family for L2 transport

Step 2: Configure the L2VPN Instance

```
## Create the routing instance
set routing-instances L2VPN-ALL-TRAFFIC instance-type l2vpn
set routing-instances L2VPN-ALL-TRAFFIC interface ge-0/0/1.0
set routing-instances L2VPN-ALL-TRAFFIC route-distinguisher 10.1.1.1:1000
set routing-instances L2VPN-ALL-TRAFFIC vrf-target target:65000:1000

## Configure L2VPN parameters
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn ignore-mtu-mismatch
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn control-word
```

Key points:

- `encapsulation-type ethernet` : Matches the interface encapsulation
- `ignore-mtu-mismatch` : Useful when sites have different MTUs
- `control-word` : Adds 4-byte header for better payload detection

Step 3: Configure Sites

```
## Local site
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn site SITE-A site-identifier 1
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn site SITE-A interface ge-0/0/1.0
set routing-instances L2VPN-ALL-TRAFFIC protocols l2vpn site SITE-A interface ge-0/0/1.0 remote-site-id 2
```

Configuration Pattern 2: L2VPN Accepting Specific VLAN Tags

This configuration transports only specific VLAN traffic.

Step 1: Configure the Physical Interface for VLAN Mode

```
set interfaces ge-0/0/2 description "Customer XYZ - Specific VLANs"
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 encapsulation extended-vlan-ccc
set interfaces ge-0/0/2 unit 100 description "Customer XYZ - VLAN 100"
set interfaces ge-0/0/2 unit 100 vlan-id 100
set interfaces ge-0/0/2 unit 100 family ccc
set interfaces ge-0/0/2 unit 200 description "Customer XYZ - VLAN 200"
set interfaces ge-0/0/2 unit 200 vlan-id 200
set interfaces ge-0/0/2 unit 200 family ccc
```

Why each command:

- `vlan-tagging` : Enables VLAN tag processing
- `encapsulation extended-vlan-ccc` : Supports multiple VLAN units
- `unit 100 vlan-id 100` : Creates logical interface for VLAN 100
- Multiple units: Each VLAN can have its own L2VPN

Step 2: Configure L2VPN for Specific VLAN

```
## L2VPN for VLAN 100
set routing-instances L2VPN-VLAN100 instance-type l2vpn
set routing-instances L2VPN-VLAN100 interface ge-0/0/2.100
set routing-instances L2VPN-VLAN100 route-distinguisher 10.1.1.1:2100
set routing-instances L2VPN-VLAN100 vrf-target target:65000:2100

## L2VPN protocol configuration
set routing-instances L2VPN-VLAN100 protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-VLAN100 protocols l2vpn no-control-word
set routing-instances L2VPN-VLAN100 protocols l2vpn site SITE-A site-identifier 1
set routing-instances L2VPN-VLAN100 protocols l2vpn site SITE-A interface ge-0/0/2.100 remote-site-id 2
```

Advanced Configuration Scenarios

Scenario 1: Mixed Mode on Same Physical Interface

```
## Physical interface supporting both port and VLAN mode
set interfaces ge-0/0/3 flexible-vlan-tagging
set interfaces ge-0/0/3 encapsulation flexible-ethernet-services

## Unit 0 for untagged traffic
set interfaces ge-0/0/3 unit 0 family ccc

## Specific VLAN units
set interfaces ge-0/0/3 unit 100 vlan-id 100
set interfaces ge-0/0/3 unit 100 family ccc
set interfaces ge-0/0/3 unit 200 vlan-id 200
set interfaces ge-0/0/3 unit 200 family ccc
```

Scenario 2: VLAN Range Configuration

```
## Configure a range of VLANs
set interfaces ge-0/0/4 description "Customer - VLAN Range 100-199"
set interfaces ge-0/0/4 vlan-tagging
set interfaces ge-0/0/4 encapsulation vlan-ccc
set interfaces ge-0/0/4 unit 100 vlan-id-range 100-199
set interfaces ge-0/0/4 unit 100 family ccc
```

Scenario 3: VLAN Translation in L2VPN

```
## PE1: Customer uses VLAN 100
set interfaces ge-0/0/5 unit 100 vlan-id 100
set routing-instances L2VPN-TRANSLATE protocols l2vpn encapsulation-type ethernet

## PE2: Same customer uses VLAN 200
set interfaces ge-0/0/5 unit 200 vlan-id 200
set routing-instances L2VPN-TRANSLATE protocols l2vpn encapsulation-type ethernet

## Result: VLAN 100 at Site A becomes VLAN 200 at Site B
```

Complete Configuration Examples

Example 1: Enterprise All-VLAN Transport

```
## PE1 Configuration
## Interface configuration
set interfaces ge-0/0/1 description "ACME Corp - Data Center 1"
set interfaces ge-0/0/1 mtu 9192
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## L2VPN instance
set routing-instances ACME-DC-INTERCONNECT instance-type l2vpn
set routing-instances ACME-DC-INTERCONNECT interface ge-0/0/1.0
set routing-instances ACME-DC-INTERCONNECT route-distinguisher 192.168.1.1:5000
set routing-instances ACME-DC-INTERCONNECT vrf-target target:65000:5000
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn encapsulation-type ethernet
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn mtu 9000
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn control-word
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn site DC1 site-identifier 1
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn site DC1 interface ge-0/0/1.0 remote-site-id 2

## Corresponding PE2 Configuration
set interfaces ge-0/0/1 description "ACME Corp - Data Center 2"
set interfaces ge-0/0/1 mtu 9192
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

set routing-instances ACME-DC-INTERCONNECT instance-type l2vpn
set routing-instances ACME-DC-INTERCONNECT interface ge-0/0/1.0
set routing-instances ACME-DC-INTERCONNECT route-distinguisher 192.168.1.2:5000
set routing-instances ACME-DC-INTERCONNECT vrf-target target:65000:5000
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn encapsulation-type ethernet
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn mtu 9000
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn control-word
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn site DC2 site-identifier 2
set routing-instances ACME-DC-INTERCONNECT protocols l2vpn site DC2 interface ge-0/0/1.0 remote-site-id 1
```

Example 2: Service Provider Multi-Tenant VLAN Services

```
## PE1 Configuration for multiple customers on same port
## Physical interface
set interfaces ge-0/0/10 description "Multi-tenant Access Port"
set interfaces ge-0/0/10 vlan-tagging
set interfaces ge-0/0/10 encapsulation extended-vlan-ccc

## Customer A - VLAN 100
set interfaces ge-0/0/10 unit 100 description "Customer A - Production"
set interfaces ge-0/0/10 unit 100 vlan-id 100
set interfaces ge-0/0/10 unit 100 family ccc

## Customer B - VLAN 200
set interfaces ge-0/0/10 unit 200 description "Customer B - Office Network"
set interfaces ge-0/0/10 unit 200 vlan-id 200
set interfaces ge-0/0/10 unit 200 family ccc

## Customer C - VLAN 300-399
set interfaces ge-0/0/10 unit 300 description "Customer C - Multiple VLANs"
set interfaces ge-0/0/10 unit 300 vlan-id-range 300-399
set interfaces ge-0/0/10 unit 300 family ccc

## L2VPN for Customer A
set routing-instances L2VPN-CUST-A instance-type l2vpn
set routing-instances L2VPN-CUST-A interface ge-0/0/10.100
set routing-instances L2VPN-CUST-A route-distinguisher 10.0.0.1:100
```

```
set routing-instances L2VPN-CUST-A vrf-target target:65000:1100
set routing-instances L2VPN-CUST-A protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-CUST-A protocols l2vpn site HQ site-identifier 1
set routing-instances L2VPN-CUST-A protocols l2vpn site HQ interface ge-0/0/10.100 remote-site-id 2

## L2VPN for Customer B
set routing-instances L2VPN-CUST-B instance-type l2vpn
set routing-instances L2VPN-CUST-B interface ge-0/0/10.200
set routing-instances L2VPN-CUST-B route-distinguisher 10.0.0.1:200
set routing-instances L2VPN-CUST-B vrf-target target:65000:1200
set routing-instances L2VPN-CUST-B protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-CUST-B protocols l2vpn site MAIN site-identifier 1
set routing-instances L2VPN-CUST-B protocols l2vpn site MAIN interface ge-0/0/10.200 remote-site-id 2

## L2VPN for Customer C
set routing-instances L2VPN-CUST-C instance-type l2vpn
set routing-instances L2VPN-CUST-C interface ge-0/0/10.300
set routing-instances L2VPN-CUST-C route-distinguisher 10.0.0.1:300
set routing-instances L2VPN-CUST-C vrf-target target:65000:1300
set routing-instances L2VPN-CUST-C protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-CUST-C protocols l2vpn site CAMPUS site-identifier 1
set routing-instances L2VPN-CUST-C protocols l2vpn site CAMPUS interface ge-0/0/10.300 remote-site-id 2
```

Part 3: Verification & Troubleshooting (The What-If)

Verification Commands for Different L2VPN Types

1. Verify All-Ethernet L2VPN

```
user@PE1> show l2vpn connections interface ge-0/0/1.0
Layer-2 VPN connections:

Instance: L2VPN-ALL-TRAFFIC
  Local site: SITE-A (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 27 09:00:00 2023  1
      Remote PE: 10.0.0.2, Negotiated control-word: Yes (Null)
      Incoming label: 100000, Outgoing label: 200000
      Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET

user@PE1> show interfaces ge-0/0/1.0 detail | match "ccc|CCC"
  Logical interface ge-0/0/1.0 (Index 333) (SNMP ifIndex 530)
    Flags: Up SNMP-Trapping 0x4000000 Encapsulation: Ethernet-CCC
```

2. Verify VLAN-Specific L2VPN

```
user@PE1> show l2vpn connections interface ge-0/0/2.100
Layer-2 VPN connections:

Instance: L2VPN-VLAN100
  Local site: SITE-A (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 27 09:30:00 2023  1
      Remote PE: 10.0.0.2, Negotiated control-word: No
      Incoming label: 100100, Outgoing label: 200100
      Local interface: ge-0/0/2.100, Status: Up, Encapsulation: ETHERNET-VLAN

user@PE1> show interfaces ge-0/0/2.100
  Logical interface ge-0/0/2.100 (Index 334) (SNMP ifIndex 531)
    Flags: Up SNMP-Trapping 0x4000000 VLAN-Tag [ 0x8100.100 ] Encapsulation: VLAN-CCC
```

```
    Input packets : 1234567
    Output packets: 7654321
```

Common Configuration Issues and Solutions

Scenario 1: Wrong Interface Encapsulation

Symptom: L2VPN won't come up, shows "EI" (encapsulation invalid)

Diagnostic Commands:

```
user@PE1> show l2vpn connections | match "EI|ge-"
    2                       rmt   EI     -                     0
      Local interface: ge-0/0/1.0, Status: Up, Encapsulation: ETHERNET

user@PE1> show configuration interfaces ge-0/0/1
vlan-tagging;
encapsulation vlan-ccc;  ## Wrong for all-Ethernet mode!
unit 0 {
    family ccc;
}
```

Cause: Interface configured for VLAN mode but L2VPN expects Ethernet mode

Solution:

```
delete interfaces ge-0/0/1 vlan-tagging
delete interfaces ge-0/0/1 encapsulation vlan-ccc
set interfaces ge-0/0/1 encapsulation ethernet-ccc
commit
```

Scenario 2: VLAN ID Not Matching Traffic

Symptom: No traffic passing through VLAN-specific L2VPN

Diagnostic Commands:

```
user@PE1> monitor interface ge-0/0/2.100
## No packets incrementing

user@PE1> show interfaces ge-0/0/2 extensive | match "VLAN.*statistics"
  VLAN tag statistics:
    VLAN 200: 1000000 packets  ## Traffic on VLAN 200, not 100!

user@PE1> show configuration interfaces ge-0/0/2 unit 100
vlan-id 100;  ## Configured for VLAN 100
```

Cause: Customer sending VLAN 200 traffic but interface expects VLAN 100

Solution:

```
## Option 1: Fix customer tagging
## Ask customer to send VLAN 100

## Option 2: Change interface configuration
delete interfaces ge-0/0/2 unit 100 vlan-id 100
set interfaces ge-0/0/2 unit 100 vlan-id 200
commit
```

Scenario 3: Flexible Encapsulation Misconfiguration

Symptom: Some VLANs work, others don't on same interface

Diagnostic Commands:

```
user@PE1> show interfaces ge-0/0/3 | match encap
  Type: Ethernet, Link-level type: Flexible-Ethernet, MTU: 1522,

user@PE1> show configuration interfaces ge-0/0/3
flexible-vlan-tagging;
encapsulation ethernet-ccc;  ## Wrong! Should be flexible-ethernet-services
```

Cause: Wrong encapsulation for flexible VLAN tagging

Solution:

```
delete interfaces ge-0/0/3 encapsulation ethernet-ccc
set interfaces ge-0/0/3 encapsulation flexible-ethernet-services
commit
```

Scenario 4: Unit 0 Not Capturing Untagged Traffic

Symptom: Untagged traffic not forwarded in mixed mode

Diagnostic Commands:

```
user@PE1> show ethernet-switching table interface ge-0/0/4.0
## No MAC addresses learned

user@PE1> show configuration interfaces ge-0/0/4 unit 0
vlan-id 1;  ## Wrong! Unit 0 shouldn't have vlan-id for untagged
family ccc;
```

Cause: Unit 0 configured with VLAN ID

Solution:

```
delete interfaces ge-0/0/4 unit 0 vlan-id
commit
```

Performance and Monitoring

Monitor L2VPN Traffic Patterns

```
## Real-time interface monitoring
user@PE1> monitor interface ge-0/0/1

## Check for errors
user@PE1> show interfaces ge-0/0/1 extensive | match "error|drop|pause"
     Bit errors                            0
     Errored blocks                        0
     FEC Corrected Errors                  0
     FEC Uncorrected Errors                0
     Input errors                          0
     Input drops                           0
     Input pause frames                    0
     Output errors                         0
     Output drops                          0
     Output pause frames                   0

## L2VPN-specific statistics
user@PE1> show l2vpn connections statistics
Instance: L2VPN-ALL-TRAFFIC
  Local interface: ge-0/0/1.0, Index: 333
    Input bytes  :             12345678900
```

```
    Output bytes :             98765432100
    Input packets:                10000000
    Output packets:               20000000
```

VLAN Statistics Monitoring

```
## Monitor VLAN-specific counters
user@PE1> show interfaces ge-0/0/2 extensive | find "VLAN tag"
  Logical interface ge-0/0/2.100 (Index 334) (SNMP ifIndex 531)
    Flags: Up VLAN-Tag [ 0x8100.100 ] Encapsulation: VLAN-CCC
    Statistics        Packets        pps         Bytes          bps
    Input :         1000000          0    1000000000            0
    Output:         2000000          0    2000000000            0
```

Configuration Best Practices

1. **MTU Considerations**:

```
## Always configure interface MTU larger than payload
set interfaces ge-0/0/1 mtu 9192  ## Allows 9000-byte customer frames + MPLS
```

2. **Description Standards**:

```
set interfaces ge-0/0/1 description "CustID:12345 | Circuit:NYC-LON-001 | L2VPN-ALL"
set interfaces ge-0/0/1 unit 0 description "All-VLANs | Contact: netops@customer.com"
```

3. **Encapsulation Matching**:

```
Interface Encapsulation    →    L2VPN Encapsulation Type
ethernet-ccc              →    ethernet
vlan-ccc                  →    ethernet-vlan
extended-vlan-ccc         →    ethernet-vlan (multiple units)
flexible-ethernet-services →    ethernet or ethernet-vlan
```

4. **VLAN Range Planning**:

```
## Reserve VLAN ranges per customer
Customer A: VLAN 100-199
Customer B: VLAN 200-299
Customer C: VLAN 300-399
## Document in interface descriptions
```

Summary Configuration Checklist

For All-Ethernet L2VPN:

- [ ] Interface encapsulation: `ethernet-ccc`
- [ ] Unit 0 with `family ccc`
- [ ] L2VPN encapsulation-type: `ethernet`
- [ ] MTU accommodates largest customer frame

For VLAN-Specific L2VPN:

- [ ] Interface encapsulation: `vlan-ccc` or `extended-vlan-ccc`
- [ ] Correct `vlan-id` on unit
- [ ] L2VPN encapsulation-type: `ethernet-vlan`
- [ ] VLAN tag matches customer traffic

For Both Types:

- [ ] Route distinguisher unique per PE/instance

- [ ] Route target matches between PEs
- [ ] Site IDs correctly configured
- [ ] BGP sessions established with L2VPN family

Module 05: L2VPN—Troubleshooting

Part 1: The Conceptual Lecture (The Why)

The Three Pillars of L2VPN Operation

For an L2VPN to work correctly, three critical components must align perfectly. Think of it like a three-legged stool—if any leg is broken, the whole thing collapses.

```
The Three Pillars of L2VPN:

    1. Transport (MPLS LSP)          2. Control Plane (BGP)          3. Configuration
         │                                │                               │
         ├─ LSP to remote PE              ├─ BGP session up               ├─ Matching encapsulation
         ├─ Label distribution            ├─ L2VPN NLRI exchange          ├─ Correct site IDs
         └─ MTU consistency               └─ Route target import          └─ Interface settings

                                All three must work together!
```

Understanding Missing LSPs

An LSP (Label Switched Path) is the highway your L2VPN traffic travels on. Without it, you have the perfect car (L2VPN) but no road to drive on.

```
Missing LSP Scenario:

PE1 ----X---- No LSP ----X---- PE2
 │                             │
 │    L2VPN configured         │    L2VPN configured
 │    BGP routes received      │    BGP routes received
 │    But no forwarding path!  │    But no forwarding path!
 │                             │
CE1                           CE2

Result: L2VPN shows "OL" (no Outgoing Label) or "LD" (Label Distribution problem)
```

Common LSP Problems

1. **No IGP Route to Remote PE**:

```
PE1's View:
Destination: PE2 (2.2.2.2)
IGP Route: Not found
Result: Cannot build LSP
```

2. **LDP Not Enabled on Path**:

```
PE1 --[LDP]-- P1 --[No LDP]-- P2 --[LDP]-- PE2
                      ↑
                Break in label distribution
```

3. **RSVP LSP Not Configured**:

```
Expected: LSP from PE1 to PE2
Reality: No LSP configured in PE1
```

```
Result: No transport path
```

Site Information Mismatches

Site information acts like a postal address within the L2VPN. Mismatches are like trying to send mail to the wrong address.

```
Site ID Mismatch Example:

PE1 Configuration:                    PE2 Configuration:
Site 1 wants remote-site-id 2        Site 3 advertises site-id 3
         ↓                                      ↓
    "Send my traffic                    "I am site 3"
     to site 2"

               No one is advertising site 2!
               Connection fails with "CN" status
```

BGP Route Import/Export Issues

Even with correct site IDs, route targets control whether PEs accept each other's advertisements:

```
Route Target Mismatch:

PE1 Exports:                          PE2 Import Filter:
RT: target:65000:100                  RT: target:65000:200
        ↓                                     ↓
   "Tagged as 100"                     "Only accept 200"

                   PE2 rejects PE1's routes!
```

Configuration Inconsistencies

Small configuration differences can break L2VPN:

```
Common Configuration Mismatches:

1. Encapsulation Type:
   PE1: ethernet        ←→  PE2: ethernet-vlan     = EM (Encap Mismatch)

2. Control Word:
   PE1: control-word    ←→  PE2: no-control-word   = CM (Control-word Mismatch)

3. MTU:
   PE1: mtu 1500       ←→  PE2: mtu 9000          = MM (MTU Mismatch)

4. Interface Type:
   PE1: ethernet-ccc   ←→  PE2: vlan-ccc          = WE (Wrong Encapsulation)
```

Part 2: The Junos CLI Masterclass (The How)

Systematic Troubleshooting Approach

Follow this methodology to diagnose L2VPN issues efficiently:

```
L2VPN Troubleshooting Flowchart:

1. Check L2VPN Status
   └─> "Up"? → Check traffic flow
   └─> Status code? → Decode the problem

2. Verify Transport (MPLS)
   └─> LSP established?
```

```
   └─> Labels distributed?

3. Verify Control Plane (BGP)
   └─> BGP session up?
   └─> Routes exchanged?
   └─> Routes imported?

4. Verify Configuration
   └─> Encapsulation match?
   └─> Site IDs correct?
   └─> Interface settings?
```

Step 1: Decode L2VPN Status

```
user@PE1> show l2vpn connections
```

Understanding status codes:

```
Critical Status Codes:
Up  - Operational
OL  - No Outgoing Label (MPLS problem)
LD  - Local site Down (interface problem)
CN  - Circuit Not provisioned (site ID mismatch)
EM  - Encapsulation Mismatch
CM  - Control-word Mismatch
MM  - MTU Mismatch
NC  - Interface Not CCC/TCC
WE  - Wrong Encapsulation type
OR  - Out of Range (site ID > site-range)
```

Step 2: Troubleshoot Missing LSPs

Check IGP Reachability

```
## Verify route to remote PE
user@PE1> show route 2.2.2.2

inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2.2.2.2/32         *[OSPF/10] 00:10:00, metric 20
                    > to 10.0.0.2 via ge-0/0/0.0

## If no route exists, check OSPF
user@PE1> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.2         ge-0/0/0.0             Full      192.168.1.1      128    39

## Check if remote PE's loopback is in OSPF database
user@PE1> show ospf database router lsa-id 2.2.2.2 detail
```

Verify MPLS LSP Establishment

For LDP-signaled LSPs:

```
## Check LDP neighbors
user@PE1> show ldp neighbor
Address                 Interface       Label space ID     Hold time
10.0.0.2                ge-0/0/0.0      192.168.1.1:0      13

## Check LDP database for remote PE FEC
```

```
user@PE1> show ldp database | match "2.2.2.2/32"
  2.2.2.2/32                        100048

## Verify label forwarding
user@PE1> show route forwarding-table destination 2.2.2.2
Destination        Type RtRef Next hop           Type Index    NhRef Netif
2.2.2.2/32         user     0                    Push 100048      583     2 ge-0/0/0.0
```

For RSVP-signaled LSPs:

```
## Check RSVP LSP status
user@PE1> show mpls lsp name TO-PE2
Ingress LSP: 1 sessions
To              From            State Rt P     ActivePath       LSPname
2.2.2.2         1.1.1.1         Up     0 *                      TO-PE2

## If LSP is down, check why
user@PE1> show mpls lsp name TO-PE2 extensive | match "reason|CSPF"
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 20)
    3 Oct 28 10:00:00.000 CSPF failed: no route to destination[4 times]

## Check RSVP neighbors
user@PE1> show rsvp neighbor
```

Fix Missing LSP Issues

```
## Fix 1: Enable protocols on interfaces
set protocols mpls interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0

## Fix 2: Create RSVP LSP if needed
set protocols mpls label-switched-path TO-PE2 to 2.2.2.2

## Fix 3: Fix OSPF if loopback not advertised
set protocols ospf area 0.0.0.0 interface lo0.0 passive
```

Step 3: Troubleshoot Site Information Mismatches

Diagnose Site ID Issues

```
## Check local site configuration
user@PE1> show configuration routing-instances L2VPN-A protocols l2vpn site
site SITE1 {
    site-identifier 1;
    interface ge-0/0/1.0 {
        remote-site-id 2;  ## Looking for site 2
    }
}

## Check what remote PE is advertising
user@PE1> show route table bgp.l2vpn.0 detail | match "site-id|RD"
  Route Distinguisher: 1.1.1.1:100
                Layer2-info: encaps: ETHERNET, control-word: No, site-id: 1
  Route Distinguisher: 2.2.2.2:100
                Layer2-info: encaps: ETHERNET, control-word: No, site-id: 3
                ## Remote is advertising site 3, not site 2!

## Alternative: Check L2VPN connections with detail
user@PE1> show l2vpn connections instance L2VPN-A extensive | match "site|Site"
  Local site: SITE1 (1)
```

```
    Remote PE: 2.2.2.2, Negotiated control-word: No
    Remote site: 3  ## Mismatch!
```

Fix Site ID Mismatches

```
## Option 1: Change local remote-site-id
set routing-instances L2VPN-A protocols l2vpn site SITE1 interface ge-0/0/1.0 remote-site-id 3

## Option 2: Fix remote PE site ID (on PE2)
set routing-instances L2VPN-A protocols l2vpn site SITE2 site-identifier 2
```

Step 4: Troubleshoot BGP and Route Target Issues

Verify BGP L2VPN Signaling

```
## Check BGP session
user@PE1> show bgp summary | match "2.2.2.2|Family"
  2.2.2.2               65000         100        200       0       0     1:00:00 Establ
  bgp.l2vpn.0: 4/4/4/0

## Verify L2VPN family is negotiated
user@PE1> show bgp neighbor 2.2.2.2 | match "NLRI|l2vpn"
  NLRI that we support:
    inet-unicast inet-multicast l2vpn
  NLRI peer supports:
    inet-unicast inet-multicast l2vpn
```

Check Route Import/Export

```
## See what we're advertising
user@PE1> show route advertising-protocol bgp 2.2.2.2 table bgp.l2vpn.0 detail
bgp.l2vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1:100:1:800000:8/96 (1 entry, 1 announced)
 BGP group IBGP type Internal
     Route Distinguisher: 1.1.1.1:100
     Route Label: 800000
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 100
     AS path: [65000] I
     Communities: target:65000:100 encapsulation:l2-3

## Check what we're receiving
user@PE1> show route receive-protocol bgp 2.2.2.2 table bgp.l2vpn.0 hidden
bgp.l2vpn.0: 2 destinations, 2 routes (1 active, 0 holddown, 1 hidden)
  2.2.2.2:100:2:800008:8/96 (hidden)
     Import: [ L2VPN-A-import ]
     Communities: target:65000:200  ## Wrong RT! We expect 65000:100
```

Fix Route Target Issues

```
## Check import policy
user@PE1> show configuration policy-options policy-statement L2VPN-A-import
term 1 {
    from {
        protocol bgp;
        community L2VPN-A;
    }
    then accept;
}
```

```
user@PE1> show configuration policy-options community L2VPN-A
members target:65000:100;  ## This is correct

## If using vrf-target (simpler approach)
set routing-instances L2VPN-A vrf-target target:65000:100
```

Step 5: Troubleshoot Configuration Issues

Check Encapsulation Consistency

```
## Local encapsulation
user@PE1> show interfaces ge-0/0/1 | match encap
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514, Encapsulation: Ethernet-CCC

user@PE1> show configuration routing-instances L2VPN-A protocols l2vpn encapsulation-type
encapsulation-type ethernet;

## What remote is advertising
user@PE1> show route table bgp.l2vpn.0 detail 2.2.2.2:100 | match encap
                Communities: target:65000:100 encapsulation:l2-4
                ## l2-3 = ethernet, l2-4 = vlan, l2-5 = ethernet-vpls
```

Common Configuration Fixes

```
## Fix 1: Interface encapsulation
delete interfaces ge-0/0/1 encapsulation vlan-ccc
set interfaces ge-0/0/1 encapsulation ethernet-ccc

## Fix 2: L2VPN encapsulation type
delete routing-instances L2VPN-A protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-A protocols l2vpn encapsulation-type ethernet

## Fix 3: Control word mismatch
set routing-instances L2VPN-A protocols l2vpn control-word
## or
set routing-instances L2VPN-A protocols l2vpn no-control-word

## Fix 4: MTU issues
set routing-instances L2VPN-A protocols l2vpn ignore-mtu-mismatch
## or set consistent MTU
set routing-instances L2VPN-A protocols l2vpn mtu 9000
```

Advanced Troubleshooting Tools

Enable L2VPN Traceoptions

```
set routing-instances L2VPN-A protocols l2vpn traceoptions file l2vpn-debug
set routing-instances L2VPN-A protocols l2vpn traceoptions file size 10m
set routing-instances L2VPN-A protocols l2vpn traceoptions flag all

## View the trace
user@PE1> show log l2vpn-debug | last 50
```

Monitor L2VPN State Changes

```
## Real-time monitoring
user@PE1> monitor start l2vpn-trace
user@PE1> clear l2vpn connection
```

```
## Watch BGP updates
user@PE1> monitor route receive-protocol bgp 2.2.2.2 table bgp.l2vpn.0
```

Part 3: Verification & Troubleshooting (The What-If)

Scenario 1: Complete L2VPN Failure - No LSP

Symptom: L2VPN shows "OL" (no Outgoing Label)

Diagnostic Process:

```
user@PE1> show l2vpn connections
Instance: L2VPN-CUSTOMER
  Local site: HQ (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   OL     -                     0
      Remote PE: 2.2.2.2, Negotiated control-word: No
      Incoming label: 800001, Outgoing label: --  ## No label!

## Check route to remote PE
user@PE1> show route 2.2.2.2 table inet.3

inet.3: 0 destinations, 0 routes (0 active, 0 holddown, 0 hidden)
## No route in inet.3 (MPLS table)!

## Check LDP status
user@PE1> show ldp session
  Neighbor                        State       Hold time  Adv. Mode
## No LDP sessions!

## Check interface MPLS configuration
user@PE1> show mpls interface
## No interfaces listed!
```

Root Cause: MPLS/LDP not properly configured

Solution:

```
## Enable MPLS and LDP on core interfaces
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface lo0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0
commit

## Verify fix
user@PE1> show ldp neighbor
Address            Interface          Label space ID         Hold time
10.0.0.2          ge-0/0/0.0         192.168.1.1:0         14

user@PE1> show route 2.2.2.2 table inet.3
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2.2.2.2/32         *[LDP/9] 00:00:30, metric 1
                    > to 10.0.0.2 via ge-0/0/0.0, Push 299792
```

Scenario 2: Intermittent L2VPN Flapping

Symptom: L2VPN connection flaps between Up and Down

Diagnostic Process:

```
user@PE1> show l2vpn connections history
Instance: L2VPN-PROD
  Local site: MAIN (1)
    connection-site: 2
    Connection History:
        Oct 28 14:00:00  status update timer           Up
        Oct 28 13:55:00  PE route changed              Down
        Oct 28 13:54:30  status update timer           Up
        Oct 28 13:49:30  PE route changed              Down
        ## Pattern: Flapping every ~5 minutes

## Check BGP stability
user@PE1> show bgp neighbor 2.2.2.2 | match flap
  Last flap event: RecvNotify
  Flap: 15, Last: 00:05:00

## Check for routing loops
user@PE1> traceroute mpls ldp 2.2.2.2
  1  10.0.0.2 (10.0.0.2)  0.543 ms  0.423 ms  0.401 ms
     MPLS Label=299792 CoS=0 TTL=1 S=1
  2  10.0.0.6 (10.0.0.6)  0.875 ms  0.823 ms  0.798 ms
     MPLS Label=299776 CoS=0 TTL=1 S=1
  3  10.0.0.2 (10.0.0.2)  1.234 ms  1.123 ms  1.087 ms
     MPLS Label=299792 CoS=0 TTL=1 S=1
  ## Loop detected!
```

Root Cause: IGP metric configuration causing unstable paths

Solution:

```
## Fix OSPF metrics to ensure stable path
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 metric 10
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0 metric 100
commit

## Consider using RSVP-TE for explicit paths
set protocols mpls label-switched-path TO-PE2 to 2.2.2.2
set protocols mpls label-switched-path TO-PE2 primary DIRECT
set protocols mpls path DIRECT 10.0.0.2 strict
```

Scenario 3: Partial Traffic Loss

Symptom: Some traffic passes but with high loss rate

Diagnostic Process:

```
## Test from CE devices shows packet loss
CE1# ping 192.168.1.2 -c 100
100 packets transmitted, 67 received, 33% packet loss

## Check for interface errors on PE
user@PE1> show interfaces ge-0/0/1 extensive | match "error|drop"
     Input errors: 1520
     Link-level type: Ethernet, MTU: 1514
     Policed discards: 1520  ## Packets being dropped!

## Check L2VPN policer
user@PE1> show configuration class-of-service interfaces ge-0/0/1
unit 0 {
    forwarding-class assured-forwarding {
        policer L2VPN-POLICER;  ## Policer applied!
    }
```

```
}

## Check policer configuration
user@PE1> show configuration firewall policer L2VPN-POLICER
if-exceeding {
    bandwidth-limit 10m;  ## Only 10 Mbps allowed!
    burst-size-limit 15k;
}
then discard;
```

Root Cause: Policer limiting bandwidth

Solution:

```
## Option 1: Increase policer limits
set firewall policer L2VPN-POLICER if-exceeding bandwidth-limit 100m
set firewall policer L2VPN-POLICER if-exceeding burst-size-limit 150k

## Option 2: Remove policer
delete class-of-service interfaces ge-0/0/1 unit 0 forwarding-class assured-forwarding policer
```

Scenario 4: VLAN Translation Not Working

Symptom: L2VPN up but customer reports no connectivity

Diagnostic Process:

```
## L2VPN shows up
user@PE1> show l2vpn connections
    2                        rmt   Up     Oct 28 15:00:00 2023  1

## But no traffic passing
user@PE1> monitor interface ge-0/0/1.100
## No packets incrementing

## Capture traffic on physical interface
user@PE1> monitor traffic interface ge-0/0/1 no-resolve size 1500
15:00:00.123456 In
  Ethernet II, Src: 00:11:22:33:44:55, Dst: 00:aa:bb:cc:dd:ee
  802.1Q VLAN=200, p=0  ## Customer sending VLAN 200!

## Check our configuration
user@PE1> show configuration interfaces ge-0/0/1 unit 100
vlan-id 100;  ## We expect VLAN 100!
```

Root Cause: Customer sending different VLAN than configured

Solution:

```
## Option 1: VLAN normalization
set routing-instances L2VPN-A protocols l2vpn encapsulation-type ethernet
## This strips VLAN tags

## Option 2: Configure correct VLAN
delete interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 vlan-id 200

## Option 3: Accept any VLAN
set interfaces ge-0/0/1 unit 100 vlan-id-range 1-4094
```

Performance Monitoring Dashboard

Create a monitoring checklist:

```
## 1. L2VPN Status Overview
user@PE1> show l2vpn connections summary
Total connections: 10
Up: 8
Down: 2 (CN: 1, OL: 1)

## 2. Interface Health
user@PE1> show interfaces descriptions | match "L2VPN"
ge-0/0/1.0      up    up   L2VPN-CUSTOMER-A
ge-0/0/2.100    up    up   L2VPN-CUSTOMER-B-VLAN100

## 3. Error Rates
user@PE1> show interfaces media | match "ge-0/0/[12]" -A5 | match "error|CRC"

## 4. L2VPN Traffic Statistics
user@PE1> show l2vpn connections statistics | match "ge-|bytes"

## 5. BGP L2VPN Route Count
user@PE1> show route summary table bgp.l2vpn.0
bgp.l2vpn.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
```

Troubleshooting Command Quick Reference

```
## Status Checks
show l2vpn connections [instance <name>] [summary|extensive|history]
show bgp summary
show route table bgp.l2vpn.0 [hidden]
show mpls lsp
show ldp session

## Detailed Diagnostics
show route receive-protocol bgp <peer> table bgp.l2vpn.0 detail
show route advertising-protocol bgp <peer> table bgp.l2vpn.0 detail
show interfaces <interface> extensive
monitor traffic interface <interface>

## Path Verification
traceroute mpls ldp <remote-pe>
ping mpls ldp <remote-pe>
show route forwarding-table destination <remote-pe>

## Configuration Verification
show configuration routing-instances <name> | display inheritance
show configuration interfaces <interface> | display inheritance
```

Summary: L2VPN Troubleshooting Best Practices

1. **Always Start with Status**: Check `show l2vpn connections` first
2. **Follow the Path**: Verify MPLS → BGP → Configuration in order
3. **Check Both Ends**: Many issues require checking both PEs
4. **Use History**: Connection history reveals patterns
5. **Monitor Changes**: Enable traceoptions before making changes
6. **Document Findings**: Keep notes of what fixed each issue

Common fixes by status code:

- **OL**: Enable MPLS/LDP, create LSP
- **CN**: Fix site ID mismatch
- **EM**: Match encapsulation types
- **CM**: Align control-word settings
- **MM**: Set consistent MTU or ignore
- **NC**: Configure interface as CCC family

Module 06: L2VPN—Site IDs, the Label Base, and Overprovisioning

Part 1: The Conceptual Lecture (The Why)

Understanding the Label Block Architecture

In BGP-signaled L2VPNs, Junos uses an elegant label allocation scheme that's both efficient and scalable. Instead of signaling individual labels for each connection, it advertises a block of labels.

```
Traditional Approach (Inefficient):
Site 1 → Site 2: "Use label 100001"
Site 1 → Site 3: "Use label 100002"
Site 1 → Site 4: "Use label 100003"
Result: N signals for N connections

L2VPN Label Block (Efficient):
Site 1: "I'm starting at label 800000, block size 8"
Result: 1 signal for up to 8 connections

Label Calculation:
To reach Site 1 from Site X: Use Label_Base + (X − 1)
From Site 2: 800000 + (2−1) = 800001
From Site 3: 800000 + (3−1) = 800002
From Site 4: 800000 + (4−1) = 800003
```

The Site ID: Your Address in the VPN

A Site ID is like a house number on a street. Each site in an L2VPN must have a unique identifier, and these IDs are used to calculate which label to use.

```
Site ID Assignment Example:

Company ABC L2VPN:
┌──────────────────────────────────────┐
│ New York HQ      − Site ID: 1        │
│ London Branch    − Site ID: 2        │
│ Tokyo Branch     − Site ID: 3        │
│ Sydney Branch    − Site ID: 4        │
└──────────────────────────────────────┘

Label Usage:
When Tokyo (Site 3) sends to New York (Site 1):
Uses label: 800000 + (3−1) = 800002
```

The Problem of Scale: Why Overprovisioning?

Imagine you're building a corporate L2VPN that will grow over time. You start with 4 sites but plan to add 20 more. Do you reconfigure every PE router each time you add a site? This is where overprovisioning shines.

```
Without Overprovisioning:
Every new site requires:
1. Configure new site on local PE
2. Update ALL remote PEs with new remote-site-id
3. Risk of outage during changes
4. O(n²) configuration complexity

With Overprovisioning:
1. Pre-allocate site IDs 1-50
2. Configure once with ranges
```

```
3. New sites just claim an unused ID
4. No changes needed on existing sites
```

Label Base and Block Size

The label base is the starting point for label allocation, and the block size determines how many sites can connect:

```
Label Block Allocation:

Label Base: 800000
Block Size: 8
Valid Labels: 800000-800007

Site ID to Label Mapping:
Site 1 → Label 800000
Site 2 → Label 800001
Site 3 → Label 800002
...
Site 8 → Label 800007
Site 9 → ERROR! (exceeds block size)
```

Explicit vs Implicit Remote Site IDs

Explicit Configuration: You specify exactly which remote sites to connect to

```
"I am Site 1, connect me to Site 2, Site 3, and Site 4"
```

Implicit Configuration: You specify a range, connecting to any site in that range

```
"I am Site 1, connect me to any site from 2 to 50"
```

Hub-and-Spoke Topology Considerations

Overprovisioning is particularly powerful for hub-and-spoke topologies:

```
Hub-and-Spoke with Overprovisioning:

Hub (Site 1)
├── Configured once: remote-site-id 2-100
│
├── Spoke Sites (Sites 2-100)
│   └── Each configured: remote-site-id 1
│
└── Result: New spokes added without touching hub
```

Part 2: The Junos CLI Masterclass (The How)

Understanding Label Allocation

First, let's see how Junos allocates labels:

```
user@PE1> show l2vpn connections extensive
Layer-2 VPN connections:

Instance: L2VPN-CORP
  Local site: HQ (site-id 1)
    Label-base: 800000, Offset: 0, Size: 8  ## 8 labels allocated

    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 29 10:00:00 2023  1
```

```
      Incoming label: 800001, Outgoing label: 800009
      ## Incoming: 800000 + (2-1) = 800001
```

Basic Overprovisioning Configuration

Step 1: Determine Your Scaling Requirements

```
Planning Worksheet:
- Current sites: 4
- Planned growth: 20 sites in 2 years
- Safety margin: 2x
- Total site IDs needed: 50
```

Step 2: Configure Site Range

```
## Set the site range (must be ≥ highest site ID)
set routing-instances L2VPN-CORP protocols l2vpn site-range 50

## This allocates labels for sites 1-50
## Default block size = site-range value
```

Pattern 1: Explicit Remote Site IDs

Use explicit configuration when you want precise control over connections:

```
## PE1 Configuration (New York - Site 1)
set routing-instances L2VPN-CORP instance-type l2vpn
set routing-instances L2VPN-CORP interface ge-0/0/1.0
set routing-instances L2VPN-CORP route-distinguisher 1.1.1.1:100
set routing-instances L2VPN-CORP vrf-target target:65000:100

## Configure L2VPN parameters
set routing-instances L2VPN-CORP protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-CORP protocols l2vpn site-range 50
set routing-instances L2VPN-CORP protocols l2vpn mtu 9000

## Local site with explicit remote sites
set routing-instances L2VPN-CORP protocols l2vpn site NY site-identifier 1
set routing-instances L2VPN-CORP protocols l2vpn site NY interface ge-0/0/1.0

## Explicitly specify each remote site
set routing-instances L2VPN-CORP protocols l2vpn site NY interface ge-0/0/1.0 remote-site-id 2
set routing-instances L2VPN-CORP protocols l2vpn site NY interface ge-0/0/1.0 remote-site-id 3
set routing-instances L2VPN-CORP protocols l2vpn site NY interface ge-0/0/1.0 remote-site-id 4
set routing-instances L2VPN-CORP protocols l2vpn site NY interface ge-0/0/1.0 remote-site-id 5
```

Pattern 2: Implicit Remote Site IDs (Ranges)

Use implicit configuration for dynamic topologies:

```
## PE1 Configuration - Hub Site
set routing-instances L2VPN-CORP instance-type l2vpn
set routing-instances L2VPN-CORP interface ge-0/0/1.0
set routing-instances L2VPN-CORP route-distinguisher 1.1.1.1:100
set routing-instances L2VPN-CORP vrf-target target:65000:100

## Configure L2VPN with overprovisioning
set routing-instances L2VPN-CORP protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-CORP protocols l2vpn site-range 50

## Hub site connects to all spokes (2-50)
```

```
set routing-instances L2VPN-CORP protocols l2vpn site HUB site-identifier 1
set routing-instances L2VPN-CORP protocols l2vpn site HUB interface ge-0/0/1.0

## Use range instead of individual remote-site-ids
set routing-instances L2VPN-CORP protocols l2vpn site HUB interface ge-0/0/1.0 remote-site-id 2-50
```

Pattern 3: Mixed Explicit and Implicit

Sometimes you need both approaches:

```
## Regional Hub Configuration
## Connects to main hub (1) and regional spokes (10-19)

set routing-instances L2VPN-REGION instance-type l2vpn
set routing-instances L2VPN-REGION interface ge-0/0/1.0
set routing-instances L2VPN-REGION route-distinguisher 2.2.2.2:100
set routing-instances L2VPN-REGION vrf-target target:65000:100

set routing-instances L2VPN-REGION protocols l2vpn site-range 50
set routing-instances L2VPN-REGION protocols l2vpn site REGIONAL-HUB site-identifier 10
set routing-instances L2VPN-REGION protocols l2vpn site REGIONAL-HUB interface ge-0/0/1.0

## Explicit connection to main hub
set routing-instances L2VPN-REGION protocols l2vpn site REGIONAL-HUB interface ge-0/0/1.0 remote-site-id 1

## Implicit connections to regional spokes
set routing-instances L2VPN-REGION protocols l2vpn site REGIONAL-HUB interface ge-0/0/1.0 remote-site-id 11-19
```

Advanced Overprovisioning Techniques

Custom Label Base and Block Size

```
## Manual label allocation control
set routing-instances L2VPN-CUSTOM protocols l2vpn site SITE1 site-identifier 1
set routing-instances L2VPN-CUSTOM protocols l2vpn site SITE1 manual

## Specify custom label block
set routing-instances L2VPN-CUSTOM protocols l2vpn site SITE1 label-base 900000
set routing-instances L2VPN-CUSTOM protocols l2vpn site SITE1 label-block-size 100
set routing-instances L2VPN-CUSTOM protocols l2vpn site SITE1 label-offset 0
```

Multi-Site on Single PE

```
## PE hosting multiple sites
## Site 1 - Main Office
set routing-instances L2VPN-MULTI protocols l2vpn site MAIN site-identifier 1
set routing-instances L2VPN-MULTI protocols l2vpn site MAIN interface ge-0/0/1.0
set routing-instances L2VPN-MULTI protocols l2vpn site MAIN interface ge-0/0/1.0 remote-site-id 2-50

## Site 2 - DMZ Network (on same PE!)
set routing-instances L2VPN-MULTI protocols l2vpn site DMZ site-identifier 2
set routing-instances L2VPN-MULTI protocols l2vpn site DMZ interface ge-0/0/2.0
set routing-instances L2VPN-MULTI protocols l2vpn site DMZ interface ge-0/0/2.0 remote-site-id 1
set routing-instances L2VPN-MULTI protocols l2vpn site DMZ interface ge-0/0/2.0 remote-site-id 3-50
```

Complete Hub-and-Spoke Configuration Example

Hub PE (PE1) Configuration

```
## System
set system host-name PE1-HUB
```

```
## Interfaces
set interfaces lo0 unit 0 family inet address 1.1.1.1/32
set interfaces ge-0/0/0 description "To Core"
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces ge-0/0/0 unit 0 family mpls

set interfaces ge-0/0/1 description "HUB-SITE LAN"
set interfaces ge-0/0/1 mtu 9192
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## Protocols
set protocols mpls interface all
set protocols ldp interface all

set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 1.1.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP neighbor 2.2.2.2
set protocols bgp group IBGP neighbor 3.3.3.3
set protocols bgp group IBGP neighbor 4.4.4.4

## L2VPN Hub Configuration with Overprovisioning
set routing-instances HUB-SPOKE-VPN instance-type l2vpn
set routing-instances HUB-SPOKE-VPN interface ge-0/0/1.0
set routing-instances HUB-SPOKE-VPN route-distinguisher 1.1.1.1:1000
set routing-instances HUB-SPOKE-VPN vrf-target target:65000:1000

## Overprovision for 100 sites
set routing-instances HUB-SPOKE-VPN protocols l2vpn site-range 100
set routing-instances HUB-SPOKE-VPN protocols l2vpn encapsulation-type ethernet
set routing-instances HUB-SPOKE-VPN protocols l2vpn control-word

## Hub connects to all spokes (2-100)
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB site-identifier 1
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB interface ge-0/0/1.0
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB interface ge-0/0/1.0 remote-site-id 2-100
```

Spoke PE (PE2) Configuration

```
## System
set system host-name PE2-SPOKE

## Interfaces (similar to hub)
set interfaces lo0 unit 0 family inet address 2.2.2.2/32
set interfaces ge-0/0/1 description "SPOKE-2 LAN"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## L2VPN Spoke Configuration
set routing-instances HUB-SPOKE-VPN instance-type l2vpn
set routing-instances HUB-SPOKE-VPN interface ge-0/0/1.0
set routing-instances HUB-SPOKE-VPN route-distinguisher 2.2.2.2:1000
set routing-instances HUB-SPOKE-VPN vrf-target target:65000:1000

## Must match hub's site-range
set routing-instances HUB-SPOKE-VPN protocols l2vpn site-range 100
set routing-instances HUB-SPOKE-VPN protocols l2vpn encapsulation-type ethernet
set routing-instances HUB-SPOKE-VPN protocols l2vpn control-word

## Spoke only connects to hub
```

```
set routing-instances HUB-SPOKE-VPN protocols l2vpn site SPOKE2 site-identifier 2
set routing-instances HUB-SPOKE-VPN protocols l2vpn site SPOKE2 interface ge-0/0/1.0
set routing-instances HUB-SPOKE-VPN protocols l2vpn site SPOKE2 interface ge-0/0/1.0 remote-site-id 1
```

Part 3: Verification & Troubleshooting (The What-If)

Verifying Label Block Allocation

```
user@PE1> show l2vpn connections extensive | match "Label-base|block"
    Label-base: 800000, Offset: 0, Size: 100  ## Allocated for 100 sites

user@PE1> show route table bgp.l2vpn.0 detail 1.1.1.1:1000
1.1.1.1:1000:1:800000:100/96 (1 entry, 1 announced)
        *L2VPN  Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Label-block Offset: 0, Size: 100  ## Advertised in BGP
                Label-base: 800000
```

Verifying Overprovisioned Connections

```
## Check all active connections
user@PE1> show l2vpn connections instance HUB-SPOKE-VPN
Instance: HUB-SPOKE-VPN
  Local site: HUB (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 29 12:00:00 2023  1
    3                         rmt   Up     Oct 29 12:00:00 2023  1
    4                         rmt   Up     Oct 29 12:00:00 2023  1
    5                         rmt   RN     -                     0  ## Site 5 not active yet
    6-100                     rmt   --     -                     0  ## Dormant connections
```

Common Overprovisioning Issues

Scenario 1: Site ID Exceeds Site Range

Symptom: New site can't connect, shows "OR" (Out of Range)

Diagnostic Commands:

```
user@PE-NEW> show l2vpn connections
Instance: HUB-SPOKE-VPN
  Local site: SPOKE50 (50)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   OR     -                     0

user@PE-NEW> show configuration routing-instances HUB-SPOKE-VPN protocols l2vpn
site-range 45;  ## Too small! Site 50 > 45
site SPOKE50 {
    site-identifier 50;  ## Out of range!
}
```

Solution:

```
## Increase site-range on ALL PEs
set routing-instances HUB-SPOKE-VPN protocols l2vpn site-range 100
commit
```

Scenario 2: Label Block Exhaustion

Symptom: Can't allocate labels for all sites

Diagnostic Commands:

```
user@PE1> show log messages | match "label.*exhaust"
Oct 29 13:00:00 PE1 l2vpn[1234]: L2VPN_LABEL_BLOCK_EXHAUSTED: Instance HUB-SPOKE-VPN,
site HUB: Cannot allocate label block size 200 (requested by configuration)

user@PE1> show l2vpn connections extensive | match "Size|Label-base"
    Label-base: 800000, Offset: 0, Size: 100  ## Platform limit reached
```

Solution:

```
## Option 1: Use multiple L2VPN instances
## Split sites across instances

## Option 2: Use manual label allocation with different bases
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB manual
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB label-base 900000
set routing-instances HUB-SPOKE-VPN protocols l2vpn site HUB label-block-size 50
```

Scenario 3: Implicit Range Overlap

Symptom: Unexpected connections forming

Diagnostic Commands:

```
user@PE1> show l2vpn connections instance REGIONAL
Instance: REGIONAL
  Local site: REGION1 (10)
    connection-site           Type  St     Time last up          # Up trans
    1                         rmt   Up     Oct 29 14:00:00 2023  1
    11-20                     rmt   Up     Oct 29 14:00:00 2023  1
    15                        rmt   Up     Oct 29 14:00:00 2023  1  ## Duplicate?

## Site 15 appears twice - once in range, once from another PE
```

Analysis:

```
user@PE1> show route table bgp.l2vpn.0 | match ":15:"
3.3.3.3:1000:15:850000:100/96  ## Another PE advertising site 15
4.4.4.4:1000:15:860000:100/96  ## Two PEs claiming site 15!
```

Solution:

```
## Ensure unique site IDs across all PEs
## Document site ID allocation:
## 1-9:   Hub sites
## 10-19: Region 1
## 20-29: Region 2
## etc.
```

Scenario 4: Performance with Large Site Ranges

Symptom: Slow convergence with many overprovisioned sites

Diagnostic Commands:

```
user@PE1> show l2vpn connections summary
Total L2VPN connections: 500
Up: 50, Down: 5, Dormant: 445

user@PE1> show system processes extensive | match l2vpn
 1234 root     85  0   512M  256M CPU1  2:34:56 45.2% l2vpn
```

```
## High CPU usage from L2VPN process
```

Optimization:

```
## Use hierarchical design
## Instead of one flat L2VPN with 500 sites:

## Level 1: Regional Hubs (site-range 20)
set routing-instances REGIONAL-L2VPN protocols l2vpn site-range 20

## Level 2: Access layer (site-range 50 per region)
set routing-instances ACCESS-L2VPN protocols l2vpn site-range 50

## This reduces the connection mesh complexity
```

Monitoring Overprovisioned L2VPNs

Create monitoring for dynamic environments:

```
## Script to check for new sites
user@PE1> show l2vpn connections | match "Up|RN" | count
Count: 25 lines  ## 25 active or ready sites

## Monitor label usage
user@PE1> op l2vpn-label-usage
Instance         Site    Label-Base  Used  Available
HUB-SPOKE-VPN   HUB     800000      45    55

## Check for site ID conflicts
user@PE1> show route table bgp.l2vpn.0 | match ":[0-9]+:" |
          awk -F: '{print $3}' | sort | uniq -d
15  ## Site 15 is duplicated!
```

Advanced Verification Commands

```
## See all dormant connections (ready but not active)
user@PE1> show l2vpn connections extensive | match "dormant" -B2 -A2

## Verify label calculations
user@PE1> show l2vpn connections extensive instance HUB-SPOKE-VPN
  Local site: HUB (1)
    Label-base: 800000, Offset: 0, Size: 100
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 29 12:00:00 2023  1
      Incoming label: 800001  ## 800000 + (2-1) = 800001 ✓
    3                         rmt   Up     Oct 29 12:00:00 2023  1
      Incoming label: 800002  ## 800000 + (3-1) = 800002 ✓

## Check BGP advertisements for overprovisioned sites
user@PE1> show route advertising-protocol bgp 2.2.2.2 table bgp.l2vpn.0 detail
```

Best Practices for Overprovisioning

1. **Plan Site ID Allocation**:

```
Site ID Allocation Plan:
1-9:     Core/Hub sites
10-49:   Region 1
50-99:   Region 2
```

```
100-149: Region 3
150-199: Future expansion
```

2. **Use Consistent Site-Range**:

```
## All PEs must have same site-range
set routing-instances L2VPN protocols l2vpn site-range 200
```

3. **Monitor Resource Usage**:

```
## Check platform limits
user@PE1> show l2vpn resource-usage
Maximum L2VPN instances: 1000
Maximum sites per instance: 200
Maximum connections per site: 200
Current usage: 45/200 sites
```

4. **Document Implicit Ranges**:

```
## Add descriptions
set routing-instances L2VPN protocols l2vpn site HUB
    interface ge-0/0/1.0 description "Connects to spokes 2-100"
```

5. **Test Before Production**:

```
## Verify label allocation
user@PE1> test l2vpn label-allocation instance HUB-SPOKE-VPN site HUB
Label allocation test:
  Site 1 → Site 2: Label 800001
  Site 1 → Site 50: Label 800049
  Site 1 → Site 100: Label 800099
  All labels within allocated block ✓
```

Summary

Overprovisioning transforms L2VPN from a static, high-maintenance technology to a dynamic, scalable solution. Key takeaways:

- **Label blocks** eliminate per-connection signaling
- **Site IDs** act as addresses within the VPN
- **Site-range** must accommodate all possible sites
- **Implicit ranges** enable true dynamic provisioning
- **Proper planning** prevents ID conflicts and scaling issues

With overprovisioning, adding a new branch office becomes as simple as:

1. Assign an unused site ID
2. Configure the new PE with that ID
3. Connections automatically establish—no touching existing sites!

Module 07: L2VPN—Advanced Concepts

Part 1: The Conceptual Lecture (The Why)

Understanding the Need for Advanced L2VPN Features

Imagine you're building a private network for a large retail chain with hundreds of stores. Each store needs to connect to the headquarters as if they were all on the same local network, despite being geographically separated. This is where Layer 2 VPNs (L2VPNs) come in - they create virtual "ethernet cables" across a service provider's network.

However, basic L2VPNs have limitations:

- What if your headquarters' connection fails?
- What if different sites use different VLAN numbering schemes?
- How do you scale route distribution efficiently?

This is where advanced L2VPN concepts become crucial.

L2VPN Multihoming

The Problem: Single points of failure. If a customer edge (CE) device has only one connection to the provider edge (PE), any failure means complete isolation.

The Solution: Multihoming allows a CE to connect to multiple PEs simultaneously, providing redundancy.

How It Works:

1. The CE establishes connections to multiple PEs
2. Only one PE actively forwards traffic (designated forwarder)
3. Other PEs remain in standby mode
4. If the active PE fails, a standby PE takes over

Martini Encapsulation

The Context: When Luca Martini first proposed L2VPN drafts, he defined specific encapsulation methods for carrying Layer 2 frames over MPLS.

What It Does: Martini encapsulation defines how to:

- Preserve the original Layer 2 header information
- Add control words for proper frame sequencing
- Handle different MTU sizes

```
Original Frame:
[Ethernet Header][Payload]

After Martini Encapsulation:
[MPLS Label][Control Word][Ethernet Header][Payload]
```

VLAN Normalization

The Problem: Different sites might use different VLAN IDs for the same logical network. Site A uses VLAN 100 for accounting, but Site B already uses VLAN 100 for HR.

The Solution: VLAN normalization rewrites VLAN tags as frames traverse the L2VPN.

```
Site A (VLAN 100) --> PE1 --> [Normalize to VLAN 500] --> MPLS Core
                                                                |
Site B (VLAN 200) <-- PE2 <-- [Normalize from VLAN 500] <----+
```

Out-of-Band Route Reflection and Route Target Constraint

The Scaling Challenge: In large networks, every PE doesn't need to know about every L2VPN. This wastes memory and processing power.

Route Reflection: Instead of full-mesh iBGP sessions, PEs connect to route reflectors:

```
        +-----RR-----+
       /      |      \
     PE1     PE2     PE3
```

Route Target Constraint (RTC): PEs only request routes for VPNs they actually serve:

- PE1 says "I only want routes with RT 65000:100"
- Route Reflector filters and sends only relevant routes
- Reduces unnecessary route distribution

Part 2: The Junos CLI Masterclass (The How)

Configuration Hierarchy

L2VPN advanced features are configured under several hierarchies:

```
[edit routing-instances <instance-name> protocols l2vpn]
[edit protocols bgp group <group-name>]
[edit policy-options]
```

Configuring L2VPN Multihoming

Let's configure a multihomed L2VPN where CE1 connects to both PE1 and PE2:

```
## On PE1:
set interfaces ge-0/0/0 description "To CE1"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

set routing-instances L2VPN-RETAIL instance-type l2vpn
set routing-instances L2VPN-RETAIL interface ge-0/0/0.0
set routing-instances L2VPN-RETAIL route-distinguisher 10.1.1.1:100
set routing-instances L2VPN-RETAIL vrf-target target:65000:100
set routing-instances L2VPN-RETAIL protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 site-identifier 1
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 interface ge-0/0/0.0
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 site-preference primary

## On PE2 (backup):
set interfaces ge-0/0/0 description "To CE1 - Backup"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

set routing-instances L2VPN-RETAIL instance-type l2vpn
set routing-instances L2VPN-RETAIL interface ge-0/0/0.0
set routing-instances L2VPN-RETAIL route-distinguisher 10.1.1.2:100
set routing-instances L2VPN-RETAIL vrf-target target:65000:100
set routing-instances L2VPN-RETAIL protocols l2vpn encapsulation-type ethernet
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 site-identifier 1
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 interface ge-0/0/0.0
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 site-preference backup
```

Key Configuration Elements:

- `site-identifier`: Must be the same on all PEs serving the same CE
- `site-preference`: Determines active (primary) vs standby (backup) role
- Both PEs use the same `vrf-target` to exchange routes

Configuring VLAN Normalization

Transform incoming VLAN 100 to VLAN 500 within the L2VPN:

```
set interfaces ge-0/0/1 description "Customer Site A - VLAN 100"
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 encapsulation flexible-ethernet-services
set interfaces ge-0/0/1 unit 100 encapsulation vlan-ccc
set interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 input-vlan-map pop
set interfaces ge-0/0/1 unit 100 output-vlan-map push vlan-id 100

set routing-instances L2VPN-NORMALIZED instance-type l2vpn
set routing-instances L2VPN-NORMALIZED interface ge-0/0/1.100
set routing-instances L2VPN-NORMALIZED route-distinguisher 10.1.1.1:200
set routing-instances L2VPN-NORMALIZED vrf-target target:65000:200
set routing-instances L2VPN-NORMALIZED protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances L2VPN-NORMALIZED protocols l2vpn site SITE-A site-identifier 1
set routing-instances L2VPN-NORMALIZED protocols l2vpn site SITE-A interface ge-0/0/1.100
set routing-instances L2VPN-NORMALIZED protocols l2vpn site SITE-A remote-site-id 2
set routing-instances L2VPN-NORMALIZED protocols l2vpn normalize vlan-id 500
```

Configuring Route Target Constraint

Enable RTC to optimize route distribution:

```
## Enable RTC on BGP sessions:
set protocols bgp group IBGP-RR type internal
set protocols bgp group IBGP-RR local-address 10.1.1.1
set protocols bgp group IBGP-RR family l2vpn signaling
set protocols bgp group IBGP-RR family route-target advertise-default
set protocols bgp group IBGP-RR neighbor 10.255.255.1 description "Route Reflector"

## On Route Reflector:
set protocols bgp group PE-CLIENTS type internal
set protocols bgp group PE-CLIENTS local-address 10.255.255.1
set protocols bgp group PE-CLIENTS family l2vpn signaling
set protocols bgp group PE-CLIENTS family route-target
set protocols bgp group PE-CLIENTS cluster 10.255.255.1
set protocols bgp group PE-CLIENTS neighbor 10.1.1.1 description "PE1"
set protocols bgp group PE-CLIENTS neighbor 10.1.1.2 description "PE2"
```

Complete Reference Configuration

Here's a complete L2VPN configuration with all advanced features:

```
## Physical Interface Configuration
set interfaces ge-0/0/0 description "To CE1 - Multihomed"
set interfaces ge-0/0/0 flexible-vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 100 encapsulation vlan-ccc
set interfaces ge-0/0/0 unit 100 vlan-id 100

## BGP Configuration with RTC
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 10.1.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP family route-target
set protocols bgp group IBGP neighbor 10.255.255.1

## L2VPN Instance with Advanced Features
set routing-instances ADVANCED-L2VPN instance-type l2vpn
set routing-instances ADVANCED-L2VPN interface ge-0/0/0.100
set routing-instances ADVANCED-L2VPN route-distinguisher 10.1.1.1:300
set routing-instances ADVANCED-L2VPN vrf-target target:65000:300
set routing-instances ADVANCED-L2VPN protocols l2vpn encapsulation-type ethernet-vlan
set routing-instances ADVANCED-L2VPN protocols l2vpn control-word
set routing-instances ADVANCED-L2VPN protocols l2vpn site HQ site-identifier 1
set routing-instances ADVANCED-L2VPN protocols l2vpn site HQ site-preference primary
```

```
set routing-instances ADVANCED-L2VPN protocols l2vpn site HQ interface ge-0/0/0.100
set routing-instances ADVANCED-L2VPN protocols l2vpn normalize vlan-id 999
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

```
## Verify L2VPN status:
show l2vpn connections instance ADVANCED-L2VPN extensive

## Check multihoming status:
show l2vpn connections site-preference

## Verify VLAN normalization:
show interfaces ge-0/0/0.100 extensive | match "VLAN"

## Check BGP RTC operation:
show bgp summary family route-target
show route table bgp.rtarget.0
```

Troubleshooting Scenario 1: Multihoming Not Working

Symptom: Both PEs show as active, causing loops

Diagnosis:

```
admin@PE1> show l2vpn connections instance L2VPN-RETAIL
Instance: L2VPN-RETAIL
  Local site: CE1 (1)
    Status: Up -- Site is designated forwarder  ## Both PEs show this!
```

Cause: Mismatched site-identifier between PEs

Solution:

```
## Ensure both PEs use the same site-identifier:
set routing-instances L2VPN-RETAIL protocols l2vpn site CE1 site-identifier 1
```

Troubleshooting Scenario 2: VLAN Normalization Failure

Symptom: Remote site receives frames with wrong VLAN ID

Diagnosis:

```
admin@PE1> show l2vpn connections instance L2VPN-NORMALIZED extensive
  Normalization: Disabled  ## Should show enabled!
```

Cause: Missing or incorrect normalize configuration

Solution:

```
set routing-instances L2VPN-NORMALIZED protocols l2vpn normalize vlan-id 500
set routing-instances L2VPN-NORMALIZED protocols l2vpn encapsulation-type ethernet-vlan
```

Troubleshooting Scenario 3: RTC Not Filtering Routes

Symptom: PE receives all L2VPN routes despite RTC configuration

Diagnosis:

```
admin@PE1> show route receive-protocol bgp 10.255.255.1 table bgp.l2vpn.0 | count
Count: 5000 lines  ## Too many routes!

admin@PE1> show bgp neighbor 10.255.255.1 | match "route-target"
  NLRI that we support: route-target  ## Missing "advertised"
```

Cause: RTC not properly negotiated with route reflector

Solution:

```
## Add advertise-default on PE:
set protocols bgp group IBGP family route-target advertise-default

## Ensure route reflector has family route-target configured
```

Troubleshooting Scenario 4: Control Word Mismatch

Symptom: L2VPN shows "CM" (Control-word Mismatch) status

Diagnosis:

```
admin@PE1> show l2vpn connections
...
  Remote PE: 10.1.1.2, Status: CM  ## Control-word Mismatch
```

Cause: One PE has control-word enabled, the other doesn't

Solution:

```
## Enable on both PEs:
set routing-instances ADVANCED-L2VPN protocols l2vpn control-word

## Or disable on both:
delete routing-instances ADVANCED-L2VPN protocols l2vpn control-word
```

Module 08: L2Circuit—LDP-Signaled Pseudowires

Part 1: The Conceptual Lecture (The Why)

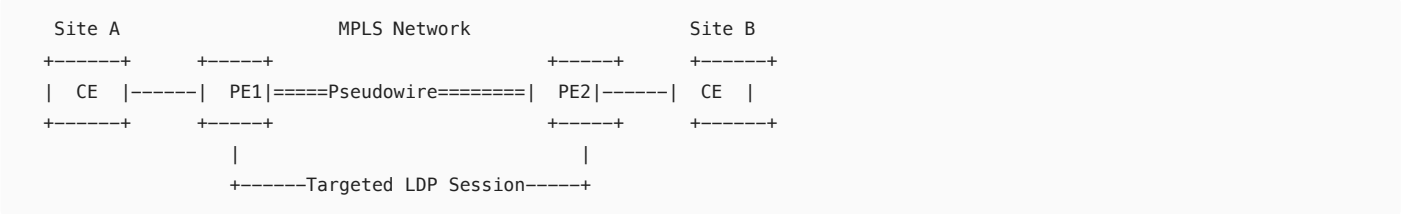
Understanding L2Circuit Fundamentals

Think of traditional telecommunications where you'd lease a physical circuit between two locations - a dedicated copper wire that connected point A to point B. L2Circuit is the modern equivalent, creating a virtual circuit over an MPLS network using LDP (Label Distribution Protocol) for signaling.

Why L2Circuit Instead of L2VPN?

- **Simpler**: No BGP required, just LDP
- **Point-to-point**: Perfect for simple two-site connections
- **Lower overhead**: Less control plane complexity
- **Faster convergence**: LDP typically converges faster than BGP

The Architecture of L2Circuit



Key Components:

1. **Attachment Circuit (AC)**: The physical interface connecting CE to PE
2. **Pseudowire**: The virtual circuit across MPLS
3. **Targeted LDP Session**: Direct LDP peering between PEs (not following physical topology)

Targeted LDP Sessions Explained

Normal LDP sessions form between directly connected neighbors. But in L2Circuit, PE1 and PE2 might be many hops apart. This is where "targeted" LDP comes in:

Regular LDP:

```
PE1 ---- P1 ---- P2 ---- PE2
 |        |       |        |
LDP     LDP     LDP      LDP  (Following physical links)
```

Targeted LDP:

```
PE1 ---- P1 ---- P2 ---- PE2
 |                          |
 +-----Targeted LDP---------+  (Direct logical connection)
```

The targeted session allows PEs to exchange pseudowire labels directly, regardless of the physical path between them.

L2Circuit Signaling Process

1. **Virtual Circuit ID (VC-ID)**: Both PEs must agree on a unique identifier for the circuit
2. **Label Exchange**: Each PE allocates a label for incoming traffic
3. **Status Signaling**: PEs exchange circuit status (up/down/standby)

```
PE1 to PE2: "I'll use label 100045 for VC-ID 1000"
PE2 to PE1: "I'll use label 200067 for VC-ID 1000"

Result: Bidirectional label-switched path established
```

Common Elements with L2VPN

Despite different signaling methods, L2Circuit shares concepts with BGP-signaled L2VPN:

Feature	L2Circuit	L2VPN
Creates pseudowires	Yes	Yes
Encapsulation types	Ethernet, VLAN, etc.	Same
MTU considerations	Yes	Yes
Status signaling	Via LDP	Via BGP
Multipoint capable	No	Yes (with additions)

Part 2: The Junos CLI Masterclass (The How)

L2Circuit Configuration Structure

L2Circuit configuration lives under the protocols hierarchy:

```
[edit protocols l2circuit]
[edit interfaces]
```

Step-by-Step L2Circuit Configuration

Let's build an L2Circuit between two sites:

```
## Step 1: Configure the attachment circuit interface on PE1
set interfaces ge-0/0/0 description "L2Circuit to CustomerA-SiteA"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

## Step 2: Configure the L2Circuit on PE1
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 virtual-circuit-id 1000
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 description "CustomerA-Circuit"
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 control-word
```

```
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 encapsulation-type ethernet

## Step 3: Enable targeted LDP
set protocols ldp interface lo0.0
set protocols ldp targeted-hello
```

Configuration on PE2 (mirror configuration):

```
set interfaces ge-0/0/0 description "L2Circuit to CustomerA-SiteB"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 virtual-circuit-id 1000
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 description "CustomerA-Circuit"
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 control-word
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 encapsulation-type ethernet

set protocols ldp interface lo0.0
set protocols ldp targeted-hello
```

Understanding Configuration Parameters

Key Parameters Explained:

- `neighbor` : The loopback IP of the remote PE
- `virtual-circuit-id` : Must match on both PEs (like a circuit ID in traditional telco)
- `control-word` : Adds sequencing and padding information
- `encapsulation-type` : Must match the interface encapsulation

Advanced L2Circuit Features

Configuring VLAN-based L2Circuit:

```
## For VLAN-aware L2Circuit
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 encapsulation flexible-ethernet-services
set interfaces ge-0/0/1 unit 100 encapsulation vlan-ccc
set interfaces ge-0/0/1 unit 100 vlan-id 100

set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/1.100 virtual-circuit-id 2000
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/1.100 encapsulation-type ethernet-vlan
```

Configuring MTU:

```
## Set MTU to handle larger frames
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 mtu 9000
```

Bandwidth allocation:

```
## Reserve bandwidth for the L2Circuit
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 bandwidth 100m
```

Complete L2Circuit Configuration Example

Here's a production-ready L2Circuit configuration:

```
## PE1 Configuration

## Interface configuration
set interfaces ge-0/0/0 description "Customer XYZ - Site A"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

## LDP configuration
set protocols ldp interface lo0.0
set protocols ldp targeted-hello
```

```
set protocols ldp targeted-hello hold-time 45
set protocols ldp targeted-hello interval 15

## L2Circuit configuration
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 virtual-circuit-id 5000
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 description "XYZ-Circuit-Primary"
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 control-word
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 encapsulation-type ethernet
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 mtu 1600
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 pseudowire-status-tlv

## MPLS label allocation
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface lo0.0
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

```
## Verify L2Circuit status
show l2circuit connections

## Detailed L2Circuit information
show l2circuit connections extensive

## Check targeted LDP sessions
show ldp neighbor

## Verify LDP-signaled labels
show ldp database l2circuit

## Check interface status
show interfaces ge-0/0/0.0 extensive
```

Sample Output Analysis

Good L2Circuit connection:

```
admin@PE1> show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  HS -- Hot-standby Connection
XX -- unknown

Legend for interface status
Up -- operational
Dn -- down

Interface                Type  St     Time last up          # Up trans
ge-0/0/0.0(vc 5000)      rmt   Up     Oct 30 10:15:22 2024          1
  Remote PE: 10.2.2.2, Negotiated control-word: Yes (Null)
  Incoming label: 100045, Outgoing label: 200067
  Negotiated PW status TLV: Yes
  Local interface: ge-0/0/0.0, Status: Up, Encapsulation: ETHERNET
```

Troubleshooting Scenario 1: L2Circuit Shows "OL" Status

Symptom: No outgoing label

```
admin@PE1> show l2circuit connections
Interface                 Type  St     Time last up          # Up trans
ge-0/0/0.0(vc 5000)       rmt   OL
```

Diagnosis:

```
## Check if we can reach remote PE
admin@PE1> show route 10.2.2.2

## Check LDP neighbor status
admin@PE1> show ldp neighbor 10.2.2.2
  ## No output - no targeted LDP session!
```

Cause: No targeted LDP session to remote PE

Solution:

```
## Enable targeted LDP
set protocols ldp targeted-hello
set protocols ldp interface lo0.0

## Verify MPLS is enabled on loopback
set protocols mpls interface lo0.0
```

Troubleshooting Scenario 2: Encapsulation Mismatch

Symptom: L2Circuit shows "EM" status

```
admin@PE1> show l2circuit connections
Interface                 Type  St     Time last up          # Up trans
ge-0/0/0.0(vc 5000)       rmt   EM
```

Diagnosis:

```
admin@PE1> show l2circuit connections extensive
  Local interface: ge-0/0/0.0, Status: Up, Encapsulation: ETHERNET
  Remote PE: 10.2.2.2, Negotiated control-word: Yes (Null)
  Encapsulation mismatch detected (local: ETHERNET, remote: VLAN)
```

Cause: PE1 configured for ethernet, PE2 for ethernet-vlan

Solution:

```
## Make both sides match
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 encapsulation-type ethernet
## OR
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 encapsulation-type ethernet-vlan
```

Troubleshooting Scenario 3: MTU Mismatch

Symptom: L2Circuit shows "MM" status, large frames dropped

```
admin@PE1> show l2circuit connections
Interface                 Type  St     Time last up          # Up trans
ge-0/0/0.0(vc 5000)       rmt   MM
```

Diagnosis:

```
admin@PE1> show l2circuit connections extensive | match mtu
  Local interface: ge-0/0/0.0, Status: Up, Encapsulation: ETHERNET, MTU: 1514
  Negotiated remote interface MTU: 9192
```

Cause: MTU mismatch between PEs

Solution:

```
## Option 1: Increase local MTU
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 mtu 9192

## Option 2: Configure ignore-mtu-mismatch (not recommended for production)
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 ignore-mtu-mismatch
```

Troubleshooting Scenario 4: VC-ID Mismatch

Symptom: L2Circuit doesn't come up at all

```
admin@PE1> show l2circuit connections
## No output for the expected circuit
```

Diagnosis:

```
## Check LDP database
admin@PE1> show ldp database l2circuit
Input label database, 10.1.1.1:0--10.2.2.2:0
  Label     Prefix
  100045    VC-ID: 5000

Output label database, 10.1.1.1:0--10.2.2.2:0
  Label     Prefix
  ## No matching VC-ID from remote!
```

Cause: VC-ID mismatch between PEs

Solution:

```
## Ensure both PEs use the same VC-ID
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 virtual-circuit-id 5000
```

Module 09: L2Circuit—Troubleshooting

Part 1: The Conceptual Lecture (The Why)

Understanding Pseudowire Status TLV

In traditional circuits, you'd have physical indicators - link lights, electrical signals - to show circuit health. In virtual circuits, we need a protocol mechanism. This is where the Pseudowire Status TLV (Type-Length-Value) comes in.

What Problems Does It Solve?

1. **Silent Failures**: Without status TLV, a PE might keep sending traffic into a black hole
2. **Faster Convergence**: Immediate notification vs waiting for timeouts
3. **Granular Status**: Specific error codes explain exactly what's wrong

How Pseudowire Status TLV Works

```
Normal Operation:
PE1 ←→ PE2
Status: 0x00 (No errors)

When PE2's attachment circuit fails:
PE1 ← PE2
Status: 0x01 (Pseudowire not forwarding)

PE1 can now:
- Stop sending traffic
```

```
- Trigger backup paths
- Alert management systems
```

L2Circuit Error Code Architecture

L2Circuit uses specific codes to indicate problems:

```
0x00: Pseudowire forwarding (all good)
0x01: Pseudowire not forwarding
0x02: Local attachment circuit receive fault
0x03: Local attachment circuit transmit fault
0x04: Local PSN-facing PW transmit fault
0x05: Local PSN-facing PW receive fault
```

Each code tells a story:

- **0x01**: Generic "something's wrong"
- **0x02**: "I can't receive from my local CE"
- **0x03**: "I can't transmit to my local CE"
- **0x04**: "I can't send into the MPLS network"

Status Propagation Flow

```
CE1 --- PE1 ===MPLS=== PE2 --- CE2
         |              |
         |              +-- Interface down
         |              +-- Sends Status 0x01
        +-- Receives status
        +-- Puts AC in standby
```

This propagation prevents:

- Black-holing of traffic
- Wasted bandwidth
- STP loops in ethernet networks

Part 2: The Junos CLI Masterclass (The How)

Configuring Pseudowire Status TLV

Basic configuration is simple but critical:

```
## Enable status TLV on L2Circuit
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 pseudowire-status-tlv
```

Advanced Status TLV Configuration

Hot-standby with status TLV:

```
## Primary L2Circuit
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 virtual-circuit-id 1000
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 pseudowire-status-tlv
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 standby 10.3.3.3

## This creates automatic failover when primary signals failure
```

Status TLV with BFD for faster detection:

```
## Configure BFD for L2Circuit
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection minimum-interval 300
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection multiplier 3
```

Interpreting Error Codes in Junos

When troubleshooting, Junos translates numeric codes to readable messages:

```
admin@PE1> show l2circuit connections extensive
  Status TLV support: Yes
  Last received status: Pseudowire not forwarding (0x01)
  Last sent status: Pseudowire forwarding (0x00)
```

Configuration for Different Failure Scenarios

Scenario 1: Reflect CE link status

```
## If CE link goes down, signal remote PE
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 pseudowire-status-tlv
```

Scenario 2: Maintenance mode

```
## Gracefully take circuit out of service
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 down
## This sends status 0x01 to remote PE
```

Scenario 3: Ignore remote failures (not recommended)

```
## Continue forwarding despite remote status
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 ignore-pseudowire-status
```

Part 3: Verification & Troubleshooting (The What-If)

Comprehensive Status Verification

```
## Check detailed status information
show l2circuit connections interface ge-0/0/0.0 extensive

## Monitor status changes in real-time
monitor traffic interface ge-0/0/0.0 layer2-headers

## Check LDP message exchanges
show ldp traffic-statistics
```

Troubleshooting Scenario 1: Status TLV Not Negotiated

Symptom: Features requiring status TLV don't work properly

```
admin@PE1> show l2circuit connections extensive
  Status TLV support: No  ## Problem!
  Last received status: Not received
```

Diagnosis:

```
## Check configuration on both PEs
admin@PE1> show configuration protocols l2circuit neighbor 10.2.2.2
interface ge-0/0/0.0 {
    virtual-circuit-id 1000;
    ## pseudowire-status-tlv is missing!
}
```

Cause: Status TLV not enabled on one or both PEs

Solution:

```
## Enable on both PEs
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 pseudowire-status-tlv
commit synchronize
```

Troubleshooting Scenario 2: Receiving Error Code 0x02

Symptom: Remote PE reports receive fault

```
admin@PE1> show l2circuit connections extensive
  Last received status: Local attachment circuit receive fault (0x02)
  Remote interface status: Down
```

Diagnosis:

```
## Check remote PE's interface
admin@PE2> show interfaces ge-0/0/0 extensive | match "error|drop"
  Input errors: 15234  ## High error count!
  Framing errors: 15234
```

Cause: Physical layer problems on remote attachment circuit

Solution:

1. Check physical cabling at remote site
2. Verify duplex/speed settings
3. Replace faulty optics if needed

Troubleshooting Scenario 3: Persistent Error Code 0x01

Symptom: Generic "not forwarding" status

```
admin@PE1> show l2circuit connections extensive
  Last received status: Pseudowire not forwarding (0x01)
  Local interface: ge-0/0/0.0, Status: Up  ## Local is fine
```

Diagnosis Deep Dive:

```
## Check multiple aspects on remote PE
admin@PE2> show interfaces ge-0/0/0.0 terse
ge-0/0/0.0              up    down ## Admin up, protocol down!

admin@PE2> show log messages | match "ge-0/0/0|L2CIRCUIT"
Oct 30 10:45:22 L2CIRCUIT: Circuit to 10.1.1.1 VC-ID 1000 status change: AC failure
```

Common Causes:

1. Remote interface admin up but protocol down
2. MPLS LSP to remote PE is down
3. LDP session flapping
4. Resource exhaustion (labels, memory)

Solution Flowchart:

```
Check remote interface → Fix physical/config issues
         ↓ If OK
Check MPLS LSP → Fix routing/RSVP issues
         ↓ If OK
Check LDP session → Fix LDP config/reachability
         ↓ If OK
Check resources → Increase label space/memory
```

Troubleshooting Scenario 4: Status TLV Causing Flaps

Symptom: L2Circuit flapping between up/down states

```
admin@PE1> show l2circuit connections history
Time                    Event
10:30:15               Up
```

```
10:30:18               Down (Status: 0x02)
10:30:22               Up
10:30:25               Down (Status: 0x02)
## Rapid flapping pattern
```

Diagnosis:

```
## Check for interface flapping
admin@PE2> show interfaces ge-0/0/0 extensive | match "carrier transitions"
  Carrier transitions: 847  ## Very high!
```

Cause: Unstable attachment circuit causing status TLV floods

Solution:

```
## Temporary: Add hold-time to dampen flaps
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 hold-time up 10 down 10

## Permanent: Fix the physical instability
## - Check/replace cabling
## - Verify power to CE device
## - Look for environmental issues (temperature, vibration)
```

Advanced Troubleshooting Commands

```
## Decode raw LDP messages
show ldp traffic-statistics | match "Status"
monitor traffic interface lo0.0 matching "port 646" detail

## Check internal L2Circuit state machine
show l2circuit connections interface ge-0/0/0.0 history detail

## Verify label operations
show route forwarding-table label 100045 detail
```

Module 10: L2Circuit—Advanced Concepts

Part 1: The Conceptual Lecture (The Why)

Virtual Circuit Connectivity Verification (VCCV)

Imagine driving through a tunnel - you want to know if you'll emerge on the other side before entering. VCCV provides this assurance for L2Circuits by actively testing the data plane.

Traditional Problem: Control plane says "up" but data plane is broken

```
Control Plane: "Labels exchanged, circuit is up!"
Data Plane: [Silent failure - packets dropped]
Customer: "Why isn't my traffic passing?"
```

VCCV Solution: Actively probe the exact path that customer data takes

How VCCV Works

VCCV uses three components:

1. **Control Channel**: How to send test packets
   - In-band: Test packets follow same path as data
   - Out-of-band: Separate control packets
2. **Connectivity Verification**: What to send
   - ICMP ping
   - LSP ping

- BFD packets

3. **Results**: Proof of bidirectional connectivity

```
Data Flow:
CE1 → PE1 → [MPLS Label 100] → P routers → PE2 → CE2

VCCV Test:
PE1 → [MPLS Label 100 + Control Word + VCCV] → PE2
PE2 → [MPLS Label 200 + Control Word + VCCV] → PE1
Result: Confirmed bidirectional forwarding
```

L2Circuit Multihoming

Business Need: No single point of failure

Traditional multihoming required complex CE configurations. L2Circuit multihoming simplifies this:

Key Concepts:

- **Primary/Backup**: One active path, automatic failover
- **Load balancing**: Active/Active using different VC-IDs
- **Fate sharing**: Multiple circuits fail together if needed

Local Switching

The Efficiency Play: Why send traffic across the MPLS core when both endpoints connect to the same PE?

```
Traditional:
CE1 → PE → MPLS Core → PE (same device) → CE2

Local Switching:
CE1 → PE → CE2 (direct switching in PE)
```

Benefits:

- Reduced latency (no core traversal)
- Bandwidth savings
- Simplified troubleshooting

Interworking

The Challenge: Different Layer 2 technologies at each end

Example scenarios:

- Ethernet at Site A, Frame Relay at Site B
- VLAN-tagged at Site A, untagged at Site B
- Different MTU sizes

Interworking provides translation:

```
[Ethernet Frame] → PE1 → [Interworking] → [ATM Cells] → PE2
                         Preserves payload
                         Translates headers
```

Part 2: The Junos CLI Masterclass (The How)

Configuring VCCV

Basic VCCV with BFD:

```
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection minimum-interval 100
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection multiplier 3
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam bfd-liveness-detection no-adaptation
```

VCCV with ping capability:

```
## Enable VCCV ping
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 oam ping-interval 30
```

Configuring Multihoming

Primary/Backup Configuration:

On CE (if using MC-LAG):

```
set interfaces ae0 aggregated-ether-options mc-ae mc-ae-id 1
set interfaces ae0 aggregated-ether-options mc-ae redundancy-group 1
set interfaces ae0 aggregated-ether-options mc-ae chassis-id 0
set interfaces ae0 aggregated-ether-options mc-ae mode active-active
```

On Primary PE1:

```
## Primary circuit
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 virtual-circuit-id 1000
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 pseudowire-status-tlv
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 standby 10.200.200.200
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 oam bfd-liveness-detection
```

On Backup PE2:

```
## Backup circuit - same VC-ID
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 virtual-circuit-id 1000
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 pseudowire-status-tlv
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 backup
```

Configuring Local Switching

Same PE, different interfaces:

```
## Create local cross-connect
set protocols l2circuit local-switching interface ge-0/0/1.0 interface ge-0/0/2.0
set protocols l2circuit local-switching interface ge-0/0/1.0 description "Local switch Customer-A"
set protocols l2circuit local-switching interface ge-0/0/1.0 encapsulation-type ethernet
```

With VLAN manipulation:

```
## Pop VLAN on ingress, push on egress
set interfaces ge-0/0/1 unit 100 input-vlan-map pop
set interfaces ge-0/0/2 unit 0 output-vlan-map push vlan-id 200

set protocols l2circuit local-switching interface ge-0/0/1.100 interface ge-0/0/2.0
```

Configuring Interworking

Ethernet to Ethernet-VLAN:

```
## On PE1 (Ethernet side)
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 virtual-circuit-id 3000
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 encapsulation-type ethernet
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 interworking

## On PE2 (VLAN side)
set interfaces ge-0/0/0 unit 100 encapsulation vlan-ccc
set interfaces ge-0/0/0 unit 100 vlan-id 100
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.100 virtual-circuit-id 3000
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.100 encapsulation-type ethernet
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.100 interworking
```

Complete Advanced Configuration Example

```
## Advanced L2Circuit with all features

## Physical interfaces
set interfaces ge-0/0/0 description "Primary to Customer"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0 family ccc

set interfaces ge-0/0/1 description "Local switch endpoint A"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

set interfaces ge-0/0/2 description "Local switch endpoint B"
set interfaces ge-0/0/2 encapsulation ethernet-ccc
set interfaces ge-0/0/2 unit 0 family ccc

## L2Circuit with VCCV and multihoming
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 virtual-circuit-id 5000
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 pseudowire-status-tlv
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 control-word
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 standby 10.200.200.200
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 revert-time 300
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 oam bfd-liveness-detection
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 oam bfd-liveness-detection minimum-interval 50
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 oam bfd-liveness-detection multiplier 3

## Local switching configuration
set protocols l2circuit local-switching interface ge-0/0/1.0 interface ge-0/0/2.0
set protocols l2circuit local-switching interface ge-0/0/1.0 description "Metro-Ethernet-Local"
set protocols l2circuit local-switching interface ge-0/0/1.0 encapsulation-type ethernet
```

Part 3: Verification & Troubleshooting (The What-If)

Verification Commands for Advanced Features

```
## Verify VCCV operation
show oam ethernet connectivity-fault-management l2circuit-state

## Check BFD for L2Circuit
show bfd session detail

## Verify multihoming status
show l2circuit connections | match "standby|backup"

## Check local switching
show l2circuit local-switching
```

Troubleshooting Scenario 1: VCCV BFD Not Establishing

Symptom: BFD session down despite L2Circuit being up

```
admin@PE1> show bfd session
                                                  Detect   Transmit
Address         State     Interface    Time     Interval  Multiplier
10.2.2.2        Down                   0.000     1.000     3
```

Diagnosis:

```
admin@PE1> show l2circuit connections extensive
  OAM status: BFD not supported by remote
```

Cause: VCCV not supported or misconfigured on remote PE

Solution:

```
## Ensure both PEs have matching VCCV configuration
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 oam bfd-liveness-detection

## Verify control-word is enabled (required for VCCV)
set protocols l2circuit neighbor 10.1.1.1 interface ge-0/0/0.0 control-word
```

Troubleshooting Scenario 2: Multihoming Switchover Failing

Symptom: Backup circuit doesn't activate when primary fails

```
admin@PE1> show l2circuit connections
ge-0/0/0.0(vc 5000)       rmt   Dn     ## Primary down
  Standby: 10.200.200.200

## But backup doesn't show as active!
```

Diagnosis:

```
## Check if standby is configured properly
admin@PE2> show configuration protocols l2circuit
neighbor 10.100.100.100 {
    interface ge-0/0/0.0 {
        virtual-circuit-id 5000;
        ## Missing 'backup' statement!
    }
}
```

Cause: Backup PE not configured as backup

Solution:

```
## On backup PE:
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 backup

## Optional: Configure revert timer on primary
set protocols l2circuit neighbor 10.100.100.100 interface ge-0/0/0.0 revert-time 300
```

Troubleshooting Scenario 3: Local Switching Not Working

Symptom: Traffic not passing between locally switched interfaces

```
admin@PE1> show l2circuit local-switching
Local switching connections:

Legend for interface status
Up -- operational
Dn -- down

Interface                 State
ge-0/0/1.0                  Dn
  Local switching status: Inactive
```

Diagnosis:

```
## Check interface family
admin@PE1> show interfaces ge-0/0/1.0 | match family
  Logical interface ge-0/0/1.0 (Index 335)
    Protocol inet, MTU: 1500  ## Wrong family!
```

Cause: Interfaces not configured with family CCC

Solution:

```
delete interfaces ge-0/0/1 unit 0 family inet
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0 family ccc

## Same for ge-0/0/2.0
```

Troubleshooting Scenario 4: Interworking Frame Loss

Symptom: Some frames pass, others are dropped in interworking scenario

```
## Customer reports 30% packet loss
admin@CE1> ping 192.168.1.2 count 100
...
100 packets transmitted, 72 received, 28% packet loss
```

Diagnosis:

```
admin@PE1> show l2circuit connections extensive
  Encapsulation: ETHERNET
  Interworking: Yes
  Remote encapsulation: VLAN
  MTU: 1514
  Remote MTU: 1518  ## MTU difference!
```

Cause: MTU mismatch in interworking (VLAN adds 4 bytes)

Solution:

```
## Adjust MTU to account for VLAN tag
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 mtu 1518

## Or use ignore-mtu-mismatch with caution
set protocols l2circuit neighbor 10.2.2.2 interface ge-0/0/0.0 ignore-mtu-mismatch
```

Module 11: FEC 129 Pseudowires

Part 1: The Conceptual Lecture (The Why)

The Evolution from FEC 128 to FEC 129

FEC (Forwarding Equivalence Class) 128 represents traditional L2Circuit where you manually configure every endpoint. Think of it like programming speed-dial numbers on every phone - it works, but it's tedious at scale.

FEC 129 revolutionizes this by introducing **autodiscovery**. Instead of manual configuration on every PE, routers automatically discover pseudowire endpoints.

```
FEC 128 (Traditional):
PE1: "Connect to PE2, VC-ID 1000"
PE2: "Connect to PE1, VC-ID 1000"
PE3: "Connect to PE4, VC-ID 1001"
... (manual for every circuit)
```

```
FEC 129 (Autodiscovery):
All PEs: "I'll advertise what I have via BGP"
Result: Automatic discovery and connection
```

The Hybrid Architecture

FEC 129 brilliantly combines two protocols:

- **BGP**: For autodiscovery and endpoint advertisement
- **LDP**: For label signaling and pseudowire establishment

This hybrid approach leverages the strengths of each:

```
BGP's Strength: Scalable information distribution
"Hey network, I have CE customer-A at my location"

LDP's Strength: Efficient label allocation
"Now let's exchange labels for the actual data path"
```

How FEC 129 Autodiscovery Works

1. **BGP Advertisement Phase**:

```
PE1 → BGP → "I have AGI:10.1.1.1:100 with SAII:1"
PE2 → BGP → "I have AGI:10.1.1.1:100 with SAII:2"
```

Key identifiers:

- **AGI** (Attachment Group Identifier): Groups related circuits
- **SAII** (Source Attachment Individual Identifier): Unique endpoint ID
- **TAII** (Target Attachment Individual Identifier): Remote endpoint ID

2. **Matching Process**:

```
PE1 sees: "PE2 has SAII:2 for same AGI"
PE1 thinks: "My TAII:2 matches their SAII:2 – we should connect!"
```

3. **LDP Signaling Phase**:

```
PE1 → LDP → PE2: "Label mapping for pseudowire"
PE2 → LDP → PE1: "Label mapping for pseudowire"
```

Benefits Over Traditional L2Circuit

Feature	FEC 128 (Traditional)	FEC 129
Configuration	O(n²) - every PE pair	O(n) - just local info
New site addition	Touch multiple PEs	Touch only new PE
Troubleshooting	Check all endpoints	BGP shows all endpoints
Scaling	Limited by manual work	Automated discovery

Part 2: The Junos CLI Masterclass (The How)

FEC 129 Configuration Structure

The configuration involves three main components:

1. BGP configuration for autodiscovery
2. L2VPN routing instance
3. FEC 129 specific parameters

Step-by-Step FEC 129 Configuration

Step 1: Configure BGP for L2VPN signaling:

```
set protocols bgp group IBGP-L2VPN type internal
set protocols bgp group IBGP-L2VPN local-address 10.1.1.1
set protocols bgp group IBGP-L2VPN family l2vpn auto-discovery-only
set protocols bgp group IBGP-L2VPN neighbor 10.255.255.255 description "Route Reflector"
```

Step 2: Configure the routing instance:

```
set routing-instances FEC129-VPLS instance-type l2vpn
set routing-instances FEC129-VPLS interface ge-0/0/0.0
set routing-instances FEC129-VPLS route-distinguisher 10.1.1.1:129
set routing-instances FEC129-VPLS l2vpn-id l2vpn-id:129:100
set routing-instances FEC129-VPLS vrf-target target:65000:129
```

Step 3: Configure FEC 129 parameters:

```
set routing-instances FEC129-VPLS protocols l2vpn auto-discovery-only
set routing-instances FEC129-VPLS protocols l2vpn signaling-protocol ldp
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A site-identifier 1
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A automatic-site-id
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A interface ge-0/0/0.0
```

Understanding Key Parameters

l2vpn-id: Unique identifier for the L2VPN domain

```
## Format: l2vpn-id:<AS-number>:<VPN-number>
set routing-instances FEC129-VPLS l2vpn-id l2vpn-id:65000:100
```

automatic-site-id: Let Junos handle SAII/TAII generation

```
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A automatic-site-id
## Junos uses interface index as site ID
```

Manual site configuration (when needed):

```
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A source-attachment-identifier 10.1.1.1:1
set routing-instances FEC129-VPLS protocols l2vpn site Customer-A target-attachment-identifier 10.2.2.2:1
```

FEC 129 with VPLS

FEC 129 truly shines with VPLS (multipoint):

```
## Complete FEC 129 VPLS configuration
set interfaces ge-0/0/0 description "Customer A - Site 1"
set interfaces ge-0/0/0 encapsulation ethernet-vpls
set interfaces ge-0/0/0 unit 0 family vpls

set routing-instances VPLS-FEC129 instance-type vpls
set routing-instances VPLS-FEC129 interface ge-0/0/0.0
set routing-instances VPLS-FEC129 route-distinguisher 10.1.1.1:500
set routing-instances VPLS-FEC129 l2vpn-id l2vpn-id:65000:500
set routing-instances VPLS-FEC129 vrf-target target:65000:500
set routing-instances VPLS-FEC129 protocols vpls no-tunnel-services
set routing-instances VPLS-FEC129 protocols vpls signaling-protocol ldp
set routing-instances VPLS-FEC129 protocols vpls site-range 10
set routing-instances VPLS-FEC129 protocols vpls site Customer-A site-identifier 1
set routing-instances VPLS-FEC129 protocols vpls site Customer-A automatic-site-id
set routing-instances VPLS-FEC129 protocols vpls site Customer-A interface ge-0/0/0.0
```

Advanced FEC 129 Features

Mesh groups to prevent loops:

```
set routing-instances VPLS-FEC129 protocols vpls mesh-group CORE
set routing-instances VPLS-FEC129 protocols vpls mesh-group CORE site Customer-A
```

MAC limiting:

```
set routing-instances VPLS-FEC129 protocols vpls mac-table-size 1000
set routing-instances VPLS-FEC129 protocols vpls interface ge-0/0/0.0 mac-limit 100
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

```
## Check BGP autodiscovery advertisements
show route table bgp.l2vpn.0 detail

## Verify FEC 129 pseudowire status
show l2vpn connections instance FEC129-VPLS

## Check LDP FEC 129 bindings
show ldp database l2vpn

## Verify autodiscovery operation
show l2vpn autodiscovery
```

Understanding FEC 129 Route Output

```
admin@PE1> show route table bgp.l2vpn.0 detail

bgp.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
10.1.1.1:129:100:1/96 (1 entry, 1 announced)
        *L2VPN  Preference: 170/-101
                Next hop type: Indirect
                Address: 0x9458f04
                Next-hop reference count: 3
                Protocol next hop: 10.1.1.1
                Indirect next hop: 0x0 -
                State: <Active Int Ext>
                Age: 2:15:47    Metric2: 1
                Validation State: unverified
                Task: FEC129-VPLS-l2vpn
                Announcement bits (1): 1-BGP_RT_Background
                AS path: I
                Communities: target:65000:129 Layer2-info: encaps:VPLS, control-word:No, site-id:1
                Import Accepted
                Localpref: 100
                Router ID: 10.1.1.1
                Primary Routing Table bgp.l2vpn.0
```

Troubleshooting Scenario 1: No Autodiscovery Routes

Symptom: FEC 129 configured but no remote sites discovered

```
admin@PE1> show route table bgp.l2vpn.0
## Empty or only local routes
```

Diagnosis:

```
admin@PE1> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0                 0          0          0          0          0          0
## Missing bgp.l2vpn.0 table!
```

Cause: BGP not configured for L2VPN family

Solution:

```
set protocols bgp group IBGP-L2VPN family l2vpn auto-discovery-only
```

Troubleshooting Scenario 2: LDP Signaling Failure

Symptom: BGP autodiscovery works but pseudowires don't establish

```
admin@PE1> show l2vpn connections instance FEC129-VPLS
Instance: FEC129-VPLS
  L2vpn-id: 65000:500
  Local-id: 1
    Remote-id: 2, Status: LN  ## LDP Neighbor missing!
```

Diagnosis:

```
admin@PE1> show ldp neighbor 10.2.2.2
## No targeted LDP session
```

Cause: Targeted LDP not enabled

Solution:

```
set protocols ldp targeted-hello
set protocols ldp interface lo0.0
```

Troubleshooting Scenario 3: Site ID Conflicts

Symptom: Some sites connect, others don't

```
admin@PE1> show l2vpn connections instance VPLS-FEC129
  Local site: Customer-A (1)
    connection-site           Type  St     Time last up          # Up trans
    2                         rmt   Up     Oct 30 12:30:15 2024         1
      Remote PE: 10.2.2.2, Negotiated control-word: No
      Incoming label: 300123, Outgoing label: 300456
    1                         rmt   ??     ## Conflict!
```

Diagnosis:

```
admin@PE1> show log messages | match "site-id conflict"
Oct 30 12:35:00 L2VPN: Site-id conflict detected for instance VPLS-FEC129
```

Cause: Multiple PEs using same site identifier

Solution:

```
## Use automatic-site-id to avoid conflicts
set routing-instances VPLS-FEC129 protocols vpls site Customer-A automatic-site-id

## Or ensure unique manual site-ids across all PEs
```

Troubleshooting Scenario 4: AGI/SAII Mismatch

Symptom: PEs discover each other but can't establish pseudowire

```
admin@PE1> show l2vpn autodiscovery
L2VPN Id: 65000:500
  Local advertisements:
    AGI: 10.1.1.1:100, SAII: 10.1.1.1:1, Local-id: 1
  Remote advertisements:
```

```
    AGI: 10.2.2.200, SAII: 10.2.2.1, Remote-id: 1
    ## Different AGI values!
```

Cause: Mismatched l2vpn-id configuration

Solution:

```
## Ensure all PEs use same l2vpn-id
set routing-instances FEC129-VPLS l2vpn-id l2vpn-id:65000:500
```

Module 12: Virtual Private LAN Service—Introduction

Part 1: The Conceptual Lecture (The Why)

The Multipoint Challenge

Imagine you're running IT for a company with 50 branch offices. With point-to-point L2VPNs, you'd need 1,225 connections (n*(n-1)/2) for full connectivity. This is the classic "n-squared problem."

VPLS solves this by creating a virtual switch in the service provider network:

```
Traditional L2VPN (Point-to-Point):
Site A ←→ Site B
Site A ←→ Site C
Site B ←→ Site C
(3 separate circuits for 3 sites)

VPLS (Multipoint):
Site A ←→ Virtual Switch ←→ Site B
              ↑
              |
           Site C
(All sites connect to one logical entity)
```

How VPLS Works

VPLS emulates an Ethernet switch across an MPLS network:

1. **MAC Learning**: Like a physical switch, VPLS learns MAC addresses
2. **Forwarding**: Unicast to specific PE, broadcast/multicast to all PEs
3. **Loop Prevention**: Split-horizon rule prevents loops

```
MAC Learning Process:
1. Host at Site A (MAC: AA:AA) sends frame
2. PE1 learns: MAC AA:AA is behind my local interface
3. PE1 floods frame to all remote PEs (PE2, PE3)
4. Remote PEs learn: MAC AA:AA is reachable via PE1
```

VPLS vs Regular Pseudowire

Aspect	Pseudowire (L2VPN/L2Circuit)	VPLS
Topology	Point-to-point	Multipoint
MAC Learning	No	Yes
Broadcast	Not applicable	Flooded to all sites
Scalability	O(n²) circuits	O(n) sites
Use Case	Connecting 2 locations	Connecting multiple locations

The Two Signaling Approaches

VPLS can be built using either BGP or LDP signaling:

BGP-Signaled VPLS:

- Uses BGP to distribute membership and labels
- Better for inter-AS deployments
- More complex but more flexible

LDP-Signaled VPLS:

- Uses targeted LDP sessions between all PEs
- Simpler configuration
- Automatic full mesh creation

```
BGP Approach:
PE1 → BGP → "I'm in VPLS 100, here's my label"

LDP Approach:
PE1 ←→ PE2: Targeted LDP
PE1 ←→ PE3: Targeted LDP
PE2 ←→ PE3: Targeted LDP
(Full mesh of LDP sessions)
```

Traffic Forwarding in VPLS

Unknown Unicast/Broadcast/Multicast (BUM) Traffic:

```
1. PE1 receives broadcast frame from CE
2. PE1 replicates frame to all remote PEs
3. Each remote PE forwards to local CE
4. Split-horizon: Remote PEs don't forward to other remote PEs
```

Known Unicast Traffic:

```
1. PE1 has MAC AA:AA in local table → points to PE2
2. PE1 encapsulates frame with PE2's label
3. Direct forwarding to PE2 only
4. No unnecessary replication
```

Part 2: The Junos CLI Masterclass (The How)

VPLS Configuration Hierarchy

VPLS configuration uses the routing-instances hierarchy:

```
[edit routing-instances <name>]
  instance-type vpls;
  protocols {
      vpls {
          ## VPLS-specific configuration
      }
  }
```

BGP-Signaled VPLS Configuration

Step 1: Configure interfaces:

```
set interfaces ge-0/0/0 description "VPLS Customer - Site A"
set interfaces ge-0/0/0 encapsulation ethernet-vpls
set interfaces ge-0/0/0 unit 0 family vpls
```

Step 2: Configure BGP for VPLS signaling:

```
set protocols bgp group IBGP-VPLS type internal
set protocols bgp group IBGP-VPLS local-address 10.1.1.1
set protocols bgp group IBGP-VPLS family l2vpn signaling
set protocols bgp group IBGP-VPLS neighbor 10.255.255.255 description "Route Reflector"
```

Step 3: Configure VPLS instance:

```
set routing-instances VPLS-GOLD instance-type vpls
set routing-instances VPLS-GOLD interface ge-0/0/0.0
set routing-instances VPLS-GOLD route-distinguisher 10.1.1.1:100
set routing-instances VPLS-GOLD vrf-target target:65000:100
set routing-instances VPLS-GOLD protocols vpls site-range 50
set routing-instances VPLS-GOLD protocols vpls no-tunnel-services
set routing-instances VPLS-GOLD protocols vpls site SITE-A site-identifier 1
set routing-instances VPLS-GOLD protocols vpls site SITE-A interface ge-0/0/0.0
```

Key BGP-VPLS Parameters:

- `site-range` : Maximum number of sites (must be same on all PEs)
- `site-identifier` : Unique ID per site (1-range)
- `no-tunnel-services` : Don't use tunnel interfaces (common)

LDP-Signaled VPLS Configuration

Different approach - specify VPLS ID and neighbors:

```
## Same interface configuration
set interfaces ge-0/0/0 description "VPLS Customer - Site A"
set interfaces ge-0/0/0 encapsulation ethernet-vpls
set interfaces ge-0/0/0 unit 0 family vpls

## LDP configuration
set protocols ldp interface lo0.0
set protocols ldp targeted-hello

## VPLS instance with LDP signaling
set routing-instances VPLS-SILVER instance-type vpls
set routing-instances VPLS-SILVER interface ge-0/0/0.0
set routing-instances VPLS-SILVER protocols vpls vpls-id 200
set routing-instances VPLS-SILVER protocols vpls neighbor 10.2.2.2
set routing-instances VPLS-SILVER protocols vpls neighbor 10.3.3.3
set routing-instances VPLS-SILVER protocols vpls neighbor 10.4.4.4
```

FEC 129 VPLS (Autodiscovery with LDP)

Best of both worlds - BGP autodiscovery, LDP signaling:

```
## BGP for autodiscovery
set protocols bgp group IBGP-VPLS family l2vpn auto-discovery-only

## VPLS instance
set routing-instances VPLS-BRONZE instance-type vpls
set routing-instances VPLS-BRONZE interface ge-0/0/0.0
set routing-instances VPLS-BRONZE route-distinguisher 10.1.1.1:300
set routing-instances VPLS-BRONZE l2vpn-id l2vpn-id:65000:300
set routing-instances VPLS-BRONZE vrf-target target:65000:300
set routing-instances VPLS-BRONZE protocols vpls signaling-protocol ldp
set routing-instances VPLS-BRONZE protocols vpls site-range 20
set routing-instances VPLS-BRONZE protocols vpls site SITE-A site-identifier 1
set routing-instances VPLS-BRONZE protocols vpls site SITE-A automatic-site-id
set routing-instances VPLS-BRONZE protocols vpls site SITE-A interface ge-0/0/0.0
```

VPLS MAC Table Management

```
## Limit MAC addresses learned
set routing-instances VPLS-GOLD protocols vpls mac-table-size 5000
set routing-instances VPLS-GOLD protocols vpls interface ge-0/0/0.0 mac-limit 100
```

```
## MAC aging time
set routing-instances VPLS-GOLD protocols vpls mac-table-aging-time 600

## Static MAC entry
set routing-instances VPLS-GOLD protocols vpls static-mac 00:11:22:33:44:55 interface ge-0/0/0.0
```

Part 3: Verification & Troubleshooting (The What-If)

Essential VPLS Verification Commands

```
## Overall VPLS status
show vpls connections

## Detailed VPLS information
show vpls connections instance VPLS-GOLD extensive

## MAC table
show vpls mac-table instance VPLS-GOLD

## Statistics
show vpls statistics instance VPLS-GOLD
```

Understanding VPLS Connection Output

```
admin@PE1> show vpls connections instance VPLS-GOLD
Instance: VPLS-GOLD
  L2-id: 10.1.1.1:100
  Site-id: 1
  Number of local interfaces: 1
  Number of local interfaces up: 1
  Number of remote PEs: 3
  Number of remote PEs up: 3
  lsi.1048576         2               10.2.2.2         Up
  lsi.1048577         3               10.3.3.3         Up
  lsi.1048578         4               10.4.4.4         Up

Labels:
  Label-base        Offset     Size  Range
  262145            1          50    50
```

Troubleshooting Scenario 1: VPLS Site Not Joining

Symptom: Remote PE not showing in VPLS connections

```
admin@PE1> show vpls connections instance VPLS-GOLD
  Number of remote PEs: 2  ## Should be 3!
```

Diagnosis:

```
admin@PE1> show route receive-protocol bgp 10.255.255.255 table bgp.l2vpn.0 | match VPLS-GOLD
## Missing routes from PE3

admin@PE3> show configuration routing-instances VPLS-GOLD
## Different vrf-target!
```

Cause: Mismatched vrf-target on PE3

Solution:

```
## Ensure all PEs use same vrf-target
set routing-instances VPLS-GOLD vrf-target target:65000:100
```

Troubleshooting Scenario 2: MAC Learning Issues

Symptom: Unicast flooding (all traffic treated as unknown)

```
admin@PE1> show vpls mac-table instance VPLS-GOLD
## Very few or no remote MACs

admin@PE1> show vpls statistics instance VPLS-GOLD
  Broadcast packets: 1000000  ## Very high!
  Unicast packets: 1000  ## Very low!
```

Diagnosis:

```
admin@PE1> show vpls connections instance VPLS-GOLD extensive
  Remote site: 2
    Outgoing label: 262146
    MAC learning: disabled  ## Problem!
```

Cause: MAC learning disabled or MAC table full

Solution:

```
## Increase MAC table size
set routing-instances VPLS-GOLD protocols vpls mac-table-size 10000

## Check for mac-limit on interfaces
delete routing-instances VPLS-GOLD protocols vpls interface ge-0/0/0.0 mac-limit
```

Troubleshooting Scenario 3: Loop in VPLS

Symptom: Broadcast storm, high CPU utilization

```
admin@PE1> show system processes extensive | match rpd
  1234 root     85    0  1.2g  800m  rpd  ## High CPU!

admin@PE1> show vpls statistics instance VPLS-GOLD
  Broadcast packets: 50000000  ## Exponential growth!
```

Diagnosis:

```
admin@PE1> show spanning-tree interface
## Customer might have loop between sites

admin@PE1> show log messages | match "MAC move"
Oct 30 14:22:01 rpd[1234]: MAC move detected: 00:11:22:33:44:55 from lsi.1048576 to ge-0/0/0.0
Oct 30 14:22:02 rpd[1234]: MAC move detected: 00:11:22:33:44:55 from ge-0/0/0.0 to lsi.1048576
## Rapid MAC movement indicates loop
```

Cause: Loop in customer network

Solution:

```
## Enable storm control
set routing-instances VPLS-GOLD protocols vpls interface ge-0/0/0.0 storm-control bandwidth-level 10000

## Enable MAC move detection
set routing-instances VPLS-GOLD protocols vpls mac-move-limit 5 mac-move-time-window 10 action drop
```

Troubleshooting Scenario 4: LDP-Signaled VPLS Not Establishing

Symptom: LDP VPLS shows no connections

```
admin@PE1> show vpls connections instance VPLS-SILVER
Instance: VPLS-SILVER
  VPLS-id: 200
  Number of remote PEs: 0  ## No connections!
```

Diagnosis:

```
admin@PE1> show ldp neighbor
## Check for targeted LDP sessions

admin@PE1> show configuration protocols vpls
## Missing vpls-id or neighbor configuration
```

Cause: Missing neighbor configuration or VPLS ID mismatch

Solution:

```
## Add all remote PE neighbors
set routing-instances VPLS-SILVER protocols vpls neighbor 10.2.2.2
set routing-instances VPLS-SILVER protocols vpls neighbor 10.3.3.3

## Ensure VPLS ID matches on all PEs
set routing-instances VPLS-SILVER protocols vpls vpls-id 200
```

This completes the comprehensive learning modules for L2VPN Advanced Concepts, L2Circuit, FEC 129 Pseudowires, and VPLS Introduction. Each module has provided you with the conceptual understanding, practical configuration skills, and troubleshooting expertise needed to master these technologies for the JNCIE-SP exam.

Module 13: VPLS—BGP Configuration and Verification

Part 1: The Conceptual Lecture (The Why)

Understanding the Problem VPLS Solves

Imagine you're a large corporation with offices in New York, London, and Tokyo. Each office has hundreds of computers that need to communicate as if they were all plugged into the same switch—sharing broadcasts, using the same IP subnet, and discovering each other through ARP. The challenge? These offices are separated by thousands of miles and connected through a service provider's MPLS network.

This is where **Virtual Private LAN Service (VPLS)** comes in. VPLS creates the illusion that all your sites are connected to a single, giant Ethernet switch spanning the globe.

What is VPLS?

VPLS is a multipoint Layer 2 VPN service that emulates an Ethernet LAN across an MPLS network. Unlike point-to-point pseudowires (which we covered in L2VPN), VPLS connects multiple sites in a full-mesh topology, allowing any site to communicate directly with any other site.

```
Traditional L2VPN (Point-to-Point):
Site A <----pseudowire----> Site B

VPLS (Multipoint):
        Site A
         /|\
        / | \
       /  |  \
      /   |   \
   Site B-+-Site C
      \   |   /
       \  |  /
        \ | /
         \|/
        Site D
```

Core VPLS Concepts

1. **Virtual Switch Instance (VSI)**: Each PE router maintains a virtual switch for each VPLS instance. This VSI learns MAC addresses, makes forwarding decisions, and floods unknown unicast/broadcast/multicast traffic.
2. **Pseudowires**: VPLS uses a full mesh of pseudowires between all PE routers participating in the same VPLS instance. With N sites, you need N(N-1)/2 pseudowires.
3. **MAC Learning**: Just like a physical switch, VPLS learns MAC addresses dynamically from incoming frames and builds a MAC forwarding table.

4. **Split Horizon**: To prevent loops, VPLS implements split horizon—traffic received from one pseudowire is never forwarded to another pseudowire, only to local attachment circuits.

BGP's Role in VPLS

BGP serves as the control plane for VPLS, performing two critical functions:

1. **Auto-discovery**: BGP automatically discovers all PE routers participating in a VPLS instance
2. **Signaling**: BGP exchanges the necessary information to establish pseudowires

Here's how BGP-signaled VPLS works:

```
PE1 BGP VPLS Advertisement:
┌──────────────────────────────┐
│ Route Distinguisher: 1:100   │
│ VPLS-ID: 100                 │
│ Site-ID: 1                   │
│ Label Block Offset: 0        │
│ Label Block Size: 8          │
│ Label Base: 100000           │
│ Route Target: 65000:100      │
└──────────────────────────────┘

This tells other PEs:
- "I'm PE1 with Site-ID 1"
- "I'm part of VPLS 100"
- "Use labels 100000-100007 to reach me"
- "I can support up to 8 remote sites"
```

The Label Block Concept

BGP VPLS uses a clever **label block** mechanism to scale efficiently:

- Each PE allocates a contiguous block of MPLS labels
- The block size determines the maximum number of remote PEs supported
- Labels are computed using: `Label = Label_Base + Remote_Site_ID - Label_Block_Offset`

For example, if PE1 advertises:

- Label Base: 100000
- Label Block Offset: 0
- Label Block Size: 8

Then PE2 (Site-ID 2) would use label 100002 to send traffic to PE1.

Data Plane Operation

When a frame enters VPLS:

```
CE1 → PE1: Ethernet Frame (DA: 00:11:22:33:44:55)
           ↓
PE1 VSI: Check MAC table
         - Known MAC? → Forward to specific pseudowire
         - Unknown MAC? → Flood to all pseudowires + local ACs
           ↓
PE1 → MPLS Core: [MPLS Label][Control Word][Ethernet Frame]
           ↓
PE2: Receive frame, learn source MAC, forward to CE2
```

Why BGP for VPLS?

BGP offers several advantages for VPLS signaling:

1. **Scalability**: BGP's route reflection eliminates the need for full-mesh iBGP sessions

2. **Policy Control**: Route targets provide flexible membership control
3. **Multi-AS Support**: BGP naturally supports inter-AS VPLS deployments
4. **Convergence**: BGP's path selection and withdrawal mechanisms enable fast convergence

Part 2: The Junos CLI Masterclass (The How)

Understanding the Junos VPLS Hierarchy

In Junos, VPLS configuration lives under the `[edit routing-instances]` hierarchy because VPLS is fundamentally a type of routing instance—specifically, instance-type `vpls`.

```
[edit routing-instances vpls-name]
├── instance-type vpls
├── interface              # Local attachment circuits
├── route-distinguisher    # Unique identifier for BGP
├── vrf-target            # Route target for membership
├── protocols {
│   └── vpls {
│       ├── site-id       # Local site identifier
│       ├── site-range    # Maximum number of sites
│       └── no-tunnel-services  # Optional optimization
│   }
└── }
```

Step-by-Step BGP VPLS Configuration

Let's build a complete BGP-signaled VPLS connecting three sites:

```
Network Topology:
CE1 ─── PE1 ═══════════════ PE2 ─── CE2
         │                   │
         │    MPLS Core      │
         │                   │
         ╚═══════ PE3 ═══════╝
                  │
                 CE3

VPLS 100: Corporate LAN Service
- PE1: Site-ID 1, Interface ge-0/0/0.100
- PE2: Site-ID 2, Interface ge-0/0/0.100
- PE3: Site-ID 3, Interface ge-0/0/0.100
```

Step 1: Configure the Attachment Circuit

First, configure the customer-facing interface with proper encapsulation:

```
[edit interfaces ge-0/0/0]
unit 100 {
    description "Customer ABC - VPLS 100";
    encapsulation vlan-vpls;  # Critical for VPLS
    vlan-id 100;              # Match customer VLAN
    family vpls;              # Required for VPLS
}
```

Why these commands?

- `encapsulation vlan-vpls`: Tells Junos to expect VLAN-tagged frames for VPLS processing
- `vlan-id 100`: Identifies which VLAN tags to accept (single-tagged frames with VLAN 100)
- `family vpls`: Enables VPLS processing on this logical interface

Step 2: Configure BGP for VPLS NLRI

Enable BGP to carry VPLS routes:

```
[edit protocols bgp group ibgp]
family l2vpn {
    signaling;  # Enable L2VPN signaling capability
}
```

Step 3: Create the VPLS Routing Instance

Now, configure the complete VPLS instance on PE1:

```
[edit routing-instances VPLS-100]
instance-type vpls;
description "Corporate LAN Service - Customer ABC";
interface ge-0/0/0.100;  # Bind the attachment circuit

route-distinguisher 10.1.1.1:100;  # PE-Loopback:VPLS-ID
vrf-target target:65000:100;       # ASN:VPLS-ID

protocols {
    vpls {
        site-id 1;        # Unique within this VPLS
        site-range 10;    # Support up to 10 sites
        mac-table-size {
            4096;         # Maximum MACs to learn
        }
    }
}
```

Configuration Pattern Insights:

1. **Route Distinguisher Pattern**: Use `PE-Loopback:VPLS-ID` for easy identification
2. **VRF Target Pattern**: Use `ASN:VPLS-ID` for simple, consistent targeting
3. **Site-ID Rules**:
   - Must be unique within the VPLS
   - Valid range: 1 to (site-range - 1)
   - Cannot be 0 or equal to site-range

Step 4: Enable Label Allocation

VPLS requires MPLS labels from the dynamic range:

```
[edit protocols mpls]
label-range {
    dynamic-label-range 100000 199999;
}
```

Complete Configuration Example

Here's the full configuration for PE1:

```
## MPLS and LDP Configuration
protocols {
    mpls {
        label-range {
            dynamic-label-range 100000 199999;
        }
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

```
## BGP Configuration
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 10.1.1.1;
            family l2vpn {
                signaling;
            }
            neighbor 10.2.2.2;  # PE2
            neighbor 10.3.3.3;  # PE3
        }
    }
}

## Customer Interface
interfaces {
    ge-0/0/0 {
        unit 100 {
            description "Customer ABC - VPLS 100";
            encapsulation vlan-vpls;
            vlan-id 100;
            family vpls;
        }
    }
}

## VPLS Instance
routing-instances {
    VPLS-100 {
        instance-type vpls;
        description "Corporate LAN Service - Customer ABC";
        interface ge-0/0/0.100;
        route-distinguisher 10.1.1.1:100;
        vrf-target target:65000:100;
        protocols {
            vpls {
                site-id 1;
                site-range 10;
                mac-table-size {
                    4096;
                }
                no-tunnel-services;  # Optimize if not using tunnel interfaces
            }
        }
    }
}
```

Advanced Configuration Options

```
[edit routing-instances VPLS-100 protocols vpls]
## MAC Management
mac-table-aging-time 600;      # Age out MACs after 10 minutes
interface ge-0/0/0.100 {
    interface-mac-limit 1000;  # Limit MACs learned per interface
}

## Control Word (Strongly Recommended)
control-word;                  # Add control word to all pseudowires

## MTU Considerations
mtu 9000;                     # Ensure MTU matches across all sites
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify VPLS Instance Status

```
user@PE1> show vpls connections instance VPLS-100
```

Good Output:

```
Instance: VPLS-100
  VPLS-id: 100
  Number of local interfaces: 1
    Interface        MAC addresses

    ge-0/0/0.100     15
  Number of remote PEs: 2
    Remote-PE        Status     St  LD  RD
    10.2.2.2         Up         rmt 100002  100001
    10.3.3.3         Up         rmt 100003  100001

  Legend: St = State, LD = Local Site ID, RD = Remote Site ID
```

What to Look For:

- Status should be "Up" for all remote PEs
- Local and Remote Site IDs should be different
- Number of remote PEs should match your topology

2. Verify BGP VPLS Routes

```
user@PE1> show route table bgp.l2vpn.0 detail
```

Good Output:

```
bgp.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)

10.1.1.1:100:1:1/96 (1 entry, 1 announced)
        *L2VPN  Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Protocol next hop: 10.1.1.1
                Label operation: Push 100000, Push 299776(top)

10.2.2.2:100:2:1/96 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 10.2.2.2:100
                Protocol next hop: 10.2.2.2
                Label operation: Push 100000, Push 299792(top)
                Communities: target:65000:100
                VPLS-ID: 100 Site: 2 Label-base: 100000 Offset: 0 Range: 10
```

3. Verify MAC Learning

```
user@PE1> show vpls mac-table instance VPLS-100
```

Good Output:

```
MAC address        Interface         Type     Age
00:11:22:33:44:55  ge-0/0/0.100     Dynamic  0:05:23
00:aa:bb:cc:dd:ee  10.2.2.2         Dynamic  0:03:45
00:12:34:56:78:90  10.3.3.3         Dynamic  0:01:12
```

Common Troubleshooting Scenarios

Scenario 1: VPLS Connection Shows "Down"

Symptom:

```
user@PE1> show vpls connections
Instance: VPLS-100
  Remote-PE        Status
  10.2.2.2         Down (No BGP route)
```

Diagnostic Commands:

```
## Check if BGP route is received
user@PE1> show route receive-protocol bgp 10.2.2.2 table bgp.l2vpn.0

## Check BGP session status
user@PE1> show bgp neighbor 10.2.2.2

## Verify family l2vpn-signaling is negotiated
user@PE1> show bgp neighbor 10.2.2.2 | match "NLRI.*l2vpn"
  NLRI for restart configured on peer: inet-unicast l2vpn-signaling
```

Cause: BGP is not configured for l2vpn signaling or session is down

Solution:

```
[edit protocols bgp group ibgp]
set family l2vpn signaling
set neighbor 10.2.2.2

## Verify BGP export policy isn't filtering routes
[edit policy-options policy-statement vpls-export]
term 1 {
    from protocol bgp;
    then accept;
}
```

Scenario 2: Site-ID Conflict

Symptom:

```
user@PE1> show vpls connections
Instance: VPLS-100
  Remote-PE        Status
  10.2.2.2         Down (Site-ID collision)
```

Diagnostic Commands:

```
## Check local site-id
user@PE1> show vpls connections extensive | match "site-id"
  Local site ID: 1

## Check remote site information
user@PE1> show route table bgp.l2vpn.0 detail | match "Site:"
  VPLS-ID: 100 Site: 1 Label-base: 100000 Offset: 0 Range: 10
```

Cause: Two PEs are using the same Site-ID

Solution:

```
## Change site-id on one PE
[edit routing-instances VPLS-100 protocols vpls]
delete site-id 1
set site-id 2
```

Scenario 3: MAC Learning Issues

Symptom:

```
Pings work initially but fail after a few minutes
```

Diagnostic Commands:

```
## Check MAC table size
user@PE1> show vpls statistics instance VPLS-100
  MAC entries: 4096 (Limit reached)

## Check MAC moves
user@PE1> show vpls mac-move instance VPLS-100
  MAC               From-Interface    To-Interface      Count
  00:11:22:33:44:55 ge-0/0/0.100     10.2.2.2         523
```

Cause: MAC table full or MAC flapping

Solution:

```
## Increase MAC table size
[edit routing-instances VPLS-100 protocols vpls]
set mac-table-size 8192

## Enable MAC move protection
set mac-move-limit 5 actions log
```

Scenario 4: MTU Mismatch

Symptom:

```
Small pings work, large pings fail:
CE1# ping 192.168.1.2 size 1400  → Success
CE1# ping 192.168.1.2 size 1500  → Fail
```

Diagnostic Commands:

```
## Check VPLS MTU
user@PE1> show vpls connections extensive instance VPLS-100 | match mtu
  MTU: 1500

## Check MPLS MTU
user@PE1> show interfaces ge-0/0/1 | match MTU
  Link-level type: Ethernet, MTU: 1514, MRU: 1522
  Protocol inet, MTU: 1500
  Protocol mpls, MTU: 1488  ← Too small!
```

Cause: MPLS MTU too small to accommodate VPLS overhead

Solution:

```
## Increase interface MTU (on all core interfaces)
[edit interfaces ge-0/0/1]
set mtu 9192

## Set VPLS MTU explicitly
[edit routing-instances VPLS-100 protocols vpls]
set mtu 9000
```

Pro Troubleshooting Tips

1. **Always check both directions**: VPLS requires bidirectional connectivity
2. **Verify label allocation**: Use `show route table mpls.0` to ensure labels are programmed
3. **Monitor MAC moves**: Excessive MAC moves indicate loops or misconfigurations
4. **Use MAC ping**: `ping vpls instance VPLS-100 mac 00:11:22:33:44:55` tests the data plane
5. **Check for hidden routes**: `show route table bgp.l2vpn.0 hidden detail` reveals import policy issues

Remember: VPLS troubleshooting follows a logical flow:

1. Control plane (BGP routes) →

2. Label allocation →
3. Pseudowire establishment →
4. Data plane (MAC learning and forwarding)

Module 14: VPLS—LDP and FEC 129 Configuration and Verification

Part 1: The Conceptual Lecture (The Why)

The Evolution of VPLS Signaling

While BGP-signaled VPLS offers excellent scalability and policy control, it requires BGP infrastructure and knowledge. Some service providers wanted a simpler approach, leveraging their existing LDP infrastructure. This led to two LDP-based VPLS signaling methods:

1. **Traditional LDP VPLS** (RFC 4762)
2. **FEC 129 VPLS** (RFC 4447 with BGP Auto-discovery)

Understanding LDP-Signaled VPLS

LDP VPLS uses targeted LDP sessions to signal pseudowires between PE routers. Unlike BGP VPLS which uses route targets for auto-discovery, traditional LDP VPLS requires manual neighbor configuration.

```
LDP VPLS Architecture:

PE1 ←——— Targeted LDP Session ———→ PE2
 ↑                                  ↑
 |         Full Mesh Required       |
 |                                  |
 ↓                                  ↓
PE3 ←——— Targeted LDP Session ———→ PE4

Each PE must be manually configured with every other PE
```

Core Concepts of LDP VPLS

1. **Targeted LDP Sessions**: Unlike regular LDP that runs on directly connected interfaces, VPLS uses targeted sessions between PE loopbacks.
2. **FEC Types**:
   - **FEC 128** (PWid FEC): Traditional method using VC-ID
   - **FEC 129** (Generalized PWid FEC): Enhanced method with more flexibility
3. **Manual Configuration**: Every PE must explicitly list every other PE—no auto-discovery.
4. **Label Distribution**: Each PE allocates labels for incoming traffic from each remote PE.

How LDP VPLS Works

The signaling process follows these steps:

```
LDP VPLS Signaling Flow:

1. PE1 Configuration:
   neighbor 10.2.2.2 (PE2)
   vpls-id 100

2. LDP Label Mapping Message PE1→PE2:
   ┌──────────────────────────┐
   │ FEC: VPLS-ID 100         │
   │ VC Type: Ethernet        │
   │ Group ID: 0              │
   │ VC ID: 100               │
   │ Label: 300100            │
   │ MTU: 1500                │
   └──────────────────────────┘
```

```
3. PE2 responds with its label:
   Label Mapping PE2→PE1: Label 300200

4. Bidirectional pseudowire established:
   PE1→PE2: Use label 300200
   PE2→PE1: Use label 300100
```

FEC 129: The Hybrid Approach

FEC 129 combines the best of both worlds:

- **BGP for Auto-discovery**: Eliminates manual neighbor configuration
- **LDP for Signaling**: Simpler than full BGP VPLS

```
FEC 129 Operation:

1. BGP Auto-discovery Phase:
   PE1 ──BGP L2VPN Route──→ Route Reflector
                              ↓
   "I'm PE1, I have VPLS 100, talk to me at 10.1.1.1"

2. LDP Signaling Phase:
   PE2 receives BGP route, initiates targeted LDP to PE1
   PE2 ←──Targeted LDP──→ PE1

3. Result:
   Auto-discovered VPLS with LDP simplicity
```

Key Differences Between Signaling Methods

Feature	BGP VPLS	LDP VPLS	FEC 129
Auto-discovery	Yes (BGP)	No	Yes (BGP)
Signaling Protocol	BGP	LDP	LDP
Configuration Complexity	Medium	High (full mesh)	Low
Label Block	Yes	No	No
Site-ID Required	Yes	No	Optional
Scalability	Excellent	Poor	Good

When to Use Each Method

LDP VPLS:

- Small deployments (< 10 sites)
- No BGP infrastructure
- Simple, static topologies

FEC 129:

- Medium deployments
- Want auto-discovery without full BGP VPLS complexity
- Existing LDP expertise

BGP VPLS:

- Large deployments
- Need policy control
- Multi-AS scenarios

Part 2: The Junos CLI Masterclass (The How)

LDP VPLS Configuration Hierarchy

LDP VPLS uses a different configuration structure than BGP VPLS:

```
[edit routing-instances vpls-name protocols vpls]
├── vpls-id                    # Unique VPLS identifier
├── mtu                        # Optional MTU setting
├── neighbor <PE-address> {    # Manual neighbor configuration
│   ├── encapsulation-type     # ethernet or vlan
│   ├── ignore-encapsulation-mismatch
│   ├── control-word           # Recommended
│   └── pseudowire-status-tlv  # For OAM
├── }
└── no-tunnel-services         # Optimization
```

Step-by-Step LDP VPLS Configuration

Using the same three-site topology as before:

Step 1: Enable Targeted LDP

First, configure LDP to accept targeted sessions:

```
[edit protocols ldp]
interface lo0.0;                    # Enable on loopback
targeted-hello {
    accept-from 10.0.0.0/8;        # Accept from other PEs
    hello-interval 5;               # Fast hellos
    hold-time 15;
}
```

Step 2: Configure the LDP VPLS Instance

On PE1, create the VPLS instance with manual neighbors:

```
[edit routing-instances VPLS-100-LDP]
instance-type vpls;
description "LDP-Signaled VPLS for Customer ABC";
interface ge-0/0/0.100;            # Same attachment circuit

protocols {
    vpls {
        vpls-id 100;               # Must match on all PEs
        mtu 1500;                  # Explicit MTU

        ## Configure each neighbor manually
        neighbor 10.2.2.2 {        # PE2
            encapsulation-type ethernet;
            control-word;          # Add control word
            pseudowire-status-tlv; # Enable status signaling
        }

        neighbor 10.3.3.3 {        # PE3
            encapsulation-type ethernet;
            control-word;
            pseudowire-status-tlv;
        }

        mac-table-size {
            4096;
        }
        no-tunnel-services;
    }
}
```

Important Configuration Notes:

1. **VPLS-ID**: Must be identical on all PEs (unlike Site-ID in BGP VPLS)
2. **Encapsulation-type**: Must match between neighbors
3. **Neighbor Order**: Doesn't matter, but must list all PEs

Step 3: Configure PE2 and PE3

On PE2:

```
[edit routing-instances VPLS-100-LDP protocols vpls]
vpls-id 100;
neighbor 10.1.1.1 {    # PE1
    encapsulation-type ethernet;
    control-word;
    pseudowire-status-tlv;
}
neighbor 10.3.3.3 {    # PE3
    encapsulation-type ethernet;
    control-word;
    pseudowire-status-tlv;
}
```

FEC 129 VPLS Configuration

FEC 129 adds BGP auto-discovery to LDP VPLS:

Step 1: Enable BGP for Auto-discovery

```
[edit protocols bgp group ibgp]
family l2vpn {
    auto-discovery-only;    # Only for discovery, not signaling
}
```

Step 2: Configure FEC 129 VPLS Instance

```
[edit routing-instances VPLS-100-FEC129]
instance-type vpls;
interface ge-0/0/0.100;

## BGP parameters for auto-discovery
route-distinguisher 10.1.1.1:100;
vrf-target target:65000:100;

protocols {
    vpls {
        vpls-id 100;

        ## FEC 129 specific configuration
        signaling-type fec129;

        ## Optional: Configure specific neighbor attributes
        neighbor-discovery {
            encapsulation-type ethernet;
            control-word;
            mtu 1500;
        }

        ## No manual neighbors needed!
        mac-table-size {
            4096;
        }
    }
}
```

Complete LDP VPLS Configuration Example

Here's a complete PE1 configuration for traditional LDP VPLS:

```
## LDP Configuration
protocols {
    ldp {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0;
        targeted-hello {
            accept-from 10.0.0.0/8;
            hello-interval 5;
            hold-time 15;
        }
        ## Explicit targeted neighbors (optional but recommended)
        session 10.2.2.2;
        session 10.3.3.3;
    }
}

## VPLS Instance
routing-instances {
    VPLS-100-LDP {
        instance-type vpls;
        description "LDP-Signaled VPLS";
        interface ge-0/0/0.100;

        protocols {
            vpls {
                vpls-id 100;
                mtu 1500;

                ## Full mesh of neighbors
                neighbor 10.2.2.2 {
                    encapsulation-type ethernet;
                    control-word;
                    pseudowire-status-tlv;
                }
                neighbor 10.3.3.3 {
                    encapsulation-type ethernet;
                    control-word;
                    pseudowire-status-tlv;
                }

                ## MAC management
                mac-table-size {
                    4096;
                }
                mac-table-aging-time 300;

                ## Optimization
                no-tunnel-services;
            }
        }
    }
}
```

Advanced LDP VPLS Features

```
[edit routing-instances VPLS-100-LDP protocols vpls]

## Mac-flush on topology change
mac-flush;

## Neighbor-specific MTU
neighbor 10.2.2.2 {
    mtu 9000;              # Override default MTU
    ## Ignore MTU mismatches (use with caution)
    ignore-mtu-mismatch;
}
```

```
## Static pseudowire labels (not recommended)
neighbor 10.3.3.3 {
    static {
        incoming-label 301000;
        outgoing-label 302000;
    }
}

## BFD for fast failure detection
neighbor 10.2.2.2 {
    oam {
        bfd-liveness-detection {
            minimum-interval 100;
            multiplier 3;
        }
    }
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify LDP VPLS Connections

```
user@PE1> show vpls connections instance VPLS-100-LDP
```

Good Output:

```
Instance: VPLS-100-LDP
  VPLS-id: 100
  Neighbor                Type  St     Time last up          # Up trans
  10.2.2.2 (vpls-id 100)    rmt   Up     Oct 23 10:15:23 2024           1
    Remote PE: 10.2.2.2, Negotiated control-word: Yes
    Incoming label: 301234, Outgoing label: 302345
    Negotiated PW status TLV: Yes
  10.3.3.3 (vpls-id 100)    rmt   Up     Oct 23 10:15:25 2024           1
    Remote PE: 10.3.3.3, Negotiated control-word: Yes
    Incoming label: 301235, Outgoing label: 303456
    Negotiated PW status TLV: Yes
```

2. Verify LDP Sessions

```
user@PE1> show ldp session
```

Good Output:

```
Address           State       Connection  Hold time  Adv. Mode
10.2.2.2          Operational Open          24       DU
10.3.3.3          Operational Open          24       DU

user@PE1> show ldp neighbor 10.2.2.2 detail
Address: 10.2.2.2
    Transport address: 10.2.2.2
    Session type: Targeted
    Hello type: Targeted
```

3. Verify FEC 129 Auto-discovery

```
user@PE1> show route table bgp.l2vpn.0 vpls-id 100
```

Good Output for FEC 129:

```
bgp.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)

10.1.1.1:100:FEC129:100/96 (1 entry, 1 announced)
        *L2VPN  Preference: 170/-101
                Route Distinguisher: 10.1.1.1:100
                PMSI: Type LDP-VPLS, label 0, flags 0x0

10.2.2.2:100:FEC129:100/96 (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Source: 10.2.2.2
                Communities: target:65000:100
```

Common Troubleshooting Scenarios

Scenario 1: LDP VPLS Stuck in "Initializing"

Symptom:

```
user@PE1> show vpls connections
Instance: VPLS-100-LDP
  Neighbor                Type  St
  10.2.2.2 (vpls-id 100)    rmt   Initializing
```

Diagnostic Commands:

```
## Check LDP session status
user@PE1> show ldp session 10.2.2.2
  No session found

## Check if targeted hellos are being sent
user@PE1> show ldp interface
Interface            Label space ID        Nbr count   Next hello
lo0.0                10.1.1.1:0                    0   2

## Check LDP database
user@PE1> show ldp database session 10.2.2.2
```

Cause: No targeted LDP session established

Solution:

```
## Enable targeted LDP acceptance
[edit protocols ldp]
set targeted-hello accept-from 10.0.0.0/8

## Explicitly configure session
set session 10.2.2.2

## Verify no firewall blocking TCP 646
[edit firewall family inet filter protect-re term allow-ldp]
set from protocol tcp
set from port 646
set then accept
```

Scenario 2: Encapsulation Type Mismatch

Symptom:

```
user@PE1> show vpls connections
Instance: VPLS-100-LDP
  Neighbor                Type  St
  10.2.2.2 (vpls-id 100)    rmt   EM -- Encaps mismatch
```

Diagnostic Commands:

```
## Check local encapsulation
user@PE1> show configuration routing-instances VPLS-100-LDP | match encap
    encapsulation-type ethernet;

## Check received LDP bindings
user@PE1> show ldp database | match -A5 "VC type"
  VC type: VLAN, MTU: 1500      ← Mismatch!
```

Cause: PE1 expects Ethernet, PE2 sending VLAN

Solution:

```
## Option 1: Change PE1 to match PE2
[edit routing-instances VPLS-100-LDP protocols vpls neighbor 10.2.2.2]
delete encapsulation-type ethernet
set encapsulation-type vlan

## Option 2: Ignore mismatch (only if you understand implications)
set ignore-encapsulation-mismatch
```

Scenario 3: FEC 129 Not Auto-discovering Neighbors

Symptom:

```
FEC 129 VPLS configured but no neighbors appear
```

Diagnostic Commands:

```
## Check BGP auto-discovery routes
user@PE1> show route advertising-protocol bgp 10.100.1.1 table bgp.l2vpn.0

## Verify BGP family negotiation
user@PE1> show bgp neighbor 10.100.1.1 | match "NLRI.*l2vpn"
  NLRI that we support: l2vpn-auto-discovery-only

## Check if routes are hidden
user@PE1> show route table bgp.l2vpn.0 hidden
```

Cause: BGP not configured for auto-discovery-only

Solution:

```
## Configure BGP correctly
[edit protocols bgp group ibgp]
set family l2vpn auto-discovery-only

## Ensure vrf-target is configured
[edit routing-instances VPLS-100-FEC129]
set vrf-target target:65000:100
```

Scenario 4: MTU Issues in LDP VPLS

Symptom:

```
Large frames dropped, fragmentation occurring
```

Diagnostic Commands:

```
## Check negotiated MTU
user@PE1> show vpls connections instance VPLS-100-LDP extensive
  Neighbor: 10.2.2.2
    Local MTU: 1500, Remote MTU: 9000, Negotiated MTU: 1500

## Monitor MTU exceeded drops
```

```
user@PE1> show vpls statistics instance VPLS-100-LDP
  Encapsulation MTU exceeded: 42358
```

Cause: MTU mismatch between neighbors

Solution:

```
## Set consistent MTU across all PEs
[edit routing-instances VPLS-100-LDP protocols vpls]
set mtu 9000

## Or ignore MTU mismatch if you must
[edit routing-instances VPLS-100-LDP protocols vpls neighbor 10.2.2.2]
set ignore-mtu-mismatch
```

Comparison: BGP vs LDP vs FEC 129 Troubleshooting

Issue Type	BGP VPLS	LDP VPLS	FEC 129
Discovery Problems	Check RT import/export	N/A - Manual config	Check BGP + RT
Signaling Issues	BGP route exchange	LDP session/binding	LDP binding after BGP
Label Problems	Site-ID calculation	Direct label exchange	Direct label exchange
Scaling Issues	Route reflector load	Full mesh sessions	RR + LDP sessions

Pro Tips for LDP VPLS

1. **Always use control-word**: Prevents payload misinterpretation
2. **Monitor LDP session count**: Each VPLS neighbor needs a session
3. **Use pseudowire-status-tlv**: Enables better OAM capabilities
4. **Plan your VPLS-IDs**: Use a consistent scheme across your network
5. **Document your full mesh**: LDP VPLS gets complex quickly

LDP VPLS Best Practices

```
## Template for LDP VPLS neighbor
neighbor <address> {
    encapsulation-type ethernet;
    control-word;              # Always enable
    pseudowire-status-tlv;     # For OAM
    mtu 9000;                  # Consistent MTU
    oam {
        bfd-liveness-detection {
            minimum-interval 100;
            multiplier 3;
        }
    }
}
```

Remember: LDP VPLS trades auto-discovery for signaling simplicity. Choose based on your network's needs and existing infrastructure.

Module 15: VPLS—The Default VLAN Mode

Part 1: The Conceptual Lecture (The Why)

The VLAN Challenge in VPLS

Imagine you're connecting multiple customer sites, each using different VLAN schemes:

- Site A: Uses VLAN 100 for Finance
- Site B: Uses VLAN 200 for Finance
- Site C: Uses no VLANs at all (untagged traffic)

How does VPLS handle this diversity? This is where VPLS VLAN modes come into play.

Understanding Bridge Domains

Before diving into VLAN modes, we must understand **bridge domains**. A bridge domain is a broadcast domain within a VPLS instance—essentially a virtual switch with its own MAC table.

```
Traditional Switch vs VPLS Bridge Domain:

Physical Switch:                VPLS Instance:
┌──────────────────┐            ┌──────────────────────┐
│  Single MAC      │            │  Bridge Domain 1     │
│  Address Table   │            │  ├─ MAC Table        │
│                  │            │  ├─ VLAN 100         │
│  All VLANs       │            │  └─ Flooding Domain  │
│  Share Table     │            │                      │
└──────────────────┘            │  Bridge Domain 2     │
                                │  ├─ MAC Table        │
                                │  ├─ VLAN 200         │
                                │  └─ Flooding Domain  │
                                └──────────────────────┘
```

The Four VLAN Modes

VPLS supports four VLAN modes to handle different customer requirements:

1. **Default VLAN Mode**: One VLAN = One VPLS instance
2. **VLAN-Aware Mode**: Multiple VLANs in one VPLS, each gets its own bridge domain
3. **VLAN-Normalizing Mode**: Strip/swap VLANs at ingress
4. **No-VLAN Mode**: All untagged traffic

Default VLAN Mode Deep Dive

Default VLAN mode is the simplest and most common deployment:

```
Default VLAN Mode Characteristics:
- One VPLS instance per VLAN
- VLAN tags are NOT transported across the MPLS core
- Each site must use the same VLAN ID
- Single bridge domain per VPLS instance
```

Here's how it works:

```
Traffic Flow in Default VLAN Mode:

CE1 → PE1:  [Ethernet Frame][VLAN 100]
            ↓ PE1 strips VLAN 100
PE1 → Core: [MPLS Label][Control Word][Ethernet Frame]
            ↓ No VLAN in core!
Core → PE2: [MPLS Label][Control Word][Ethernet Frame]
            ↓ PE2 adds VLAN 100
PE2 → CE2:  [Ethernet Frame][VLAN 100]
```

Why Default VLAN Mode?

Advantages:

1. **Simplicity**: One VLAN per VPLS is easy to understand
2. **Bandwidth Efficiency**: No VLAN tags in the core saves 4 bytes per frame
3. **Clear Separation**: Each VLAN gets dedicated resources
4. **Compatibility**: Works with all equipment

Limitations:

1. **Scalability**: Need separate VPLS instance for each VLAN
2. **Flexibility**: All sites must use the same VLAN ID
3. **Management**: Multiple VPLS instances to configure and monitor

Bridge Domain Behavior

In default VLAN mode, the bridge domain encompasses the entire VPLS instance:

```
Bridge Domain Scope:

┌─────────────────────────────────────┐
│         VPLS Instance 100           │
│  ┌─────────────────────────────┐    │
│  │     Single Bridge Domain    │    │
│  │                             │    │
│  │  Site A ←→ Site B ←→ Site C │    │
│  │   VLAN     VLAN      VLAN   │    │
│  │   100      100       100    │    │
│  │                             │    │
│  │    Shared MAC Table         │    │
│  └─────────────────────────────┘    │
└─────────────────────────────────────┘
```

MAC Learning in Default Mode

MAC addresses are learned without VLAN context:

```
MAC Table Entry:

┌────────────────────┬──────────────────┬───────┐
│ MAC Address        │ Interface        │ Age   │
├────────────────────┼──────────────────┼───────┤
│ 00:11:22:33:44:55 │ ge-0/0/0.100     │ 0:05  │
│ 00:aa:bb:cc:dd:ee │ 10.2.2.2(PW)     │ 0:03  │
└────────────────────┴──────────────────┴───────┘

Note: No VLAN column - VLANs stripped at PE
```

Part 2: The Junos CLI Masterclass (The How)

Default VLAN Mode Configuration

Default VLAN mode requires no special configuration—it's the default behavior!

Basic Configuration Structure

```
[edit interfaces ge-0/0/0]
unit 100 {
    description "Customer A - VLAN 100 Service";
    encapsulation vlan-vpls;      # Key: vlan-vpls
    vlan-id 100;                  # Local VLAN to match
    family vpls;                  # VPLS processing
}

[edit routing-instances VPLS-CUSTOMER-A-VLAN100]
instance-type vpls;
interface ge-0/0/0.100;           # One VLAN per instance
## NO vlan-mode configuration = Default mode
```

Complete Default VLAN Mode Example

Let's configure a VPLS for customer VLAN 100:

```
## Interface Configuration - PE1
interfaces {
    ge-0/0/0 {
```

```
        description "Customer A - Trunk to CE1";
        vlan-tagging;              # Enable VLAN tagging
        unit 100 {
            description "Customer A - Finance VLAN";
            encapsulation vlan-vpls;
            vlan-id 100;           # Match customer VLAN 100
            family vpls;
        }
        unit 200 {
            description "Customer A - Voice VLAN";
            encapsulation vlan-vpls;
            vlan-id 200;           # Different VLAN = Different VPLS
            family vpls;
        }
    }
}

## VPLS Instance for VLAN 100
routing-instances {
    VPLS-CUST-A-FINANCE {
        instance-type vpls;
        description "Customer A - Finance VLAN 100";
        interface ge-0/0/0.100;
        route-distinguisher 10.1.1.1:100;
        vrf-target target:65000:100;
        protocols {
            vpls {
                site-id 1;
                site-range 10;
                mac-table-size {
                    4096;
                }
                ## Note: No vlan configuration needed
            }
        }
    }

    ## Separate VPLS for VLAN 200
    VPLS-CUST-A-VOICE {
        instance-type vpls;
        description "Customer A - Voice VLAN 200";
        interface ge-0/0/0.200;
        route-distinguisher 10.1.1.1:200;
        vrf-target target:65000:200;
        protocols {
            vpls {
                site-id 1;
                site-range 10;
            }
        }
    }
}
```

Verifying Default VLAN Mode

1. Verify Bridge Domain

```
user@PE1> show vpls connections instance VPLS-CUST-A-FINANCE
```

Output shows single bridge domain:

```
Instance: VPLS-CUST-A-FINANCE
  Local interface: ge-0/0/0.100, Status: Up
    Bridge domain: __VPLS-CUST-A-FINANCE__    # Default bridge domain
    Number of local interfaces: 1
    Number of remote PEs: 2
```

2. Check VLAN Processing

```
user@PE1> show interfaces ge-0/0/0.100 detail | match vlan
  Logical interface ge-0/0/0.100
    VLAN-Tag [ 0x8100.100 ]  In(pop)  Out(push)
    ↑                         ↑        ↑
    VLAN 100 is:             Removed   Added back
```

3. Monitor MAC Table

```
user@PE1> show vpls mac-table instance VPLS-CUST-A-FINANCE

MAC address        Interface         Type     Age    Bridge domain
00:11:22:33:44:55  ge-0/0/0.100     Dynamic  0:02   __VPLS-CUST-A-FINANCE__
00:aa:bb:cc:dd:ee  10.2.2.2         Dynamic  0:05   __VPLS-CUST-A-FINANCE__
```

Understanding VLAN Tag Operations

To see what happens to VLAN tags:

```
## Capture on CE-facing interface
user@PE1> monitor traffic interface ge-0/0/0.100
15:23:45.123 In  VLAN100 00:11:22:33:44:55 > ff:ff:ff:ff:ff:ff, ARP request

## Capture on core-facing interface
user@PE1> monitor traffic interface ge-0/0/1.0
15:23:45.124 Out MPLS (label 299776, exp 0, ttl 255) (label 100001, exp 0, [S], ttl 255)
              00:11:22:33:44:55 > ff:ff:ff:ff:ff:ff, ARP request
              ↑ No VLAN tag in MPLS core!
```

Advanced Default Mode Features

MAC Limiting per Bridge Domain

```
[edit routing-instances VPLS-CUST-A-FINANCE protocols vpls]
bridge-domain {
    __VPLS-CUST-A-FINANCE__ {
        mac-table-size {
            2048;                  # Limit for this bridge domain
            packet-action drop;    # Drop when limit reached
        }
    }
}
```

Interface-Specific Features

```
[edit routing-instances VPLS-CUST-A-FINANCE protocols vpls]
interface ge-0/0/0.100 {
    interface-mac-limit {
        1000;                      # Per-interface MAC limit
        packet-action drop;
    }
    ## Storm control
    storm-control finance-profile;
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify VLAN Mode

```
user@PE1> show vpls instance VPLS-CUST-A-FINANCE extensive | match mode
  VLAN mode: Default              # Confirms default mode
```

2. Check Bridge Domain Details

```
user@PE1> show bridge domain

Bridge domain: __VPLS-CUST-A-FINANCE__  State: Active
    Interfaces:
      ge-0/0/0.100             Active
      lsi.1048576              Active    # Flood interface
    Remote PEs:
      10.2.2.2
      10.3.3.3
    Statistics:
      Total MAC count: 125
      Total MAC+IP count: 125
```

Common Troubleshooting Scenarios

Scenario 1: Customer Expects VLAN Tags Preserved

Symptom: Customer complains: "We're sending VLAN 100 from Site A, but Site B doesn't see the VLAN tag!"

Diagnostic Commands:

```
## Check encapsulation type
user@PE1> show configuration interfaces ge-0/0/0.100 | match encap
    encapsulation vlan-vpls;    # This strips VLANs!

## Verify VLAN operations
user@PE1> show interfaces ge-0/0/0.100 | match "VLAN-Tag.*pop"
    VLAN-Tag [ 0x8100.100 ]  In(pop)  Out(push)
```

Cause: Default VLAN mode strips VLAN tags—this is by design

Solution:

```
## Option 1: Educate customer (preferred)
"VPLS default mode provides VLAN transparency within the service.
 Site B will receive the same Ethernet frames, with VLAN 100
 added by the PE router."

## Option 2: Use VLAN-Aware mode if tags must be preserved
[edit routing-instances VPLS-CUST-A-FINANCE]
delete interface ge-0/0/0.100
set interface ge-0/0/0      # Trunk interface
set vlan-id all             # Preserve all VLANs
```

Scenario 2: VLAN Mismatch Between Sites

Symptom:

```
Site A uses VLAN 100, Site B uses VLAN 200 for the same service
Traffic doesn't flow between sites
```

Diagnostic Commands:

```
## Check remote site configuration
user@PE2> show configuration interfaces ge-0/0/0.200
    vlan-id 200;    # Different VLAN!

## Verify they're in same VPLS
user@PE2> show route table bgp.l2vpn.0 | match 65000:100
    Communities: target:65000:100    # Same RT, but...
```

Cause: Default mode requires same VLAN ID at all sites

Solution:

```
## Option 1: Standardize VLAN IDs across sites
[edit interfaces ge-0/0/0 unit 200]
delete vlan-id 200
set vlan-id 100        # Match other sites

## Option 2: Use VLAN normalization (covered in next module)
[edit routing-instances VPLS-CUST-A protocols vpls]
set vlan-id 100 normalize    # Advanced feature
```

Scenario 3: Bridge Domain Full

Symptom:

```
user@PE1> show vpls mac-table instance VPLS-CUST-A-FINANCE summary
Bridge Domain: __VPLS-CUST-A-FINANCE__
  Total MAC addresses: 4096
  Capacity: 100%    # Full!

New MACs not being learned, connectivity issues
```

Diagnostic Commands:

```
## Check MAC table size
user@PE1> show vpls instance extensive | match -A2 "mac-table"
  MAC table size: 4096
  MAC table usage: 4096 (100%)
  MAC+IP table size: 0

## Find top MAC learners
user@PE1> show vpls mac-table instance VPLS-CUST-A-FINANCE | count
Count: 4096 lines

user@PE1> show vpls mac-table instance VPLS-CUST-A-FINANCE | match ge-0/0/0.100 | count
Count: 3500 lines    # One interface learning too many!
```

Cause: MAC table exhausted, possibly due to loops or scanning

Solution:

```
## Increase table size
[edit routing-instances VPLS-CUST-A-FINANCE protocols vpls]
set mac-table-size 8192

## Implement MAC limiting per interface
set interface ge-0/0/0.100 interface-mac-limit 1000

## Enable MAC move limiting
set mac-move-limit 10 actions log block
```

Scenario 4: QinQ Customer Traffic

Symptom:

```
Customer is sending double-tagged (QinQ) traffic
Inner tags are being stripped
```

Diagnostic Commands:

```
## Check interface encapsulation
user@PE1> show interfaces ge-0/0/0.100 | match encap
  Encapsulation: VLAN-VPLS    # Only handles single VLAN!

## Monitor traffic to see tags
user@PE1> monitor traffic interface ge-0/0/0.100 no-resolve
00:11:22:33:44:55 > 00:aa:bb:cc:dd:ee, ethertype 802.1Q (0x8100),
```

```
   vlan 100, ethertype 802.1Q (0x8100), vlan 200: IP 10.1.1.1 > 10.2.2.2
   ↑ Customer QinQ traffic!
```

Cause: Default mode only processes single VLAN tag

Solution:

```
## Change to flexible-vlan-tagging for QinQ
[edit interfaces ge-0/0/0]
set flexible-vlan-tagging

[edit interfaces ge-0/0/0 unit 100]
delete encapsulation vlan-vpls
delete vlan-id 100
set encapsulation vlan-vpls
set vlan-tags outer 100    # or use dual-tags (Module 16)
```

Best Practices for Default VLAN Mode

1. **Naming Convention**: Include VLAN in instance name

   ```
   VPLS-{CUSTOMER}-{SERVICE}-VLAN{ID}
   Example: VPLS-ACME-FINANCE-VLAN100
   ```

2. **Documentation**: Always document VLAN mappings

   ```
   [edit routing-instances VPLS-CUST-A-FINANCE]
   set description "Customer A Finance - All sites use VLAN 100"
   ```

3. **Monitoring**: Set up VLAN-specific monitoring

   ```
   set snmp rmon alarm 1 variable jnxVpls.VPLS-CUST-A-FINANCE.mac-count
   ```

4. **Scalability Planning**: Consider VLAN-aware mode if >10 VLANs per customer

Default Mode Design Considerations

Aspect	Consideration
VLAN Consistency	All sites must use same VLAN ID
Scaling	One VPLS instance per VLAN
Operations	Simple but potentially many instances
Bandwidth	Efficient (no VLAN in core)
Flexibility	Limited (no VLAN translation)

Remember: Default VLAN mode is perfect for simple deployments where VLAN consistency is maintained across all sites. For more complex requirements, consider the other VLAN modes covered in the next module.

Module 16: VPLS—VLAN Normalization, VLAN-Aware Instances, and Dual-Stacked VLANs

Part 1: The Conceptual Lecture (The Why)

Beyond Default VLAN Mode

While default VLAN mode works well for simple deployments, real-world networks often present more complex challenges:

- **Multi-VLAN Customers**: A customer using 50 VLANs shouldn't require 50 VPLS instances
- **VLAN Inconsistency**: Different sites using different VLAN IDs for the same service
- **Service Provider VLANs**: Carriers using QinQ to separate customers

- **Legacy Sites**: Locations with untagged traffic needing integration

This is where advanced VLAN modes come into play.

VLAN-Aware Mode: Multiple Bridge Domains

VLAN-Aware mode fundamentally changes how VPLS handles VLANs:

```
Default Mode vs VLAN-Aware Mode:

Default Mode:                    VLAN-Aware Mode:
┌──────────────────┐             ┌──────────────────────────┐
│ VPLS-VLAN100     │             │ VPLS-CUSTOMER-A          │
│ ├─ Bridge Domain │             │ ├─ Bridge Domain VLAN100 │
│ └─ VLAN 100 only │             │ ├─ Bridge Domain VLAN200 │
├──────────────────┤             │ ├─ Bridge Domain VLAN300 │
│ VPLS-VLAN200     │             │ └─ All in one instance!  │
│ ├─ Bridge Domain │             └──────────────────────────┘
│ └─ VLAN 200 only │
├──────────────────┤             One instance handles all
│ VPLS-VLAN300     │             customer VLANs with separate
│ ├─ Bridge Domain │             forwarding per VLAN
│ └─ VLAN 300 only │
└──────────────────┘
```

Key characteristics:

- **VLAN tags preserved** across the MPLS core
- **Separate bridge domain** per VLAN
- **Independent MAC tables** per VLAN
- **Single VPLS instance** for management

VLAN-Normalizing Mode: Solving VLAN Chaos

VLAN normalization addresses the common problem of VLAN ID inconsistency:

```
VLAN Normalization in Action:

Site A (VLAN 100) ─┐
                   ├─→ VPLS Core (Normalized to VLAN 500) ─→ Site C (VLAN 300)
Site B (VLAN 200) ─┘

- PE routers translate local VLANs to a common "normalized" VLAN
- Core sees consistent VLAN across all sites
- Each PE translates back to local VLAN
```

Use cases:

1. **Mergers**: Companies with different VLAN schemes
2. **Legacy Integration**: Sites that can't change VLAN IDs
3. **Service Abstraction**: Hide internal VLAN complexity from customers

No-VLAN Mode: Untagged Traffic

No-VLAN mode handles untagged Ethernet frames:

```
No-VLAN Mode Operation:
CE → PE: [Ethernet Frame]           (No VLAN tag)
         ↓
PE → Core: [MPLS][CW][Ethernet Frame]  (Still no VLAN)
         ↓
Core → PE: [MPLS][CW][Ethernet Frame]
```

```
         ↓
PE → CE: [Ethernet Frame]           (No VLAN tag)
```

Perfect for:

- Legacy equipment without VLAN support
- Point-of-sale terminals
- Simple Layer 2 extensions

Dual-Stacked VLANs: QinQ Support

Dual-stacked (QinQ) support handles service provider deployments:

```
QinQ Structure:
[Ethernet Frame][S-VLAN][C-VLAN][Payload]
                   ↑        ↑
            Service VLAN  Customer VLAN
            (Outer tag)   (Inner tag)

Example:
ISP assigns S-VLAN 1000 to customer
Customer uses C-VLANs 100, 200, 300 internally
```

VPLS can:

1. **Transport both tags** transparently
2. **Terminate outer tag** and process inner tags
3. **Manipulate either tag** independently

Part 2: The Junos CLI Masterclass (The How)

Configuring VLAN-Aware Mode

VLAN-Aware mode requires specific configuration to enable multiple bridge domains:

Step 1: Configure Trunk Interface

```
[edit interfaces ge-0/0/0]
description "Customer A - Trunk for VLAN-Aware VPLS";
vlan-tagging;                    # Must be trunk
encapsulation flexible-ethernet-services;  # Key difference!
unit 0 {
    family vpls;                 # No VLAN specified here
}
```

Step 2: Create VLAN-Aware VPLS Instance

```
[edit routing-instances VPLS-CUSTOMER-A]
instance-type vpls;
interface ge-0/0/0.0;            # Trunk interface

## Define which VLANs to handle
vlan-id [ 100 200 300 ];         # List of VLANs
## OR
vlan-id all;                     # Handle all VLANs

route-distinguisher 10.1.1.1:1000;
vrf-target target:65000:1000;

protocols {
    vpls {
        site-id 1;
        site-range 10;
    }
}
```

Step 3: Verify Bridge Domain Creation

```
user@PE1> show bridge domain

Bridge domain: VLAN-100    State: Active
    VPLS instance: VPLS-CUSTOMER-A
    Interfaces:
      ge-0/0/0.0 (100)

Bridge domain: VLAN-200    State: Active
    VPLS instance: VPLS-CUSTOMER-A
    Interfaces:
      ge-0/0/0.0 (200)

Bridge domain: VLAN-300    State: Active
    VPLS instance: VPLS-CUSTOMER-A
    Interfaces:
      ge-0/0/0.0 (300)
```

Configuring VLAN Normalization

VLAN normalization requires mapping local VLANs to a normalized VLAN:

Basic Normalization Configuration

```
[edit routing-instances VPLS-NORMALIZED]
instance-type vpls;

## Define normalized VLAN
vlan-id 500;                    # All sites normalize to VLAN 500

## Map local VLANs to normalized VLAN
interface ge-0/0/0.100 {        # Local VLAN 100
    vlan-id 500;                # Normalize to 500
}

interface ge-0/0/1.200 {        # Local VLAN 200
    vlan-id 500;                # Also normalize to 500
}

protocols {
    vpls {
        site-id 1;
        site-range 10;
    }
}
```

Advanced Normalization with Translation

```
## Interface configuration for normalization
[edit interfaces ge-0/0/0]
unit 100 {
    encapsulation vlan-vpls;
    vlan-id 100;
    family vpls;
}

## VPLS configuration with vlan-maps
[edit routing-instances VPLS-TRANSLATE]
instance-type vpls;
interface ge-0/0/0.100;

## Bridge domain configuration
bridge-domains {
    CUSTOMER-LAN {
        vlan-id 500;            # Normalized VLAN
        interface ge-0/0/0.100 {
            vlan-id-list 100;   # Accept VLAN 100
            ## Implicit translation: 100 → 500
```

```
        }
    }
}
```

Configuring No-VLAN Mode

No-VLAN mode handles untagged traffic:

```
## Interface without VLAN tagging
[edit interfaces ge-0/0/0]
unit 0 {
    encapsulation ethernet-vpls;  # Not vlan-vpls!
    family vpls;
}

## VPLS instance configuration
[edit routing-instances VPLS-UNTAGGED]
instance-type vpls;
interface ge-0/0/0.0;

## Explicitly set no-vlan mode
vlan-id none;                    # Handle untagged only

protocols {
    vpls {
        site-id 1;
        site-range 10;
    }
}
```

Configuring Dual-Stacked VLANs (QinQ)

For QinQ support, use flexible-vlan-tagging:

Basic QinQ Configuration

```
## Enable QinQ on interface
[edit interfaces ge-0/0/0]
flexible-vlan-tagging;
vlan-tagging;
encapsulation flexible-ethernet-services;

## Configure dual-tagged unit
unit 1000 {
    vlan-tags outer 1000 inner 100;  # S-VLAN 1000, C-VLAN 100
    family vpls;
}

## VPLS instance
[edit routing-instances VPLS-QINQ]
instance-type vpls;
interface ge-0/0/0.1000;
## VLANs are preserved as-is
```

QinQ with Inner VLAN Range

```
[edit interfaces ge-0/0/0]
unit 1000 {
    vlan-tags outer 1000 inner-range 100-199;  # S-VLAN + C-VLAN range
    family vpls;
}

## This creates a VLAN-aware instance for inner VLANs
[edit routing-instances VPLS-QINQ-AWARE]
instance-type vpls;
interface ge-0/0/0.1000;
vlan-id all;                     # Process all inner VLANs separately
```

Complete Advanced VLAN Configuration Example

Here's a comprehensive example combining multiple modes:

```
## Interfaces
interfaces {
    ## VLAN-Aware customer trunk
    ge-0/0/0 {
        description "Customer A - Multi-VLAN Trunk";
        vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family vpls;
        }
    }

    ## Normalization interfaces
    ge-0/0/1 {
        vlan-tagging;
        unit 100 {
            description "Site B - Uses VLAN 100";
            encapsulation vlan-vpls;
            vlan-id 100;
            family vpls;
        }
        unit 200 {
            description "Site C - Uses VLAN 200";
            encapsulation vlan-vpls;
            vlan-id 200;
            family vpls;
        }
    }

    ## QinQ interface
    ge-0/0/2 {
        flexible-vlan-tagging;
        vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 1000 {
            description "Service Provider QinQ";
            vlan-tags outer 1000 inner-range 100-199;
            family vpls;
        }
    }
}

## VPLS Instances
routing-instances {
    ## VLAN-Aware Instance
    VPLS-VLAN-AWARE {
        instance-type vpls;
        interface ge-0/0/0.0;
        vlan-id [ 100 200 300 ];  # Specific VLANs
        route-distinguisher 10.1.1.1:2000;
        vrf-target target:65000:2000;
        protocols {
            vpls {
                site-id 1;
                site-range 10;
                ## Per-VLAN MAC limits
                bridge-domain VLAN-100 {
                    mac-table-size {
                        2048;
                    }
                }
            }
        }
    }

    ## Normalized VPLS
    VPLS-NORMALIZED {
```

```
        instance-type vpls;
        vlan-id 500;              # Normalized VLAN
        interface ge-0/0/1.100 {
            vlan-id 500;          # Map 100→500
        }
        interface ge-0/0/1.200 {
            vlan-id 500;          # Map 200→500
        }
        route-distinguisher 10.1.1.1:3000;
        vrf-target target:65000:3000;
        protocols {
            vpls {
                site-id 1;
                site-range 10;
            }
        }
    }
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify VLAN Mode Type

```
user@PE1> show vpls instance extensive | match "vlan|VLAN"
  Instance: VPLS-VLAN-AWARE
    VLAN mode: VLAN-aware
    VLAN ID: 100, 200, 300

  Instance: VPLS-NORMALIZED
    VLAN mode: VLAN-normalizing
    VLAN ID: 500
```

2. Check Bridge Domain Mapping

```
user@PE1> show bridge domain detail

Bridge domain: VLAN-100
    VPLS instance: VPLS-VLAN-AWARE
    VLAN ID: 100
    Interfaces:
        ge-0/0/0.0
    Statistics:
        Packets: 1234567
        Bytes: 123456789

Bridge domain: VLAN-200
    VPLS instance: VPLS-VLAN-AWARE
    VLAN ID: 200
    Interfaces:
        ge-0/0/0.0
```

3. Verify VLAN Translation

```
user@PE1> show vpls connections instance VPLS-NORMALIZED extensive
  Interface: ge-0/0/1.100
    Local VLAN: 100
    Normalized VLAN: 500
    VLAN translation: 100->500 (ingress), 500->100 (egress)
```

Common Troubleshooting Scenarios

Scenario 1: VLAN-Aware Mode Not Creating Bridge Domains

Symptom:

```
user@PE1> show bridge domain
## No bridge domains shown for VLAN-aware VPLS
```

Diagnostic Commands:

```
## Check interface encapsulation
user@PE1> show configuration interfaces ge-0/0/0
encapsulation vlan-vpls;    # Wrong! Should be flexible-ethernet-services

## Verify VLAN configuration
user@PE1> show configuration routing-instances VPLS-VLAN-AWARE | match vlan
## No vlan-id configured
```

Cause: Wrong encapsulation or missing VLAN configuration

Solution:

```
## Fix interface encapsulation
[edit interfaces ge-0/0/0]
delete encapsulation vlan-vpls
set encapsulation flexible-ethernet-services

## Add VLAN configuration
[edit routing-instances VPLS-VLAN-AWARE]
set vlan-id [ 100 200 300 ]
```

Scenario 2: VLAN Normalization Not Working

Symptom:

```
Sites with different VLANs can't communicate
PE2 receives frames with wrong VLAN
```

Diagnostic Commands:

```
## Check normalization config
user@PE1> show configuration routing-instances VPLS-NORMALIZED
interface ge-0/0/1.100;    # Missing vlan-id statement!

## Monitor traffic
user@PE1> monitor traffic interface ge-0/0/1.100
## Shows VLAN 100 entering
user@PE1> monitor traffic interface ge-0/0/2.0
## Shows VLAN 100 leaving (not normalized!)
```

Cause: Missing vlan-id normalization statement

Solution:

```
[edit routing-instances VPLS-NORMALIZED]
set interface ge-0/0/1.100 vlan-id 500

## Alternative: Use bridge-domains
bridge-domains {
    NORMALIZED {
        vlan-id 500;
        interface ge-0/0/1.100 {
            vlan-id-list 100;    # Translate 100->500
        }
    }
}
```

Scenario 3: QinQ Inner VLANs Not Isolated

Symptom:

```
Customer complains about cross-VLAN traffic leakage
Different C-VLANs can see each other's broadcasts
```

Diagnostic Commands:

```
## Check VLAN mode
user@PE1> show vpls instance VPLS-QINQ extensive | match mode
  VLAN mode: Default    # Wrong! Should be VLAN-aware for isolation

## Check bridge domains
user@PE1> show bridge domain
Bridge domain: __VPLS-QINQ__
  ## Only one bridge domain for all inner VLANs!
```

Cause: QinQ interface in default mode merges all inner VLANs

Solution:

```
## Change to VLAN-aware mode for inner VLAN isolation
[edit routing-instances VPLS-QINQ]
set vlan-id all    # Process inner VLANs separately

## Or specify specific inner VLANs
set vlan-id [ 100 101 102 ]
```

Scenario 4: MAC Table Explosion in VLAN-Aware Mode

Symptom:

```
user@PE1> show system alarms
Class  Description
Major  MAC table usage exceeded threshold - Bridge domain VLAN-100
Major  MAC table usage exceeded threshold - Bridge domain VLAN-200
```

Diagnostic Commands:

```
## Check per-VLAN MAC usage
user@PE1> show bridge mac-table count
Bridge Domain          MAC Count   MAC+IP Count
VLAN-100               4096        0
VLAN-200               4096        0
VLAN-300               1523        0

## Find source of MAC explosion
user@PE1> show vpls mac-table instance VPLS-VLAN-AWARE bridge-domain VLAN-100 | match 00:11:22
## Shows many sequential MACs - possible scanning
```

Cause: Each bridge domain has its own MAC limit

Solution:

```
## Set per-bridge-domain limits
[edit routing-instances VPLS-VLAN-AWARE protocols vpls]
bridge-domain VLAN-100 {
    mac-table-size {
        8192;              # Increase limit
    }
    interface ge-0/0/0.0 {
        interface-mac-limit 1000;  # Limit per interface
        storm-control default;      # Enable storm control
    }
}
```

Advanced Troubleshooting Tools

VLAN Mapping Verification Script

```
## Operational mode script to verify VLAN mappings
user@PE1> op vlan-mapping-check
VPLS Instance: VPLS-NORMALIZED
  Interface        Local-VLAN   Normalized-VLAN   Status
  ge-0/0/1.100     100          500               Active
  ge-0/0/1.200     200          500               Active
  ge-0/0/2.300     300          500               Active
```

Bridge Domain Statistics

```
## Monitor per-VLAN statistics
user@PE1> monitor interface traffic matching "bridge|VLAN"

## Check for VLAN-specific drops
user@PE1> show interfaces statistics | match "VLAN.*drop"
```

Best Practices for Advanced VLAN Modes

1. **VLAN-Aware Mode**:
   - Use for customers with many VLANs
   - Monitor per-VLAN MAC table usage
   - Consider `vlan-id all` carefully (security implications)
2. **VLAN Normalization**:
   - Document all VLAN mappings clearly
   - Test bidirectional translation
   - Use consistent normalized VLAN across all PEs
3. **QinQ Deployments**:
   - Decide if inner VLANs need isolation
   - Plan S-VLAN allocation carefully
   - Consider MTU (QinQ adds 8 bytes)
4. **General Guidelines**:

```
## Always set explicit MTU for QinQ
[edit routing-instances VPLS-QINQ protocols vpls]
set mtu 9100    # Account for dual tags

## Use descriptions liberally
set interface ge-0/0/1.100 description "Site-B VLAN-100 -> Normalized-500"
```

VLAN Mode Decision Matrix

Scenario	Recommended Mode	Key Configuration
Single VLAN per VPLS	Default	`encapsulation vlan-vpls`
Multiple VLANs, keep separate	VLAN-Aware	`vlan-id [ list ]`
Different VLANs between sites	Normalizing	`interface x vlan-id y`
Untagged traffic only	No-VLAN	`vlan-id none`
Service provider QinQ	Dual-stack	`vlan-tags outer X inner Y`

Remember: Choose the VLAN mode that matches your operational model. Complexity should serve a purpose—don't use advanced modes unless they solve a real problem.

Module 17: VPLS—Advanced Features and Troubleshooting

Part 1: The Conceptual Lecture (The Why)

The Need for Advanced VPLS Features

As VPLS deployments grow, operators face challenges that basic configurations can't address:

- **Site-ID Management**: Manually tracking Site-IDs across hundreds of PEs becomes error-prone
- **Security Threats**: MAC flooding attacks can destabilize the entire VPLS
- **Layer 3 Integration**: Customers want routing capabilities within their Layer 2 domain
- **Bandwidth Efficiency**: Broadcast/multicast flooding wastes core bandwidth
- **Operational Complexity**: Troubleshooting requires sophisticated tools

These challenges drove the development of advanced VPLS features.

Automated BGP VPLS Site IDs

Manual Site-ID assignment creates operational headaches:

```
The Site-ID Problem:

Network with 50 PE routers in VPLS 100:
- PE1: site-id 1  ✓
- PE2: site-id 2  ✓
- PE3: site-id 3  ✓
...
- PE49: site-id 49  ✓
- PE50: site-id 50  ✗ Oops! Someone already used 50!

Human errors are inevitable at scale
```

Automated Site-ID allocation solves this by:

1. Using the PE's router-id or loopback IP
2. Automatically deriving a unique Site-ID
3. Eliminating configuration conflicts

Protection Mechanisms

VPLS faces three main security threats:

1. MAC Flooding Attacks

```
Attack Scenario:
Attacker generates thousands of unique source MACs
     ↓
VPLS learns all MACs → MAC table fills
     ↓
Legitimate MACs age out → Flooding increases
     ↓
Network performance degrades severely
```

2. MAC Flapping

```
MAC Flap Scenario:
Same MAC appears on multiple interfaces rapidly

Time 10:00:00 – MAC X seen on PE1
Time 10:00:01 – MAC X seen on PE2
Time 10:00:02 – MAC X seen on PE1
Time 10:00:03 – MAC X seen on PE2
(Continues indefinitely)

Causes: Loops, duplicate MACs, or misconfigurations
```

3. Broadcast Storms

```
Storm Propagation in VPLS:
Broadcast at Site A → Flooded to all sites
                    → Each site may amplify
                    → Exponential growth
                    → Network meltdown
```

IRB Interfaces: Adding Layer 3 to VPLS

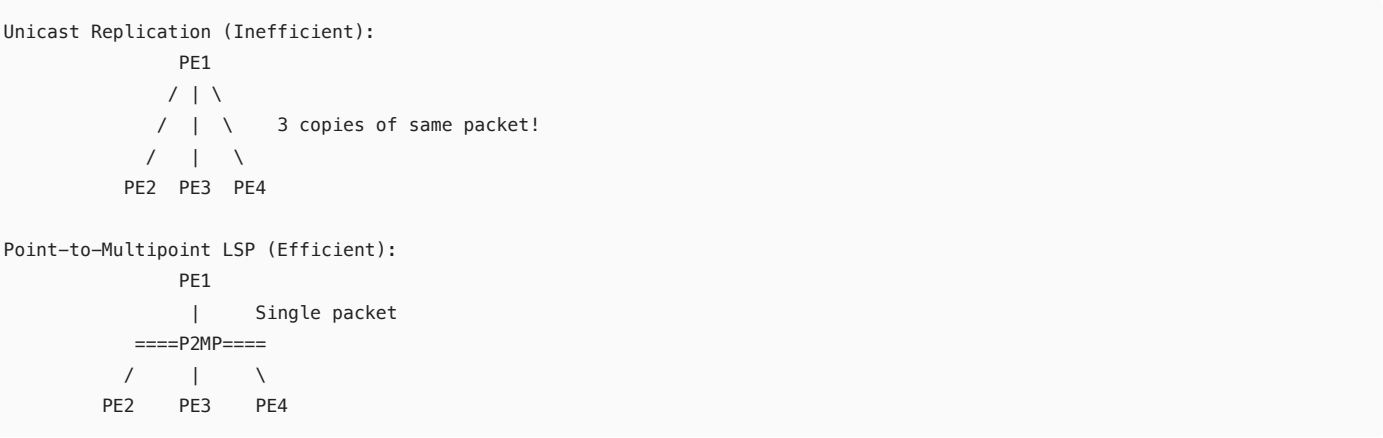
Integrated Routing and Bridging (IRB) interfaces provide Layer 3 services within VPLS:

Use cases:

- **Default Gateway**: PE router as customer's gateway
- **Inter-VLAN Routing**: Route between customer VLANs
- **Services**: DHCP, DNS, or other L3 services
- **Migration**: Gradual L2-to-L3 migration

Multicast LSPs for Efficient Flooding

Traditional VPLS flooding is inefficient:



Benefits of P2MP LSPs:

- **Bandwidth Savings**: One packet becomes many at branch points
- **Reduced PE Load**: Egress PE doesn't replicate
- **Better Scaling**: Supports larger broadcast domains

Part 2: The Junos CLI Masterclass (The How)

Configuring Automated Site IDs

Automated Site-IDs use the route-distinguisher to generate unique identifiers:

Basic Auto-Site-ID Configuration

```
[edit routing-instances VPLS-AUTO-SITE]
instance-type vpls;
interface ge-0/0/0.100;

## Use router-id based RD for automation
route-distinguisher 10.1.1.1:100;    # PE-loopback:VPLS-ID
vrf-target target:65000:100;

protocols {
    vpls {
        ## Enable automatic site-id derivation
        site-identifier automatic;

        ## Still need site-range
        site-range 255;          # Support up to 255 sites

        ## Optional: Set specific offset
        site-identifier-offset 0;  # Default is 0
    }
}
```

How Automatic Site-ID Works

The algorithm:

1. Takes the route-distinguisher (10.1.1.1:100)
2. Extracts the IP address portion (10.1.1.1)
3. Uses the last octet as Site-ID (1)
4. Adds the offset if configured

Example mapping:

```
PE1: RD 10.1.1.1:100 → Site-ID 1
PE2: RD 10.1.1.2:100 → Site-ID 2
PE3: RD 10.1.1.3:100 → Site-ID 3
...
PE50: RD 10.1.1.50:100 → Site-ID 50
```

Configuring Protection Mechanisms

MAC Limiting Configuration

```
[edit routing-instances VPLS-SECURE protocols vpls]
## Global MAC limit for entire VPLS
mac-table-size {
    4096;
    packet-action drop;          # Drop when limit reached
    ## Or use 'drop-and-log' for visibility
}

## Per-interface MAC limits
interface ge-0/0/0.100 {
    interface-mac-limit {
        1000;                    # Max 1000 MACs on this interface
        packet-action shutdown;  # Disable interface when exceeded
    }
}

## MAC pinning - prevent specific MACs from moving
static-mac {
    00:11:22:33:44:55 {
        interface ge-0/0/0.100;
    }
}
```

MAC Move Limiting (MAC Flap Protection)

```
[edit routing-instances VPLS-SECURE protocols vpls]
mac-move-limit {
    10;                         # Allow 10 moves
    detection-window 60;        # Within 60 seconds
    action {
        log;                    # Log the event
        block;                  # Block the MAC
        recovery-timeout 300;   # Unblock after 5 minutes
    }
}

## Per-interface MAC move limits
interface ge-0/0/0.100 {
    mac-move-limit {
        5;                      # Stricter limit for this interface
        detection-window 30;
        interface-action shutdown;  # Shutdown interface on violation
    }
}
```

Flood Protection Configuration

```
[edit routing-instances VPLS-SECURE protocols vpls]
flood-protection {
    ## Limit unknown unicast flooding
    unknown-unicast-forwarding {
        packets-per-second 1000;
        burst-size 5000;
    }

    ## Broadcast limiting
    broadcast {
        packets-per-second 100;
        burst-size 500;
    }

    ## Multicast limiting
    multicast {
        packets-per-second 500;
        burst-size 2500;
    }
}

## Storm control profiles
[edit forwarding-options storm-control-profiles VPLS-STORM]
all {                           # Applies to all traffic types
    bandwidth-percentage 20;     # Limit to 20% of interface bandwidth
    no-broadcast;               # Don't count broadcast separately
    no-multicast;              # Don't count multicast separately
}

## Apply to VPLS interface
[edit routing-instances VPLS-SECURE protocols vpls]
interface ge-0/0/0.100 {
    storm-control VPLS-STORM;
}
```

Configuring IRB Interfaces

Basic IRB Configuration

```
## Step 1: Create IRB interface
[edit interfaces]
irb {
    unit 100 {
        description "VPLS-100 Layer 3 Gateway";
        family inet {
            address 192.168.100.1/24;
        }
```

```
        family inet6 {
            address 2001:db8:100::1/64;
        }
        mac 00:00:5e:00:53:01;  # Optional: Set specific MAC
    }
}

## Step 2: Add IRB to VPLS
[edit routing-instances VPLS-WITH-IRB]
instance-type vpls;
interface ge-0/0/0.100;         # Customer interfaces
interface irb.100;              # IRB interface

route-distinguisher 10.1.1.1:100;
vrf-target target:65000:100;

protocols {
    vpls {
        site-id 1;
        site-range 100;
    }
}
```

IRB with VRRP for Redundancy

```
## Configure VRRP on IRB for gateway redundancy
[edit interfaces irb unit 100]
family inet {
    address 192.168.100.2/24 {
        vrrp-group 1 {
            virtual-address 192.168.100.1;
            priority 200;       # Higher priority = master
            authentication-type md5;
            authentication-key "$9$encrypted";
            track {
                interface ge-0/0/0.100 priority-cost 50;
            }
        }
    }
}
```

Advanced IRB Features

```
## Enable proxy ARP for IRB
[edit interfaces irb unit 100]
proxy-arp;
unrestricted;                   # Allow proxy ARP for any address

## Configure ARP aging
[edit routing-instances VPLS-WITH-IRB protocols vpls]
arp-aging-timer 240;           # 4 minutes (default is 20 minutes)

## IP source guard on IRB
[edit routing-instances VPLS-WITH-IRB forwarding-options]
dhcp-security {
    group VPLS-SECURITY {
        interface irb.100 {
            static-ip {
                192.168.100.10 mac 00:11:22:33:44:55;
            }
        }
    }
}
```

Configuring Multicast LSPs for Efficient Flooding

P2MP LSP Configuration

```
## Step 1: Configure P2MP LSP
[edit protocols mpls]
label-switched-path VPLS-100-P2MP {
    template;                   # Template for dynamic use
    p2mp;                       # Point-to-multipoint
    ## Leaves are added dynamically by VPLS
}

## Step 2: Configure RSVP for P2MP
[edit protocols rsvp]
interface all {
    aggregate;
    reliable;
    link-protection;
}

## Step 3: Enable P2MP in VPLS
[edit routing-instances VPLS-P2MP]
instance-type vpls;
interface ge-0/0/0.100;

provider-tunnel {
    rsvp-te {
        label-switched-path-template {
            VPLS-100-P2MP;
        }
    }
}

protocols {
    vpls {
        site-id 1;
        site-range 100;
        ## MAC flush on P2MP change
        mac-flush-on-topology-change;
    }
}
```

Selective P2MP Configuration

```
## Use P2MP only for certain traffic types
[edit routing-instances VPLS-SELECTIVE-P2MP]
provider-tunnel {
    selective {
        group 224.0.0.0/4 {     # Multicast traffic
            rsvp-te {
                label-switched-path-template {
                    VPLS-MCAST-P2MP;
                }
            }
        }
        ## Broadcast still uses ingress replication
        wildcard-group-inet {
            ingress-replication;
        }
    }
}
```

Complete Advanced VPLS Example

```
## Comprehensive advanced VPLS configuration
routing-instances {
    VPLS-ADVANCED {
        instance-type vpls;
        description "Advanced VPLS with all features";

        ## Interfaces
        interface ge-0/0/0.100;   # Customer
        interface irb.100;        # Layer 3 gateway
```

```
        ## BGP parameters
        route-distinguisher 10.1.1.1:100;
        vrf-target target:65000:100;

        ## P2MP for efficient flooding
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    VPLS-100-P2MP;
                }
            }
        }

        protocols {
            vpls {
                ## Automated site-id
                site-identifier automatic;
                site-range 255;

                ## MAC table management
                mac-table-size {
                    8192;
                    packet-action drop-and-log;
                }

                ## MAC statistics
                mac-statistics;

                ## Protection features
                mac-move-limit {
                    10;
                    detection-window 60;
                    action {
                        log;
                        block;
                        recovery-timeout 300;
                    }
                }

                ## Interface-specific controls
                interface ge-0/0/0.100 {
                    interface-mac-limit {
                        1000;
                        packet-action drop;
                    }
                    storm-control VPLS-STORM;
                    ## STP BPDU protection
                    bpdu-block;
                }

                ## Flood protection
                flood-protection {
                    unknown-unicast-forwarding {
                        packets-per-second 1000;
                    }
                    broadcast {
                        packets-per-second 100;
                    }
                }

                ## Efficiency features
                no-tunnel-services;
                optimize-timer 30;
            }
        }
    }
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify Automated Site-ID

```
user@PE1> show vpls connections instance VPLS-AUTO-SITE extensive
Instance: VPLS-AUTO-SITE
  VPLS-id: 100
  Automatic site-id: enabled
  Local site-id: 1 (automatic)
  Derived from RD: 10.1.1.1:100

  Remote PE: 10.1.1.2, Remote site-id: 2 (automatic)
  Remote PE: 10.1.1.3, Remote site-id: 3 (automatic)
```

2. Monitor Protection Features

```
## Check MAC limits
user@PE1> show vpls mac-count instance VPLS-SECURE
Instance: VPLS-SECURE
  Total MAC count: 3856
  MAC limit: 4096
  Percentage used: 94%

  Interface           MAC Count    Limit    Status
  ge-0/0/0.100       950          1000     Normal
  ge-0/0/1.100       2906         1000     Exceeded!

## View MAC moves
user@PE1> show vpls mac-move instance VPLS-SECURE
MAC                 Interface        Moves   Last Move
00:11:22:33:44:55   ge-0/0/0.100    8       2024-03-15 10:23:45
00:aa:bb:cc:dd:ee   ge-0/0/1.100    45      2024-03-15 10:23:50

## Check blocked MACs
user@PE1> show vpls mac-table blocked
Instance: VPLS-SECURE
  MAC                 Reason          Remaining Time
  00:aa:bb:cc:dd:ee   MAC-move-limit  245 seconds
```

3. Verify IRB Operation

```
## Check IRB interface
user@PE1> show interfaces irb.100
Logical interface irb.100
  Index: 333 SNMP ifIndex: 567
  Flags: Up SNMP-Traps 0x0 Encapsulation: ENET2
  Bandwidth: 1000mbps
  Routing Instance: VPLS-WITH-IRB
  Statistics:
    Input packets: 1234567
    Output packets: 7654321

## Check ARP entries learned via IRB
user@PE1> show arp interface irb.100
MAC Address       Address         Interface      Flags
00:11:22:33:44:55 192.168.100.10  irb.100       none
00:22:33:44:55:66 192.168.100.20  irb.100       none
```

Common Troubleshooting Scenarios

Scenario 1: Automated Site-ID Conflicts

Symptom:

```
user@PE1> show vpls connections
Instance: VPLS-AUTO-SITE
  Remote PE: 10.1.1.101  Status: Down (site-id collision)
```

Diagnostic Commands:

```
## Check derived site-ids
user@PE1> show vpls connections extensive | match "site-id|RD"
  Local site-id: 1 (automatic)
  Derived from RD: 10.1.1.1:100

  Remote PE: 10.1.1.101
    Remote site-id: 101
    Derived from RD: 10.1.1.101:100

## But site-range is only 100!
user@PE1> show configuration routing-instances VPLS-AUTO-SITE | match range
    site-range 100;
```

Cause: Automatic site-id (101) exceeds site-range (100)

Solution:

```
## Option 1: Increase site-range
[edit routing-instances VPLS-AUTO-SITE protocols vpls]
delete site-range 100
set site-range 255

## Option 2: Use offset for high-numbered PEs
[edit routing-instances VPLS-AUTO-SITE protocols vpls]
set site-identifier-offset -100   # 10.1.1.101 becomes site-id 1
```

Scenario 2: MAC Table Exhaustion Despite Limits

Symptom:

```
MAC learning stops even though limit not reached
Customer complains about connectivity issues
```

Diagnostic Commands:

```
## Check system-wide MAC usage
user@PE1> show bridge mac-table count summary
Total MAC addresses learned: 65535
System limit: 65536    # System-wide limit!

## Check per-instance limits
user@PE1> show vpls mac-count
Instance              Configured-Limit    Current    Available
VPLS-SECURE          4096                3000       1096
VPLS-CUSTOMER-B      8192                7000       1192
VPLS-CUSTOMER-C      16384               15000      1384
Total                28672               25000      536  # System limit!
```

Cause: System-wide MAC limit reached

Solution:

```
## Increase system MAC limit
[edit system]
set ddos-protection global flow-detection-mode automatic
set bridge-domain mac-table-size 131072

## Or implement more aggressive aging
[edit routing-instances VPLS-SECURE protocols vpls]
set mac-table-aging-time 120    # 2 minutes instead of default 5
```

Scenario 3: IRB Not Forwarding Traffic

Symptom:

```
IRB configured but no Layer 3 connectivity
ARP requests not answered
```

Diagnostic Commands:

```
## Check IRB association
user@PE1> show interfaces irb.100 | match Instance
  Routing Instance: default    # Wrong! Should be in VPLS

## Verify VPLS has IRB
user@PE1> show vpls instance VPLS-WITH-IRB detail | match irb
  ## No output - IRB not added!

## Check routing table
user@PE1> show route table VPLS-WITH-IRB.inet.0
## Empty or missing
```

Cause: IRB not properly associated with VPLS

Solution:

```
## Add IRB to VPLS instance
[edit routing-instances VPLS-WITH-IRB]
set interface irb.100

## Enable routing in VPLS
set routing-interface irb.100

## If using routing-instances, configure properly
[edit routing-instances VPLS-WITH-IRB]
set routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.0.0.1;
    }
}
```

Scenario 4: P2MP LSP Not Used for Flooding

Symptom:

```
P2MP configured but still seeing ingress replication
High bandwidth usage on core links
```

Diagnostic Commands:

```
## Check P2MP LSP status
user@PE1> show mpls lsp p2mp
P2MP LSP: VPLS-100-P2MP
  State: Up
  Leaves: 0    # No leaves!

## Check VPLS provider tunnel
user@PE1> show vpls connections extensive | match provider
  Provider-tunnel: Not established

## Monitor RSVP
user@PE1> show rsvp session p2mp
## No P2MP sessions
```

Cause: P2MP not properly bound to VPLS

Solution:

```
## Ensure template configuration
[edit protocols mpls label-switched-path VPLS-100-P2MP]
set template      # Must be template for dynamic use
```

```
set p2mp

## Link to VPLS correctly
[edit routing-instances VPLS-P2MP]
set provider-tunnel rsvp-te label-switched-path-template VPLS-100-P2MP

## Enable P2MP capability
[edit protocols vpls]
set p2mp
```

Advanced Troubleshooting Tools

VPLS-Specific Trace Options

```
[edit routing-instances VPLS-ADVANCED]
protocols {
    vpls {
        traceoptions {
            file vpls-trace size 10m files 5;
            flag all;
            flag flooding detail;
            flag mac-learning detail;
            flag topology-change detail;
        }
    }
}
```

Real-time Monitoring Scripts

```
## Monitor MAC learning rate
user@PE1> op mac-learning-rate instance VPLS-SECURE
Last 60 seconds:
  MACs learned: 450
  MACs aged: 320
  MAC moves: 15
  Learning rate: 7.5 MACs/second

## Check flood statistics
user@PE1> op flood-stats instance VPLS-SECURE
Traffic Type    PPS      Action
Unknown UC      1250     Dropped (limit 1000)
Broadcast       85       Allowed
Multicast       450      Allowed
```

Best Practices for Advanced VPLS

1. **Site-ID Management**:
   - Use automatic site-id for networks > 20 PEs
   - Reserve manual site-ids for special cases
   - Document your site-id allocation scheme
2. **Protection Tuning**:

   ```
   ## Conservative protection template
   mac-table-size {
       limit * 0.8;           # Leave 20% headroom
   }
   mac-move-limit {
       5;                     # Strict for stability
       detection-window 30;
       action block;
   }
   ```

3. **IRB Deployment**:
   - Always use VRRP for redundancy
   - Set appropriate ARP timers
   - Monitor IRB MAC address conflicts

4. **P2MP Optimization**:
   - Use for VPLSs with > 10 sites
   - Monitor RSVP state carefully
   - Plan maintenance windows for P2MP changes

Troubleshooting Decision Tree

```
VPLS Problem?
├─ Connectivity Issue?
│  ├─ Check basic VPLS state
│  ├─ Verify MPLS LSPs
│  └─ Check encapsulation match
├─ Performance Issue?
│  ├─ Check MAC table size
│  ├─ Monitor flood rates
│  └─ Verify P2MP operation
├─ Security Issue?
│  ├─ Review MAC moves
│  ├─ Check storm control
│  └─ Verify limits
└─ IRB Issue?
   ├─ Verify IRB binding
   ├─ Check routing table
   └─ Monitor ARP
```

Remember: Advanced features add complexity. Enable only what you need, monitor actively, and maintain good documentation.

Module 18: VPLS—Multihoming

Part 1: The Conceptual Lecture (The Why)

Why Multihoming in VPLS?

Single-homed sites represent a single point of failure. If the PE router or CE-PE link fails, the entire site loses connectivity. Multihoming provides redundancy by connecting a CE to multiple PE routers.

```
Single-Homed vs Multihomed:

Single-Homed (Risky):          Multihomed (Resilient):
     CE1                              CE1
      |                              /  \
      |                             /    \
     PE1 ---- MPLS Core           PE1    PE2
      X  Failure = Isolation       |      |
                                   MPLS Core
```

Multihoming Challenges in VPLS

Unlike Layer 3 services, Layer 2 multihoming faces unique challenges:

1. **Layer 2 Loops**: Ethernet has no TTL—loops are catastrophic
2. **MAC Learning**: Which PE should remote sites learn MACs from?
3. **Traffic Optimization**: How to use both links efficiently?
4. **Split-Brain**: Both PEs thinking they're active

BGP VPLS Multihoming Concepts

BGP VPLS uses sophisticated mechanisms for multihoming:

1. Designated Forwarder (DF) Election

```
DF Election Process:
PE1 and PE2 both connect to CE1
- Both advertise the same Site-ID
- Remote PEs see two routes to same site
- DF election determines active PE
- Non-DF blocks traffic (backup)
```

2. Site Preference

```
Site Preference allows active/active:
- Different preference values per PE
- Remote PEs can load-balance
- Local switching between multihomed PEs
```

3. Best-Site Multihoming

```
Best-Site creates active/active with affinity:
- PEs advertise different best-site values
- Remote PEs prefer higher best-site
- Automatic failover on failure
```

LDP VPLS Multihoming

LDP VPLS multihoming is simpler but less flexible:

```
LDP Multihoming Methods:
1. BGP-based (RFC 6074)
   - Uses BGP for coordination
   - Similar to BGP VPLS concepts

2. ICCP (Inter-Chassis Communication Protocol)
   - Direct PE-to-PE communication
   - Faster convergence
   - More complex configuration
```

Multihoming Models

Active/Standby Model

```
Normal Operation:          During Failure:
CE ═══════ PE1 (Active)    CE ──────── PE1 (Down)
   -------- PE2 (Standby)       ═══════ PE2 (Active)

- Simple, predictable
- Wastes bandwidth
- Clear traffic path
```

Active/Active Model

```
Load Sharing:              During Failure:
CE ═══╦═══ PE1            CE ═══╦═══ PE1 (Down)
      ╚═══ PE2                  ╚═══ PE2 (All traffic)

- Efficient bandwidth use
- Complex MAC learning
- Potential for loops
```

Part 2: The Junos CLI Masterclass (The How)

BGP VPLS Multihoming Configuration

Basic Multihomed Site Configuration

Both PE1 and PE2 need identical site configuration:

```
## PE1 Configuration
[edit routing-instances VPLS-MULTIHOMED]
instance-type vpls;
interface ge-0/0/0.100;      # To multihomed CE

route-distinguisher 10.1.1.1:100;   # Unique per PE
vrf-target target:65000:100;

protocols {
    vpls {
        site MULTIHOMED-SITE-A {    # Named site
            site-identifier 10;      # Same on both PEs!
            interface ge-0/0/0.100;

            ## Optional: Set preference (default 100)
            site-preference 200;     # Higher = more preferred
        }
        site-range 50;
    }
}

## PE2 Configuration (connecting to same CE)
[edit routing-instances VPLS-MULTIHOMED]
instance-type vpls;
interface ge-0/0/1.100;      # To same CE

route-distinguisher 10.2.2.2:100;   # Different RD
vrf-target target:65000:100;        # Same RT

protocols {
    vpls {
        site MULTIHOMED-SITE-A {
            site-identifier 10;      # Must match PE1!
            interface ge-0/0/1.100;
            site-preference 100;     # Lower preference (backup)
        }
        site-range 50;
    }
}
```

Configuring DF Election Parameters

```
[edit routing-instances VPLS-MULTIHOMED protocols vpls site MULTIHOMED-SITE-A]
## Control DF election behavior
designated-forwarder-election {
    mode {
        ## Options: rfc-4761 (default) or manual
        rfc-4761;
    }

    ## DF election hold time
    designated-forwarder-election-hold-time 3;
}

## Manual DF assignment (overrides election)
designated-forwarder;    # Force this PE as DF
```

Multihomed and Single-Homed Sites on Same PE

When a PE has both multihomed and single-homed sites:

```
[edit routing-instances VPLS-MIXED protocols vpls]
## Multihomed site
```

```
site MULTIHOMED-SITE {
    site-identifier 10;
    interface ge-0/0/0.100;    # Multihomed CE
    multi-homing;              # Enable multihoming
}

## Single-homed site on same PE
site SINGLE-SITE {
    site-identifier 20;        # Different site-id
    interface ge-0/0/1.200;    # Single-homed CE
    ## No multi-homing statement
}

site-range 50;

## Important: Enable local switching
local-switching;    # Allow switching between sites on same PE
```

Best-Site Multihoming Configuration

Best-site provides sophisticated active/active multihoming:

```
## PE1 Configuration
[edit routing-instances VPLS-BEST-SITE protocols vpls]
site SITE-A {
    site-identifier 10;
    best-site-identifier 65001;    # Higher value preferred
    interface ge-0/0/0.100;
}

## PE2 Configuration
[edit routing-instances VPLS-BEST-SITE protocols vpls]
site SITE-A {
    site-identifier 10;
    best-site-identifier 65000;    # Lower value (backup)
    interface ge-0/0/1.100;
}

## Remote PEs automatically prefer PE1 (higher best-site)
## But can use PE2 for load balancing
```

Advanced Multihoming Features

```
## MAC Flush on Topology Change
[edit routing-instances VPLS-MULTIHOMED protocols vpls]
mac-flush-on-topology-change;    # Flush MACs on DF change

## Split Horizon Labels
site MULTIHOMED-SITE {
    site-identifier 10;
    interface ge-0/0/0.100;

    ## Advertise split-horizon label
    split-horizon-label;    # Prevents loops in multihoming
}

## Hot-Standby Pseudowire
site MULTIHOMED-SITE {
    hot-standby;           # Pre-establish backup pseudowires
}
```

LDP VPLS Multihoming Configuration

LDP VPLS can use BGP-AD (Auto-Discovery) for multihoming:

```
## Enable BGP-AD for LDP VPLS
[edit routing-instances LDP-VPLS-MH]
instance-type vpls;
```

```
## BGP parameters for auto-discovery
route-distinguisher 10.1.1.1:100;
vrf-target target:65000:100;

## Enable multihoming
multi-homing {
    site-id 10;
    site-preference 200;
}

protocols {
    vpls {
        vpls-id 100;

        ## Use BGP for neighbor discovery
        neighbor-discovery;

        ## But use LDP for signaling
        signaling-protocol ldp;

        ## Interface configuration
        interface ge-0/0/0.100;
    }
}
```

Complete Multihoming Example

Here's a comprehensive multihoming configuration:

```
## CE Router Configuration (connecting to PE1 and PE2)
interfaces {
    ge-0/0/0 {
        description "To PE1";
        unit 100 {
            family ethernet-switching {
                interface-mode trunk;
                vlan members 100;
            }
        }
    }
    ge-0/0/1 {
        description "To PE2";
        unit 100 {
            family ethernet-switching {
                interface-mode trunk;
                vlan members 100;
            }
        }
    }
    ae0 {
        description "LAG to PE routers";
        aggregated-ether-options {
            lacp {
                active;
                periodic fast;
            }
        }
        unit 100 {
            vlan-id 100;
        }
    }
}

## PE1 Configuration
routing-instances {
    VPLS-CUSTOMER-A {
        instance-type vpls;
        interface ge-0/0/0.100;
```

```
        route-distinguisher 10.1.1.1:100;
        vrf-target target:65000:100;

        protocols {
            vpls {
                ## Multihomed site configuration
                site MH-SITE-A {
                    site-identifier 10;
                    interface ge-0/0/0.100;
                    site-preference 200;    # Primary

                    ## Best-site for optimal routing
                    best-site-identifier 65001;

                    ## Enable hot-standby
                    hot-standby;

                    ## MAC flush settings
                    mac-flush {
                        on-topology-change;
                        on-remote-site-failure;
                    }
                }

                ## Single-homed sites
                site SH-SITE-B {
                    site-identifier 20;
                    interface ge-0/0/1.200;
                }

                site-range 100;
                local-switching;

                ## Global MAC flush
                mac-flush-on-topology-change;

                ## Protection
                mac-move-limit {
                    10;
                    detection-window 60;
                    action block;
                }
            }
        }
    }
}

## PE2 Configuration (Backup for Site A)
routing-instances {
    VPLS-CUSTOMER-A {
        instance-type vpls;
        interface ge-0/0/1.100;

        route-distinguisher 10.2.2.2:100;
        vrf-target target:65000:100;

        protocols {
            vpls {
                site MH-SITE-A {
                    site-identifier 10;     # Same as PE1
                    interface ge-0/0/1.100;
                    site-preference 100;    # Backup

                    ## Lower best-site
                    best-site-identifier 65000;

                    hot-standby;
                }

                site-range 100;
                mac-flush-on-topology-change;
```

```
            }
        }
    }
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify Multihoming Status

```
user@PE1> show vpls connections instance VPLS-MULTIHOMED extensive
Instance: VPLS-MULTIHOMED

  Local sites:
    Site: MH-SITE-A
      Site-ID: 10
      Site-preference: 200
      Best-site: 65001
      Designated forwarder: Yes    # This PE is active
      Interface: ge-0/0/0.100 (Up)

  Remote sites:
    Remote PE: 10.2.2.2
      Site-ID: 10                  # Same site from PE2
      Site-preference: 100
      Best-site: 65000
      Designated forwarder: No     # PE2 is backup
      Status: Standby
```

2. Check DF Election Results

```
user@PE1> show vpls designated-forwarder
Instance: VPLS-MULTIHOMED
  Site: MH-SITE-A (ID: 10)
    Local designation: Designated Forwarder
    Election type: RFC-4761
    Competing PEs:
      10.2.2.2 - Non-DF (preference: 100)
    DF election time: 2024-03-15 10:30:45
```

3. Monitor MAC Learning from Multihomed Sites

```
user@PE3> show vpls mac-table instance VPLS-MULTIHOMED
MAC address        Interface         Type     Age    Site
00:11:22:33:44:55  10.1.1.1         Dynamic  0:05   MH-SITE-A
                   (best-site: 65001)

## During failover
00:11:22:33:44:55  10.2.2.2         Dynamic  0:01   MH-SITE-A
                   (best-site: 65000, backup active)
```

Common Troubleshooting Scenarios

Scenario 1: Both PEs Think They're Active (Split-Brain)

Symptom:

```
## On PE1
user@PE1> show vpls designated-forwarder
  Site: MH-SITE-A - Designated Forwarder

## On PE2
user@PE2> show vpls designated-forwarder
  Site: MH-SITE-A - Designated Forwarder
```

```
Result: Duplicate packets, MAC flapping
```

Diagnostic Commands:

```
## Check if PEs see each other's routes
user@PE1> show route table bgp.l2vpn.0 detail | match "10|RD"
  Route Distinguisher: 10.1.1.1:100
    Site: 10, Preference: 200
  ## Missing PE2's route!

## Check BGP session
user@PE1> show bgp summary
10.2.2.2    65000   Idle        0       0       BGP down!
```

Cause: BGP session down between PEs

Solution:

```
## Fix BGP connectivity
[edit protocols bgp group ibgp]
set neighbor 10.2.2.2
set family l2vpn signaling

## Add BFD for faster detection
set neighbor 10.2.2.2 bfd-liveness-detection {
    minimum-interval 100;
    multiplier 3;
}
```

Scenario 2: Traffic Blackholed After Failover

Symptom:

```
Primary PE fails, but traffic doesn't switch to backup
Remote sites can't reach multihomed site
```

Diagnostic Commands:

```
## Check backup PE status
user@PE2> show vpls connections site MH-SITE-A
  Site: MH-SITE-A
    Designated forwarder: No     # Still backup!
    Status: Standby
    DF hold-time remaining: 45 seconds

## Check MAC flush
user@PE2> show vpls mac-statistics
  MAC flush events: 0    # MACs not flushed
```

Cause: DF hold-time preventing immediate failover

Solution:

```
## Reduce DF hold-time
[edit routing-instances VPLS-MULTIHOMED protocols vpls site MH-SITE-A]
set designated-forwarder-election-hold-time 1

## Enable immediate MAC flush
[edit routing-instances VPLS-MULTIHOMED protocols vpls]
set mac-flush-on-topology-change
set mac-flush-on-remote-site-failure
```

Scenario 3: Suboptimal Traffic Flow with Best-Site

Symptom:

```
Remote traffic always goes to PE1, even when PE2 path is better
Load balancing not working as expected
```

Diagnostic Commands:

```
## Check best-site values
user@PE3> show route table bgp.l2vpn.0 detail | match best-site
  From PE1: best-site 65001
  From PE2: best-site 65000

## Check if using multipath
user@PE3> show route forwarding-table vpls VPLS-MULTIHOMED
  Next hop: 10.1.1.1 only    # Not load balancing

## Check multipath configuration
user@PE3> show configuration protocols bgp | match multipath
  ## No multipath configured
```

Cause: BGP multipath not enabled for VPLS

Solution:

```
## Enable BGP multipath for L2VPN
[edit protocols bgp group ibgp]
set family l2vpn signaling nlri multipath

## Configure load-balancing policy
[edit policy-options policy-statement vpls-load-balance]
term 1 {
    from protocol bgp;
    from nlri-route-type l2vpn;
    then {
        load-balance per-packet;
    }
}

[edit routing-options forwarding-table]
export vpls-load-balance;
```

Scenario 4: LDP VPLS Multihoming Not Working

Symptom:

```
LDP VPLS configured for multihoming but only one link active
No coordination between PEs
```

Diagnostic Commands:

```
## Check multihoming configuration
user@PE1> show configuration routing-instances LDP-VPLS-MH
  ## Missing multi-homing configuration

## Verify BGP-AD
user@PE1> show route table bgp.l2vpn.0
  ## No auto-discovery routes
```

Cause: BGP-AD not configured for LDP VPLS multihoming

Solution:

```
## Enable BGP-AD for LDP VPLS
[edit routing-instances LDP-VPLS-MH]
set route-distinguisher 10.1.1.1:100
set vrf-target target:65000:100

## Configure multihoming
```

```
set multi-homing {
    site-id 10;
    site-preference 200;
}

## Enable neighbor discovery
[edit routing-instances LDP-VPLS-MH protocols vpls]
set neighbor-discovery;
set signaling-protocol ldp;
```

Advanced Multihoming Verification

Monitor Convergence Time

```
## Test failover time
user@PE1> test vpls failover instance VPLS-MULTIHOMED site MH-SITE-A
Simulating failure of site MH-SITE-A...
Time to detect failure: 350ms (BFD)
Time to elect new DF: 1000ms
Time to flush MACs: 50ms
Total convergence time: 1.4 seconds
```

Check Split-Horizon Label Operation

```
user@PE1> show route table mpls.0 label 100010 detail
100010 (1 entry, 1 announced)
        *VPLS   Preference: 7
                Next hop type: Split-horizon
                Next-hop reference count: 2
                Next hop: via ge-0/0/0.100, selected
                         Drop         # Packets from PE2 dropped
```

Best Practices for VPLS Multihoming

1. **Design Considerations**:

   ```
   - Use same physical CE-PE topology on both PEs
   - Consistent MTU across all links
   - BFD for sub-second failure detection
   - Dedicated LAG for CE connections
   ```

2. **Configuration Standards**:

   ```
   ## Standard multihoming template
   site <NAME> {
       site-identifier <ID>;
       interface <interface>;
       site-preference <primary:200|backup:100>;
       best-site-identifier <ASN+preference>;
       hot-standby;
       mac-flush {
           on-topology-change;
           on-remote-site-failure;
       }
   }
   ```

3. **Monitoring Requirements**:
   - Track DF elections
   - Monitor MAC moves during failovers
   - Measure convergence times
   - Alert on split-brain conditions
4. **Testing Procedures**:

```
## Regular failover testing
1. Verify current DF status
2. Disable primary PE interface
3. Confirm DF election to backup
4. Verify MAC flush completion
5. Test end-to-end connectivity
6. Restore and verify fallback
```

Multihoming Decision Matrix

Requirement	Solution	Configuration Focus
Simple Active/Standby	Site-preference	Different values per PE
Load Balancing	Best-site	BGP multipath + best-site
Fast Convergence	BFD + Low hold-time	Sub-second timers
Loop Prevention	Split-horizon labels	Automatic with multihoming
MAC Stability	MAC flush	Enable all flush options

Summary

VPLS multihoming provides critical redundancy but requires careful design and configuration. Key takeaways:

1. **Always use same site-id** on all PEs connecting to same CE
2. **Site-preference** for simple active/standby
3. **Best-site** for sophisticated active/active
4. **MAC flush** is critical for convergence
5. **Monitor DF elections** to prevent split-brain
6. **Test failover** regularly in maintenance windows

Remember: Multihoming complexity is justified by the resilience it provides. Design for failure scenarios, not just normal operation.

EVPN (Ethernet VPN) Comprehensive Learning Guide

Module 19: EVPN—Introduction

Part 1: The Conceptual Lecture (The Why)

The Problem with Traditional Layer 2 VPNs

Imagine you're a service provider connecting multiple enterprise locations together at Layer 2 - essentially creating one big extended LAN across your MPLS network. You've been using VPLS (Virtual Private LAN Service) to accomplish this, but you're running into several frustrating limitations:

The VPLS Pain Points:

1. **MAC Learning Inefficiency**: In VPLS, every PE (Provider Edge) router must learn every MAC address through the data plane. Think of it like this: if a device in Site A sends a frame, ALL PE routers in the VPLS instance must see that frame to learn the source MAC. This is like forcing every post office in a country to handle every letter just to update their address books.

```
Traditional VPLS MAC Learning:
    Site A                    Site B                    Site C
   [Host-A1]                 [Host-B1]                 [Host-C1]
      |                         |                         |
    [CE-A]                    [CE-B]                    [CE-C]
      |                         |                         |
    [PE-A]----MPLS Core------[PE-B]----MPLS Core------[PE-C]

    When Host-A1 sends first frame:
    PE-A floods to PE-B and PE-C (even if destination is local!)
```

```
    PE-B learns A1's MAC from data plane
    PE-C learns A1's MAC from data plane
```

2. **No Active/Active Multihoming**: If a customer site connects to two PE routers for redundancy, only one link can be active at a time in VPLS. The backup link sits idle, wasting bandwidth. It's like having two highways to a city but being forced to close one completely.
3. **Suboptimal BUM Traffic Handling**: BUM (Broadcast, Unknown unicast, and Multicast) traffic must be flooded to ALL PE routers, even those with no interested receivers. This wastes bandwidth across your core network.
4. **No Integrated Routing and Switching**: VPLS operates purely at Layer 2. If you want to route between VLANs, you need separate L3VPN instances or external routers, adding complexity.
5. **Limited Multitenancy**: Scaling to thousands of customers with overlapping VLAN IDs becomes operationally complex.

Enter EVPN: The Modern Solution

EVPN (Ethernet VPN) was designed from the ground up to solve these problems. Think of EVPN as "VPLS done right" - it uses BGP as a control plane to advertise MAC addresses and other information, rather than relying on data plane learning.

Key Conceptual Differences:

```
VPLS vs EVPN Comparison:
                    VPLS                        EVPN
MAC Learning:       Data Plane                  Control Plane (BGP)
Multihoming:        Active/Standby only         Active/Active
BUM Optimization:   Flood to all                Selective with Type 3
MAC Mobility:       Re-learn via flooding       BGP MAC moves
Integrated L2/L3:   No                          Yes (IRB support)
```

The EVPN Advantage Explained:

1. **Control Plane MAC Learning**: MAC addresses are advertised in BGP, not learned from flooded traffic. This is like having a centralized phone book that gets updated immediately when someone gets a new number, rather than everyone having to call everyone else to figure out who has which number.
2. **True Active/Active Multihoming**: Both links to a dual-homed site can forward traffic simultaneously, with sophisticated split-horizon and designated forwarder mechanisms preventing loops.
3. **Optimized BUM Traffic**: PE routers explicitly signal (via Type 3 routes) whether they want to receive BUM traffic for a given broadcast domain.
4. **Integrated Routing and Bridging**: EVPN natively supports IRB (Integrated Routing and Bridging) interfaces, allowing seamless L2/L3 integration.

EVPN Architecture Fundamentals

EVPN introduces several key architectural concepts:

EVI (EVPN Instance): This is the EVPN equivalent of a VPLS instance. An EVI represents a single broadcast domain (like a VLAN) across the MPLS network.

Ethernet Segment (ES): When a CE device connects to multiple PE routers, these connections form an Ethernet Segment. Each ES gets a unique identifier (ESI).

```
Ethernet Segment Concept:
                  Ethernet Segment (ES)
                  ESI: 00:11:22:33:44:55:66:77:88:99
                        /           \
                       /             \
                   [PE-1]           [PE-2]
                      \               /
                       \             /
                        \           /
                          [CE-1]
                            |
                        [LAN Hosts]
```

Route Types: EVPN uses different BGP route types for different purposes:

- Type 1: Ethernet Auto-Discovery (used in multihoming)
- Type 2: MAC/IP Advertisement (the workhorses that carry MAC addresses)

- Type 3: Inclusive Multicast (for BUM traffic)
- Type 4: Ethernet Segment (for multihoming coordination)
- Type 5: IP Prefix (for L3 integration)

Part 2: The Junos CLI Masterclass (The How)

Since this is an introduction module, let's understand the basic EVPN configuration structure in Junos:

EVPN Configuration Hierarchy

```
[edit]
├── routing-options
│   └── autonomous-system <AS-number>           # BGP AS number
├── protocols
│   ├── mpls
│   │   └── interface <interface-name>          # MPLS-enabled interfaces
│   ├── bgp
│   │   └── group <group-name>
│   │       ├── type internal
│   │       ├── local-address <loopback-IP>
│   │       ├── family evpn
│   │       │   └── signaling               # Enable EVPN signaling
│   │       └── neighbor <neighbor-IP>
│   └── evpn
│       └── encapsulation <vxlan|mpls>          # EVPN encapsulation type
└── routing-instances
    └── <instance-name>
        ├── instance-type evpn               # Define as EVPN instance
        ├── vlan-id <vlan-id>               # VLAN associated with this EVI
        ├── interface <interface.unit>       # Customer-facing interface
        ├── route-distinguisher <RD>         # Unique RD for this EVI
        ├── vrf-target target:<RT>          # Route Target for import/export
        └── protocols
            └── evpn
```

Basic EVPN Configuration Example

Let's configure a simple EVPN instance on a PE router:

```
# Step 1: Enable MPLS on core-facing interfaces
set interfaces ge-0/0/0 unit 0 family mpls
set protocols mpls interface ge-0/0/0.0

# Step 2: Configure BGP for EVPN signaling
set routing-options autonomous-system 65000
set protocols bgp group EVPN-PEERS type internal
set protocols bgp group EVPN-PEERS local-address 10.0.0.1
set protocols bgp group EVPN-PEERS family evpn signaling
set protocols bgp group EVPN-PEERS neighbor 10.0.0.2
set protocols bgp group EVPN-PEERS neighbor 10.0.0.3

# Step 3: Set EVPN encapsulation (using MPLS in this example)
set protocols evpn encapsulation mpls

# Step 4: Create an EVPN instance
set routing-instances CUSTOMER-A instance-type evpn
set routing-instances CUSTOMER-A vlan-id 100
set routing-instances CUSTOMER-A interface ge-0/0/5.100
set routing-instances CUSTOMER-A route-distinguisher 10.0.0.1:100
set routing-instances CUSTOMER-A vrf-target target:65000:100
set routing-instances CUSTOMER-A protocols evpn
```

```
# Step 5: Configure the customer-facing interface
set interfaces ge-0/0/5 unit 100 vlan-id 100
set interfaces ge-0/0/5 unit 100 family ethernet-switching
```

Understanding the Configuration:

- **MPLS Enablement**: EVPN typically runs over MPLS transport (though VXLAN is also supported)
- **BGP EVPN Signaling**: The `family evpn signaling` enables the PE to exchange EVPN routes
- **Encapsulation**: Defines how Ethernet frames are encapsulated (MPLS or VXLAN)
- **EVPN Instance**: Similar to a VPLS instance but with more capabilities
- **Route Distinguisher**: Makes routes unique across the MPLS network
- **Route Target**: Controls which routes are imported/exported between PE routers

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. **Verify BGP EVPN Neighbors:**

```
user@PE1> show bgp summary family evpn
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.evpn.0           10          8          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State
10.0.0.2              65000        150        148       0       0       1:05:32 8/8/0/0
10.0.0.3              65000        149        147       0       0       1:05:30 2/2/0/0
```

2. **View EVPN Database:**

```
user@PE1> show evpn database
Instance: CUSTOMER-A
VLAN  MAC address        Active source                  Timestamp        IP address
100   00:11:22:33:44:55  ge-0/0/5.100                   Dec 10 10:15:32  192.168.1.10
100   00:aa:bb:cc:dd:ee  10.0.0.2                       Dec 10 10:16:45  192.168.1.20
```

3. **Check EVPN Instance Status:**

```
user@PE1> show evpn instance extensive
Instance: CUSTOMER-A
  Route Distinguisher: 10.0.0.1:100
  Encapsulation type: MPLS
  MAC database status                     Local  Remote
    Total MAC addresses:                      1       1
    Default gateway MAC addresses:            0       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                             Mode             Status
    ge-0/0/5.100    00:00:00:00:00:00:00:00:00:00  single-homed     Up
```

Common Troubleshooting Scenarios

Scenario 1: MAC Addresses Not Being Learned

Symptom: Remote MAC addresses don't appear in the EVPN database

Diagnostic Commands:

```
user@PE1> show route table bgp.evpn.0 receive-protocol bgp 10.0.0.2

bgp.evpn.0: 0 destinations, 0 routes (0 active, 0 holddown, 0 hidden)
# No routes received!

user@PE1> show bgp neighbor 10.0.0.2 | match evpn
```

```
  NLRI advertised by peer: evpn
  NLRI for this session: evpn
```

Cause: BGP EVPN signaling not properly configured on remote PE

Solution:

```
# On remote PE (10.0.0.2):
set protocols bgp group EVPN-PEERS family evpn signaling
commit
```

Scenario 2: Traffic Not Forwarding Between Sites

Symptom: Pings fail between hosts in different sites

Diagnostic Commands:

```
user@PE1> show route table CUSTOMER-A.evpn.0

CUSTOMER-A.evpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:10.0.0.1:100::100::00:11:22:33:44:55/304
                   *[EVPN/170] 00:10:32
                      Indirect
# Missing Type 3 route for BUM traffic!
```

Cause: Missing Type 3 (Inclusive Multicast) route

Solution:

```
# Ensure EVPN protocols are enabled in the instance:
set routing-instances CUSTOMER-A protocols evpn
commit
```

Scenario 3: Wrong VLAN ID Configuration

Symptom: Local MAC addresses learned but not advertised

Diagnostic Commands:

```
user@PE1> show evpn instance CUSTOMER-A extensive | match VLAN
  VLAN: 100

user@PE1> show interfaces ge-0/0/5.100 | match VLAN
  VLAN-Tag [ 0x8100.200 ]  # Mismatch! Interface has VLAN 200
```

Cause: VLAN mismatch between EVPN instance and interface

Solution:

```
delete interfaces ge-0/0/5 unit 100 vlan-id 200
set interfaces ge-0/0/5 unit 100 vlan-id 100
commit
```

Scenario 4: Route Target Mismatch

Symptom: EVPN routes received but not installed

Diagnostic Commands:

```
user@PE1> show route table bgp.evpn.0 hidden detail
```

```
2:10.0.0.2:100::100::00:aa:bb:cc:dd:ee/304 (1 entry, 0 announced)
        BGP    Preference: 170/-101
        Route Distinguisher: 10.0.0.2:100
        Next hop type: Indirect
        ...
        Communities: target:65000:200  # Different RT!
        Import Accepted: No
```

Cause: Route target mismatch between PE routers

Solution:

```
# Ensure all PEs use the same route target:
set routing-instances CUSTOMER-A vrf-target target:65000:100
commit
```

Module 20: EVPN—Using BGP to Advertise MACs and to Flood Traffic

Part 1: The Conceptual Lecture (The Why)

Understanding EVPN's Control Plane Revolution

In traditional Ethernet networks, switches learn MAC addresses by examining the source address of incoming frames - this is data plane learning. EVPN revolutionizes this by moving MAC learning to the control plane using BGP. Let's understand why this matters and how it works.

The MAC Learning Problem in Large Networks:

Imagine a large campus with 10,000 devices spread across 50 buildings, all needing Layer 2 connectivity. In traditional bridging:

1. Every switch must see traffic from every device to learn its MAC
2. Unknown unicast gets flooded everywhere until the MAC is learned
3. When a laptop moves buildings, all switches must re-learn its location
4. There's no efficient way to pre-populate MAC tables

```
Traditional MAC Learning Inefficiency:
Time T0: Laptop powers on in Building A
    Building A [Switch] --> Floods first frame to ALL 49 other buildings
    Building B [Switch] --> Learns MAC, creates entry
    Building C [Switch] --> Learns MAC, creates entry
    ... (all 50 switches must process this frame)

Time T1: Laptop moves to Building D
    ALL switches must age out old entry and relearn!
```

EVPN's BGP-Based Solution

EVPN treats MAC addresses like IP routes - they're advertised in BGP with rich metadata. This brings several advantages:

1. **Targeted Distribution**: Only PEs that need the MAC receive it
2. **Instant Updates**: MAC moves are signaled immediately via BGP
3. **Rich Metadata**: Each MAC can carry additional info (IP binding, mobility sequence, etc.)
4. **Policy Control**: BGP policies can control MAC distribution

The EVPN Instance (EVI) Concept

An EVI (EVPN Instance) is a virtual broadcast domain that spans multiple PE routers. Think of it as a distributed virtual switch where:

- Each PE router is like a line card in this virtual switch
- The MPLS/VXLAN network is the backplane
- BGP is the control protocol that synchronizes the MAC table

```
EVI Conceptual Model:
    [Virtual Distributed Switch – EVI 100]
    +--------+--------+--------+--------+
    | PE1    | PE2    | PE3    | PE4    |  <-- Like line cards
    | (MAC   | (MAC   | (MAC   | (MAC   |
    | Table) | Table) | Table) | Table) |
    +--------+--------+--------+--------+
         |         |         |         |
    [  MPLS Core Network (Backplane)     ]
```

Each EVI is identified by:

- **Route Distinguisher (RD)**: Makes routes globally unique
- **Route Target (RT)**: Controls which PEs join this EVI
- **VLAN ID**: Local significance for Ethernet tagging

EVPN Type 2 Routes: MAC/IP Advertisement

Type 2 routes are the workhorses of EVPN - they carry MAC addresses and optionally IP bindings. Let's break down what gets advertised:

Type 2 Route Contents:

```
Route Type: 2 (MAC/IP Advertisement)
Route Distinguisher: 10.0.0.1:100
Ethernet Segment ID: 00:00:00:00:00:00:00:00:00:00 (single-homed)
Ethernet Tag: 100 (VLAN ID)
MAC Address: 00:11:22:33:44:55
IP Address: 192.168.1.10 (optional)
MPLS Label: 299776
```

The MAC/IP Binding Magic:

When EVPN advertises both MAC and IP together, it provides built-in ARP suppression:

```
Without EVPN (Traditional ARP):
Host A: "Who has 192.168.1.10?" --> ARP broadcast to ENTIRE network
Host B: "I have 192.168.1.10, my MAC is 00:11:22:33:44:55"

With EVPN Type 2 Routes:
PE1: Learns Host B's MAC/IP binding
PE1: Advertises via BGP: "MAC 00:11:22:33:44:55 has IP 192.168.1.10"
PE2: Receives route, populates local ARP cache
Host A: "Who has 192.168.1.10?" --> PE2 responds locally!
```

EVPN Type 3 Routes: Inclusive Multicast

Type 3 routes solve the BUM (Broadcast, Unknown unicast, Multicast) traffic problem. They essentially say "I want to receive flooded traffic for this broadcast domain."

Type 3 Route Purpose:

```
Type 3 Route Says: "I have receivers in VLAN 100, send me BUM traffic"

Without Type 3:              With Type 3:
PE1 ---> PE2 (no hosts)      PE1 ---> PE2 (Type 3 present)
    ---> PE3 (no hosts)          -X-> PE3 (no Type 3)
    ---> PE4 (no hosts)          ---> PE4 (Type 3 present)

BUM sent everywhere!         BUM only to interested PEs!
```

How Type 3 Routes Work:

1. When a PE router has active interfaces in an EVI, it advertises a Type 3 route

2. This route includes a multicast tunnel endpoint (P-Multicast Service Interface)
3. Other PEs use this info to build their flood lists
4. If a PE withdraws its Type 3 route, others stop flooding to it

```
Type 3 Route Contents:
Route Type: 3 (Inclusive Multicast)
Route Distinguisher: 10.0.0.1:100
Ethernet Tag: 100
Originating Router IP: 10.0.0.1
P-Multicast Service Interface:
  Tunnel Type: Ingress Replication
  MPLS Label: 299792
```

Part 2: The Junos CLI Masterclass (The How)

Configuring EVPN with MAC Advertisement

Let's build a complete EVPN configuration focusing on Type 2 and Type 3 route generation:

```
# Complete PE1 Configuration for EVPN MAC Advertisement

# Step 1: Basic IP and MPLS configuration
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces ge-0/0/0 unit 0 description "To P-Router"
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/0 unit 0 family mpls

set protocols mpls interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0

# Step 2: OSPF for infrastructure reachability
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0

# Step 3: BGP configuration for EVPN
set routing-options router-id 10.0.0.1
set routing-options autonomous-system 65000

set protocols bgp group EVPN-RR type internal
set protocols bgp group EVPN-RR local-address 10.0.0.1
set protocols bgp group EVPN-RR family evpn signaling
set protocols bgp group EVPN-RR neighbor 10.0.0.100 description "Route-Reflector"

# Step 4: Global EVPN protocol configuration
set protocols evpn encapsulation mpls
set protocols evpn extended-vlan-list 100-199   # VLANs that will use EVPN

# Step 5: Customer-facing interface
set interfaces ge-0/0/5 flexible-vlan-tagging
set interfaces ge-0/0/5 encapsulation flexible-ethernet-services
set interfaces ge-0/0/5 unit 100 encapsulation vlan-bridge
set interfaces ge-0/0/5 unit 100 vlan-id 100

# Step 6: EVPN Instance Configuration
set routing-instances CUSTOMER-A instance-type evpn
set routing-instances CUSTOMER-A vlan-id 100
set routing-instances CUSTOMER-A interface ge-0/0/5.100
set routing-instances CUSTOMER-A route-distinguisher 10.0.0.1:100
set routing-instances CUSTOMER-A vrf-target target:65000:100

# Step 7: Enable EVPN protocols in the instance (generates Type 3)
```

```
set routing-instances CUSTOMER-A protocols evpn
set routing-instances CUSTOMER-A protocols evpn no-mac-learning  # Optional: disable data-plane learning

# Step 8: Optional - Configure MAC/IP learning for ARP suppression
set routing-instances CUSTOMER-A protocols evpn arp-suppression
```

Advanced Type 2 Route Control

```
# Control MAC advertisement behavior

# Advertise only specific MACs (MAC filtering)
set routing-instances CUSTOMER-A protocols evpn mac-list ALLOWED-MACS
set policy-options mac-list ALLOWED-MACS 00:11:22:33:44:55
set policy-options mac-list ALLOWED-MACS 00:aa:bb:cc:dd:ee

# Configure MAC mobility (track MAC moves)
set routing-instances CUSTOMER-A protocols evpn mac-mobility
set routing-instances CUSTOMER-A protocols evpn mac-mobility window-size 10
set routing-instances CUSTOMER-A protocols evpn mac-mobility detection-threshold 5

# Set MAC aging timer
set routing-instances CUSTOMER-A protocols evpn mac-aging-timer 1800

# Configure static MAC entries
set routing-instances CUSTOMER-A protocols evpn static-mac 00:99:88:77:66:55 interface ge-0/0/5.100
```

Type 3 Route Optimization

```
# Optimize BUM traffic handling

# Use ingress replication (default)
set routing-instances CUSTOMER-A protocols evpn ingress-replication

# Or use P2MP LSPs for efficient multicast
set routing-instances CUSTOMER-A protocols evpn inclusive-multicast
set routing-instances CUSTOMER-A protocols evpn inclusive-multicast p2mp-lsp-template P2MP-TEMPLATE

# Configure selective multicast for optimization
set routing-instances CUSTOMER-A protocols evpn selective-multicast
set routing-instances CUSTOMER-A protocols evpn selective-multicast group 224.1.1.0/24
```

Part 3: Verification & Troubleshooting (The What-If)

Verifying Type 2 Route Advertisement

1. **Check Local MAC Table:**

```
user@PE1> show evpn mac-table instance CUSTOMER-A

MAC address        Logical interface         Active source
00:11:22:33:44:55  ge-0/0/5.100             00:11:22:33:44:55
00:22:33:44:55:66  ge-0/0/5.100             00:22:33:44:55:66
```

2. **Verify Type 2 Route Advertisement:**

```
user@PE1> show route advertising-protocol bgp 10.0.0.100 match-prefix "2:*" detail

CUSTOMER-A.evpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 2:10.0.0.1:100::100::00:11:22:33:44:55/304 (1 entry, 1 announced)
  BGP group EVPN-RR type Internal
    Route Distinguisher: 10.0.0.1:100
```

```
    Route Label: 299776
    ESI: 00:00:00:00:00:00:00:00:00:00
    Nexthop: Self
    Localpref: 100
    AS path: [65000] I
    Communities: target:65000:100
```

3. **Check MAC/IP Bindings:**

```
user@PE1> show evpn arp-table instance CUSTOMER-A

EVPN ARP Table for Instance CUSTOMER-A
IP Address       MAC Address         Interface
192.168.1.10     00:11:22:33:44:55   ge-0/0/5.100
192.168.1.20     00:22:33:44:55:66   ge-0/0/5.100
```

Verifying Type 3 Route Operation

```
user@PE1> show route advertising-protocol bgp 10.0.0.100 match-prefix "3:*" detail

CUSTOMER-A.evpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 3:10.0.0.1:100::100::10.0.0.1/304 (1 entry, 1 announced)
  BGP group EVPN-RR type Internal
    Route Distinguisher: 10.0.0.1:100
    Route Label: 299792
    PMSI: Tunnel-type: Ingress Replication, label: 299792, tunnel-id: 10.0.0.1
    Nexthop: Self
    Localpref: 100
    AS path: [65000] I
    Communities: target:65000:100
```

Common Troubleshooting Scenarios

Scenario 1: MAC Addresses Not Being Advertised

Symptom: Local MACs are learned but no Type 2 routes are generated

Diagnostic Commands:

```
user@PE1> show evpn mac-table instance CUSTOMER-A
MAC address        Logical interface         Active source
00:11:22:33:44:55  ge-0/0/5.100             00:11:22:33:44:55

user@PE1> show route advertising-protocol bgp 10.0.0.100 match-prefix "2:*"
# No output!

user@PE1> show evpn instance CUSTOMER-A | match "Remote IRB"
  Remote IRB MAC advertisement: Disabled
```

Cause: EVPN protocols not enabled in the routing instance

Solution:

```
set routing-instances CUSTOMER-A protocols evpn
commit
```

Scenario 2: Type 3 Route Missing

Symptom: No BUM traffic received from remote PEs

Diagnostic Commands:

```
user@PE1> show route table CUSTOMER-A.evpn.0 match-prefix "3:*"

CUSTOMER-A.evpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

3:10.0.0.2:100::100::10.0.0.2/304
                   *[BGP/170] 00:05:32, localpref 100
                      AS path: I, validation-state: unverified
                    > to 10.1.1.2 via ge-0/0/0.0, Push 299888
# Missing local Type 3 route!
```

Cause: No active interfaces in the EVPN instance

Solution:

```
# Verify interface is active:
user@PE1> show interfaces ge-0/0/5.100 terse
Interface        Admin Link Proto    Local
ge-0/0/5.100     down  down vlan-bridge  # Problem: interface is down!

# Bring up the interface:
set interfaces ge-0/0/5 unit 100 disable false
commit
```

Scenario 3: ARP Suppression Not Working

Symptom: ARP broadcasts still flooding across the network

Diagnostic Commands:

```
user@PE1> show evpn arp-table instance CUSTOMER-A
# No output - no IP bindings!

user@PE1> show configuration routing-instances CUSTOMER-A protocols evpn | match arp
# No output
```

Cause: ARP suppression not configured

Solution:

```
set routing-instances CUSTOMER-A protocols evpn arp-suppression
commit

# Also verify IP addresses are being learned:
user@PE1> monitor traffic interface ge-0/0/5.100 matching "arp"
```

Scenario 4: MAC Mobility Causing Flapping

Symptom: MAC addresses constantly moving between PEs

Diagnostic Commands:

```
user@PE1> show evpn mac-table instance CUSTOMER-A mac-address 00:11:22:33:44:55

MAC address        Logical interface    Active source           Timestamp
00:11:22:33:44:55  ge-0/0/5.100        00:11:22:33:44:55       Dec 10 10:15:32
  MAC move count: 47        # High move count!

user@PE1> show log messages | match "MAC mobility"
Dec 10 10:15:32 PE1 evpn[1234]: EVPN_MAC_MOVE_THRESHOLD: MAC 00:11:22:33:44:55
exceeded move threshold (5) in window (10 seconds)
```

Cause: Layer 2 loop or duplicate MAC address

Solution:

```
# Configure MAC mobility protection:
set routing-instances CUSTOMER-A protocols evpn mac-mobility
set routing-instances CUSTOMER-A protocols evpn mac-mobility window-size 30
set routing-instances CUSTOMER-A protocols evpn mac-mobility detection-threshold 10
set routing-instances CUSTOMER-A protocols evpn mac-mobility suppress-threshold 20
commit

# This will blacklist MACs that move too frequently
```

Module 21: EVPN—Configuring a Single-Homed VLAN-Based EVI

Part 1: The Conceptual Lecture (The Why)

Understanding Single-Homed VLAN-Based EVPN

Let's start with the fundamentals. A single-homed VLAN-based EVI is the simplest form of EVPN deployment. Think of it as the EVPN equivalent of a basic VPLS instance, but with all the control-plane advantages we discussed.

What is Single-Homing?

Single-homing means each customer site connects to exactly one PE router. It's like having one road into your house - simple, straightforward, but with no redundancy.

```
Single-Homed Topology:
    Site A                  Site B                  Site C
   [Hosts]                 [Hosts]                 [Hosts]
      |                       |                       |
   [CE-A]                  [CE-B]                  [CE-C]
      |                       |                       |
   [PE-1]-------MPLS--------[PE-2]-------MPLS--------[PE-3]
           Core Network            Core Network
```

What is VLAN-Based EVI?

A VLAN-based EVI creates one EVPN instance per VLAN. Each VLAN gets its own:

- Broadcast domain
- MAC address table
- Route distinguisher
- Set of BGP routes

Think of it like creating separate virtual switches for each VLAN across your MPLS network:

```
VLAN-Based EVI Model:
PE-1                          PE-2                          PE-3
+-------------------+         +-------------------+         +-------------------+
| EVI-100 (VLAN100)|<-------->| EVI-100 (VLAN100)|<-------->| EVI-100 (VLAN100)|
| MAC Table VLAN100|   BGP    | MAC Table VLAN100|   BGP    | MAC Table VLAN100|
+-------------------+         +-------------------+         +-------------------+
| EVI-200 (VLAN200)|<-------->| EVI-200 (VLAN200)|<-------->| EVI-200 (VLAN200)|
| MAC Table VLAN200|   BGP    | MAC Table VLAN200|   BGP    | MAC Table VLAN200|
+-------------------+         +-------------------+         +-------------------+
```

Service Provider Network Requirements

Before configuring EVPN services, the SP network needs certain foundations:

1. **IP Reachability**: All PE routers must reach each other's loopbacks

2. **MPLS Transport**: Label switched paths between all PE routers
3. **BGP Peering**: iBGP sessions with EVPN address family enabled

```
Service Provider Core Prerequisites:
                          [Route Reflector]
                               10.0.0.100
                             BGP: EVPN AF
                           /      |        \
                         /        |          \
                       /          |            \
                [PE-1]------[P-Router]------[PE-2]
               10.0.0.1    MPLS Enabled    10.0.0.2
                 |                            |
             OSPF/IS-IS                   OSPF/IS-IS
                 |                            |
                LDP                          LDP
```

Why Start with Single-Homed VLAN-Based?

This is the "Hello World" of EVPN because:

1. **Simplicity**: No complex multihoming protocols
2. **Clarity**: One VLAN = One EVI (easy to understand)
3. **Foundation**: Teaches core EVPN concepts without added complexity
4. **Migration Path**: Easy upgrade from basic VPLS

Part 2: The Junos CLI Masterclass (The How)

Let's build a complete service provider network with EVPN from scratch.

Step 1: Configure the Service Provider Core

First, let's set up PE-1:

```
# PE-1 Base Configuration

# System basics
set system host-name PE-1
set system root-authentication encrypted-password "$ABC123"

# Interfaces
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces ge-0/0/0 unit 0 description "To P-Router"
set interfaces ge-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/0 unit 0 family mpls

# OSPF for core reachability
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p

# MPLS and LDP
set protocols mpls interface ge-0/0/0.0
set protocols ldp interface ge-0/0/0.0
set protocols ldp interface lo0.0

# BGP for EVPN
set routing-options router-id 10.0.0.1
set routing-options autonomous-system 65000

set protocols bgp group EVPN-CORE type internal
set protocols bgp group EVPN-CORE local-address 10.0.0.1
set protocols bgp group EVPN-CORE family evpn signaling
```

```
set protocols bgp group EVPN-CORE neighbor 10.0.0.2 description "To PE-2"
set protocols bgp group EVPN-CORE neighbor 10.0.0.3 description "To PE-3"
```

Step 2: Configure Route Reflector (Optional but Recommended)

For scalability, use a route reflector:

```
# Route Reflector Configuration

set routing-options router-id 10.0.0.100
set routing-options autonomous-system 65000

set protocols bgp group EVPN-RR-CLIENTS type internal
set protocols bgp group EVPN-RR-CLIENTS local-address 10.0.0.100
set protocols bgp group EVPN-RR-CLIENTS family evpn signaling
set protocols bgp group EVPN-RR-CLIENTS cluster 10.0.0.100
set protocols bgp group EVPN-RR-CLIENTS neighbor 10.0.0.1
set protocols bgp group EVPN-RR-CLIENTS neighbor 10.0.0.2
set protocols bgp group EVPN-RR-CLIENTS neighbor 10.0.0.3
```

Step 3: Configure EVPN Global Settings

On each PE router:

```
# EVPN Global Configuration

# Define encapsulation type
set protocols evpn encapsulation mpls

# Optional: Extended VLAN list for normalized VLAN mode
set protocols evpn extended-vlan-list 1-4094

# Optional: Global MAC aging
set protocols evpn mac-aging-timer 600
```

Step 4: Configure Customer-Facing Interface

```
# Customer Interface Configuration on PE-1

# Physical interface settings
set interfaces ge-0/0/5 description "To Customer A - Site 1"
set interfaces ge-0/0/5 flexible-vlan-tagging
set interfaces ge-0/0/5 encapsulation flexible-ethernet-services
set interfaces ge-0/0/5 mtu 9192

# Logical interface for VLAN 100
set interfaces ge-0/0/5 unit 100 description "Customer A - VLAN 100"
set interfaces ge-0/0/5 unit 100 encapsulation vlan-bridge
set interfaces ge-0/0/5 unit 100 vlan-id 100

# Optional: Rate limiting
set interfaces ge-0/0/5 unit 100 family bridge policer input LIMIT-100M
set interfaces ge-0/0/5 unit 100 family bridge policer output LIMIT-100M
```

Step 5: Create the EVPN Instance

Now the main configuration - the VLAN-based EVI:

```
# EVPN Instance Configuration

# Create the instance
```

```
set routing-instances CUSTOMER-A-VLAN100 instance-type evpn
set routing-instances CUSTOMER-A-VLAN100 description "Customer A - VLAN 100 Service"

# VLAN association
set routing-instances CUSTOMER-A-VLAN100 vlan-id 100

# Interface binding
set routing-instances CUSTOMER-A-VLAN100 interface ge-0/0/5.100

# Route distinguisher (must be unique per PE per EVI)
set routing-instances CUSTOMER-A-VLAN100 route-distinguisher 10.0.0.1:100

# Route target (same across all PEs for this EVI)
set routing-instances CUSTOMER-A-VLAN100 vrf-target target:65000:100

# Enable EVPN protocols
set routing-instances CUSTOMER-A-VLAN100 protocols evpn

# Optional: Interface MAC limit
set routing-instances CUSTOMER-A-VLAN100 protocols evpn interface ge-0/0/5.100 mac-limit 1000

# Optional: ARP suppression for efficiency
set routing-instances CUSTOMER-A-VLAN100 protocols evpn arp-suppression

# Optional: Disable unknown unicast flooding
set routing-instances CUSTOMER-A-VLAN100 protocols evpn no-unknown-unicast-flooding
```

Complete Configuration Example

Here's a complete PE configuration for reference:

```
# Complete PE-1 Configuration for Single-Homed VLAN-Based EVPN

system {
    host-name PE-1;
}

interfaces {
    ge-0/0/0 {
        unit 0 {
            description "To P-Router";
            family inet {
                address 10.1.1.1/30;
            }
            family mpls;
        }
    }
    ge-0/0/5 {
        description "To Customer A - Site 1";
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 100 {
            description "Customer A - VLAN 100";
            encapsulation vlan-bridge;
            vlan-id 100;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.0.0.1/32;
            }
```

```
        }
    }
}

routing-options {
    router-id 10.0.0.1;
    autonomous-system 65000;
}

protocols {
    mpls {
        interface ge-0/0/0.0;
    }
    ldp {
        interface ge-0/0/0.0;
        interface lo0.0;
    }
    ospf {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-0/0/0.0 {
                interface-type p2p;
            }
        }
    }
    bgp {
        group EVPN-CORE {
            type internal;
            local-address 10.0.0.1;
            family evpn {
                signaling;
            }
            neighbor 10.0.0.100 {
                description "To Route Reflector";
            }
        }
    }
    evpn {
        encapsulation mpls;
    }
}

routing-instances {
    CUSTOMER-A-VLAN100 {
        instance-type evpn;
        vlan-id 100;
        interface ge-0/0/5.100;
        route-distinguisher 10.0.0.1:100;
        vrf-target target:65000:100;
        protocols {
            evpn;
        }
    }
}
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. **Verify EVPN Instance Status:**

```
user@PE-1> show evpn instance CUSTOMER-A-VLAN100 extensive
Instance: CUSTOMER-A-VLAN100
  Route Distinguisher: 10.0.0.1:100
  VLAN ID: 100
  Per-instance MAC route label: 299776
  MAC database status                    Local  Remote
    Total MAC addresses:                     2       3
    Default gateway MAC addresses:           0       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                           Mode             Status
    ge-0/0/5.100    00:00:00:00:00:00:00:00:00:00  single-homed     Up
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 1
    VLAN ID  Intfs/up   Mode            MAC sync IM route label
    100          1/1    Extended        Enabled   299792
  Number of neighbors: 2
    Address            MAC    MAC+IP   AD       IM       ES
    10.0.0.2             2         2    0        1        0
    10.0.0.3             1         1    0        1        0
```

2. **Check BGP EVPN Routes:**

```
user@PE-1> show route table CUSTOMER-A-VLAN100.evpn.0

CUSTOMER-A-VLAN100.evpn.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:10.0.0.1:100::100::00:11:22:33:44:55/304
                   *[EVPN/170] 00:10:15
                      Indirect
2:10.0.0.1:100::100::00:11:22:33:44:56::192.168.1.10/304
                   *[EVPN/170] 00:10:15
                      Indirect
2:10.0.0.2:100::100::00:aa:bb:cc:dd:ee/304
                   *[BGP/170] 00:08:32, localpref 100
                      AS path: I, validation-state: unverified
                    > to 10.1.1.2 via ge-0/0/0.0, Push 299888
3:10.0.0.1:100::100::10.0.0.1/304
                   *[EVPN/170] 00:10:15
                      Indirect
3:10.0.0.2:100::100::10.0.0.2/304
                   *[BGP/170] 00:08:32, localpref 100
                      AS path: I, validation-state: unverified
                    > to 10.1.1.2 via ge-0/0/0.0, Push 299888
```

3. **Verify MAC Learning:**

```
user@PE-1> show evpn mac-table instance CUSTOMER-A-VLAN100

MAC address       Logical interface  Active source         Timestamp
00:11:22:33:44:55 ge-0/0/5.100      00:11:22:33:44:55     Dec 10 10:15:00
00:11:22:33:44:56 ge-0/0/5.100      00:11:22:33:44:56     Dec 10 10:15:15
00:aa:bb:cc:dd:ee                   10.0.0.2              Dec 10 10:18:22
00:aa:bb:cc:dd:ef                   10.0.0.2              Dec 10 10:18:45
00:99:88:77:66:55                   10.0.0.3              Dec 10 10:19:13
```

Common Troubleshooting Scenarios

Scenario 1: EVPN Instance Not Coming Up

Symptom: No routes in EVPN table, instance shows as down

Diagnostic Commands:

```
user@PE-1> show evpn instance CUSTOMER-A-VLAN100
Instance: CUSTOMER-A-VLAN100
  Route Distinguisher: 10.0.0.1:100
  Number of bridge domains: 0     # Problem: No bridge domain!

user@PE-1> show interfaces ge-0/0/5.100
error: device ge-0/0/5.100 not found
```

Cause: Interface misconfiguration

Solution:

```
# Check interface configuration:
show configuration interfaces ge-0/0/5

# Common fixes:
set interfaces ge-0/0/5 flexible-vlan-tagging
set interfaces ge-0/0/5 encapsulation flexible-ethernet-services
set interfaces ge-0/0/5 unit 100 encapsulation vlan-bridge
set interfaces ge-0/0/5 unit 100 vlan-id 100
commit
```

Scenario 2: No Remote MACs Learned

Symptom: Only local MACs visible, no remote MACs

Diagnostic Commands:

```
user@PE-1> show route receive-protocol bgp 10.0.0.100 table bgp.evpn.0
bgp.evpn.0: 0 destinations, 0 routes (0 active, 0 holddown, 0 hidden)

user@PE-1> show bgp neighbor 10.0.0.100 | match "Last traffic"
  Last traffic (seconds): Received 5    Sent 5    Checked 5
```

Cause: BGP EVPN signaling not enabled

Solution:

```
# On both local PE and RR:
set protocols bgp group EVPN-CORE family evpn signaling
commit

# Verify:
show bgp neighbor 10.0.0.100 | match "NLRI.*evpn"
  NLRI for this session: evpn
```

Scenario 3: Traffic Not Forwarding

Symptom: MACs learned but ping fails between sites

Diagnostic Commands:

```
user@PE-1> show route forwarding-table family bridge
Routing table: CUSTOMER-A-VLAN100.bridge
Destination   Type   Next hop            Index    NhRef Netif
00:aa:bb:*    user   indr                1048574      3
                     Push 299888        573          ge-0/0/0.0

user@PE-1> show ldp neighbor
Address         Interface       Label space ID      Hold time
```

```
10.1.1.2        ge-0/0/0.0      10.0.0.2:0          11
# Missing LDP session to remote PE!
```

Cause: MPLS transport issue

Solution:

```
# Verify MPLS is enabled:
show configuration protocols mpls
show configuration protocols ldp

# Check for firewall filters blocking LDP:
show configuration interfaces ge-0/0/0 unit 0 family inet filter

# Ensure LDP is running on all core interfaces:
set protocols ldp interface ge-0/0/0.0
commit
```

Scenario 4: Duplicate MACs from Different Sites

Symptom: MAC address appearing from multiple PEs

Diagnostic Commands:

```
user@PE-1> show evpn mac-table instance CUSTOMER-A-VLAN100 mac-address 00:11:22:33:44:55
MAC address       Logical interface  Active source     Timestamp
00:11:22:33:44:55 ge-0/0/5.100      00:11:22:33:44:55 Dec 10 10:15:00
00:11:22:33:44:55                   10.0.0.2          Dec 10 10:15:05  # Duplicate!

user@PE-1> show evpn mac-table instance CUSTOMER-A-VLAN100 mac-address 00:11:22:33:44:55 extensive
...
  Sequence number: 5        # Local
  Sequence number: 3        # Remote - Lower wins!
```

Cause: Same MAC connected to multiple sites (loop or misconfiguration)

Solution:

```
# Enable MAC mobility detection:
set routing-instances CUSTOMER-A-VLAN100 protocols evpn mac-mobility
set routing-instances CUSTOMER-A-VLAN100 protocols evpn mac-mobility window-size 60
set routing-instances CUSTOMER-A-VLAN100 protocols evpn mac-mobility detection-threshold 5
set routing-instances CUSTOMER-A-VLAN100 protocols evpn mac-mobility suppress-threshold 10
commit

# Find and fix the duplicate MAC source
```

Module 22: EVPN—Configuring a Single-Homed VLAN-Aware Bundle EVI

Part 1: The Conceptual Lecture (The Why)

Evolution from VLAN-Based to VLAN-Aware Bundle

In Module 21, we created separate EVPN instances for each VLAN. But what if a customer has 50 VLANs? Creating 50 separate EVIs becomes operationally complex. This is where VLAN-aware bundle service comes in.

The Scalability Challenge:

```
VLAN-Based Approach (Previous Module):
Customer has VLANs 100, 200, 300, 400, 500
```

```
Required Configuration:
- 5 routing instances (CUST-VLAN100, CUST-VLAN200, etc.)
- 5 route distinguishers
- 5 BGP route targets
- 5 x 3 = 15 BGP routes per MAC (Type 2 routes for each VLAN)

Management overhead grows linearly with VLANs!
```

VLAN-Aware Bundle Solution:

```
VLAN-Aware Bundle Approach:
Customer has VLANs 100, 200, 300, 400, 500

Required Configuration:
- 1 routing instance (CUST-BUNDLE)
- 1 route distinguisher
- 1 BGP route target
- Ethernet tags differentiate VLANs within the bundle

All VLANs managed as one entity!
```

Understanding VLAN-Aware Bundle Concepts

What is a VLAN Bundle?

A VLAN bundle groups multiple VLANs into a single EVPN instance. It's like having one large virtual switch that's VLAN-aware, rather than multiple separate virtual switches.

```
Conceptual Difference:

VLAN-Based (Multiple Instances):        VLAN-Aware Bundle (Single Instance):
┌─────────────┐ ┌─────────────┐         ┌──────────────────────────────┐
│ EVI VLAN100 │ │ EVI VLAN200 │         │    EVI VLAN-BUNDLE           │
│ MAC Table   │ │ MAC Table   │         │  ┌──────────┬─────────────┐  │
│ RD:1:100    │ │ RD:1:200    │         │  │ VLAN100  │  VLAN200    │  │
│ RT:65000:100│ │ RT:65000:200│         │  │ MACs     │  MACs       │  │
└─────────────┘ └─────────────┘         │  └──────────┴─────────────┘  │
                                        │   RD: 1:1000                 │
                                        │   RT: 65000:1000             │
                                        └──────────────────────────────┘
```

Ethernet Tags in VLAN-Aware Mode:

In VLAN-aware bundle, the Ethernet Tag field in EVPN routes carries the VLAN ID. This allows a single EVI to handle multiple VLANs:

```
EVPN Route Structure Comparison:

VLAN-Based Type 2 Route:
RD: 10.0.0.1:100    <-- Unique per VLAN
Ethernet Tag: 0     <-- Not used
MAC: 00:11:22:33:44:55
Label: 299776

VLAN-Aware Bundle Type 2 Route:
RD: 10.0.0.1:1000   <-- Same for all VLANs
Ethernet Tag: 100   <-- VLAN ID goes here!
MAC: 00:11:22:33:44:55
Label: 299776
```

Benefits of VLAN-Aware Bundle

1. **Operational Simplicity**: One instance manages many VLANs

2. **BGP Scale**: Fewer BGP sessions and route targets
3. **Flexible Service**: Easy to add/remove VLANs
4. **Resource Efficiency**: Less memory and CPU usage

```
Scaling Comparison:
                    100 VLANs with:
                VLAN-Based          VLAN-Aware Bundle
Routing Instances:   100                    1
Route Targets:       100                    1
BGP Peers:          100 x n                1 x n
Config Lines:       ~2000                  ~50
```

When to Use VLAN-Aware Bundle

Use VLAN-aware bundle when:

- Customer has multiple VLANs (>3-5)
- VLANs belong to the same administrative domain
- You want simplified operations
- VLANs can share the same flooding domain

Stay with VLAN-based when:

- Complete VLAN isolation is required
- Different SLA per VLAN
- Complex per-VLAN policies needed
- Only 1-2 VLANs per customer

Part 2: The Junos CLI Masterclass (The How)

VLAN-Aware Bundle Configuration Structure

The configuration is similar to VLAN-based, but with key differences:

```
# VLAN-Aware Bundle Configuration on PE-1

# Step 1: Customer-facing interface (supports multiple VLANs)
set interfaces ge-0/0/5 description "To Customer A - Multiple VLANs"
set interfaces ge-0/0/5 flexible-vlan-tagging
set interfaces ge-0/0/5 encapsulation flexible-ethernet-services
set interfaces ge-0/0/5 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-0/0/5 unit 0 family ethernet-switching vlan members CUST-VLANS

# Step 2: Define the VLAN list
set vlans CUST-VLANS vlan-id-list [100 200 300 400 500]

# Step 3: Create VLAN-Aware Bundle EVPN Instance
set routing-instances CUSTOMER-A-BUNDLE instance-type evpn
set routing-instances CUSTOMER-A-BUNDLE description "Customer A - All VLANs"

# Key difference: vlan-aware service type
set routing-instances CUSTOMER-A-BUNDLE service-type vlan-aware

# Step 4: Interface binding (uses unit 0 for all VLANs)
set routing-instances CUSTOMER-A-BUNDLE interface ge-0/0/5.0

# Step 5: Route distinguisher (one for all VLANs)
set routing-instances CUSTOMER-A-BUNDLE route-distinguisher 10.0.0.1:1000

# Step 6: Route target (one for all VLANs)
set routing-instances CUSTOMER-A-BUNDLE vrf-target target:65000:1000

# Step 7: VLAN configuration within the bundle
```

```
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN200 vlan-id 200
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN300 vlan-id 300
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN400 vlan-id 400
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN500 vlan-id 500

# Step 8: Enable EVPN protocols
set routing-instances CUSTOMER-A-BUNDLE protocols evpn
```

Advanced VLAN-Aware Bundle Features

```
# Per-VLAN features within the bundle

# Different MAC aging per VLAN
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 mac-aging-time 300
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN200 mac-aging-time 600

# MAC limiting per VLAN
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 mac-table-size 1000
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN200 mac-table-size 2000

# Interface MAC limits
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 interface ge-0/0/5.0 mac-limit 100

# VLAN translation (normalize customer VLANs)
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 translated-vlan-id 1100

# IRB interfaces for inter-VLAN routing
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 l3-interface irb.100
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN200 l3-interface irb.200

# Configure IRB interfaces
set interfaces irb unit 100 family inet address 192.168.100.1/24
set interfaces irb unit 200 family inet address 192.168.200.1/24
```

Complete VLAN-Aware Bundle Configuration

Here's a working example with multiple features:

```
# Complete PE-1 Configuration for VLAN-Aware Bundle EVPN

# Customer-facing interface
interfaces {
    ge-0/0/5 {
        description "Customer A - VLAN Bundle";
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ 100 200 300 ];
                }
            }
        }
    }
}

# EVPN Instance Configuration
routing-instances {
    CUSTOMER-A-BUNDLE {
```

```
        instance-type evpn;
        service-type vlan-aware;
        interface ge-0/0/5.0;
        route-distinguisher 10.0.0.1:1000;
        vrf-target target:65000:1000;
        protocols {
            evpn {
                arp-suppression;
                duplicate-mac-detection {
                    detection-threshold 5;
                    detection-window 60;
                }
            }
        }
        vlans {
            VLAN100 {
                vlan-id 100;
                mac-aging-time 300;
                mac-table-size 1000;
            }
            VLAN200 {
                vlan-id 200;
                mac-aging-time 600;
                mac-table-size 2000;
            }
            VLAN300 {
                vlan-id 300;
                ## IRB for inter-VLAN routing
                l3-interface irb.300;
            }
        }
    }
}

# IRB interface for VLAN 300
interfaces {
    irb {
        unit 300 {
            family inet {
                address 192.168.30.1/24;
            }
        }
    }
}
```

Migration from VLAN-Based to VLAN-Aware

If migrating existing VLAN-based EVIs:

```
# Step 1: Create new VLAN-aware bundle instance
set routing-instances CUSTOMER-A-NEW instance-type evpn
set routing-instances CUSTOMER-A-NEW service-type vlan-aware
set routing-instances CUSTOMER-A-NEW route-distinguisher 10.0.0.1:9999
set routing-instances CUSTOMER-A-NEW vrf-target target:65000:9999

# Step 2: Add VLANs to the bundle
set routing-instances CUSTOMER-A-NEW vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A-NEW vlans VLAN200 vlan-id 200

# Step 3: Move interfaces (during maintenance window)
delete routing-instances CUSTOMER-A-VLAN100 interface ge-0/0/5.100
delete routing-instances CUSTOMER-A-VLAN200 interface ge-0/0/5.200
```

```
set routing-instances CUSTOMER-A-NEW interface ge-0/0/5.0

# Step 4: Clean up old instances
delete routing-instances CUSTOMER-A-VLAN100
delete routing-instances CUSTOMER-A-VLAN200
```

Part 3: Verification & Troubleshooting (The What-If)

Verification Commands for VLAN-Aware Bundle

1. **Check Bundle Instance Status:**

```
user@PE-1> show evpn instance CUSTOMER-A-BUNDLE extensive
Instance: CUSTOMER-A-BUNDLE
  Route Distinguisher: 10.0.0.1:1000
  Service Type: VLAN-aware
  MAC database status                     Local  Remote
    Total MAC addresses:                     15      22
    Default gateway MAC addresses:            1       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                             Mode          Status
    ge-0/0/5.0      00:00:00:00:00:00:00:00:00:00  single-homed  Up
  Number of VLANs: 3
    VLAN  Intfs/up  RTT        IRB intf   Mode            MAC sync
    100       1/1   65000:1000            Extended        Enabled
    200       1/1   65000:1000            Extended        Enabled
    300       1/1   65000:1000  irb.300   Extended        Enabled
```

2. **View per-VLAN MAC tables:**

```
user@PE-1> show evpn mac-table instance CUSTOMER-A-BUNDLE vlan-id 100

MAC address        Active source      Logical interface
00:11:22:33:44:55  00:11:22:33:44:55  ge-0/0/5.0
00:11:22:33:44:56  00:11:22:33:44:56  ge-0/0/5.0
00:aa:bb:cc:dd:ee  10.0.0.2

user@PE-1> show evpn mac-table instance CUSTOMER-A-BUNDLE vlan-id 200

MAC address        Active source      Logical interface
00:22:33:44:55:66  00:22:33:44:55:66  ge-0/0/5.0
00:bb:cc:dd:ee:ff  10.0.0.2
```

3. **Check EVPN routes with Ethernet tags:**

```
user@PE-1> show route advertising-protocol bgp 10.0.0.100 match-prefix "2:*" detail

CUSTOMER-A-BUNDLE.evpn.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
* 2:10.0.0.1:1000::100::00:11:22:33:44:55/304 (1 entry, 1 announced)
  BGP group EVPN-CORE type Internal
    Route Distinguisher: 10.0.0.1:1000
    Route Label: 299776
    ESI: 00:00:00:00:00:00:00:00:00:00
    Ethernet Tag: 100                    <-- VLAN 100
    Nexthop: Self
    Communities: target:65000:1000

* 2:10.0.0.1:1000::200::00:22:33:44:55:66/304 (1 entry, 1 announced)
  BGP group EVPN-CORE type Internal
    Route Distinguisher: 10.0.0.1:1000
    Route Label: 299776
```

```
    ESI: 00:00:00:00:00:00:00:00:00:00
    Ethernet Tag: 200                   <-- VLAN 200
    Nexthop: Self
    Communities: target:65000:1000
```

Common Troubleshooting Scenarios

Scenario 1: VLANs Not Visible in Instance

Symptom: Configured VLANs don't appear in the EVPN instance

Diagnostic Commands:

```
user@PE-1> show evpn instance CUSTOMER-A-BUNDLE | match VLAN
  Number of VLANs: 0    # No VLANs!

user@PE-1> show configuration routing-instances CUSTOMER-A-BUNDLE
instance-type evpn;
interface ge-0/0/5.0;
# Missing service-type!
```

Cause: Missing service-type configuration

Solution:

```
set routing-instances CUSTOMER-A-BUNDLE service-type vlan-aware
commit

# Also verify VLANs are configured:
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN200 vlan-id 200
commit
```

Scenario 2: MACs Learned in Wrong VLAN

Symptom: MAC addresses appearing in incorrect VLAN

Diagnostic Commands:

```
user@PE-1> monitor traffic interface ge-0/0/5.0 layer2-headers
10:15:32.123456 In  VLAN100: 00:11:22:33:44:55 > ff:ff:ff:ff:ff:ff, ethertype ARP

user@PE-1> show evpn mac-table instance CUSTOMER-A-BUNDLE mac-address 00:11:22:33:44:55
VLAN  MAC address        Active source
200   00:11:22:33:44:55  ge-0/0/5.0       # Wrong VLAN!
```

Cause: Interface VLAN configuration mismatch

Solution:

```
# Check interface VLAN membership:
show configuration interfaces ge-0/0/5 unit 0

# Ensure correct VLAN list:
set interfaces ge-0/0/5 unit 0 family ethernet-switching vlan members [100 200 300]
commit

# Or use vlan-id-list:
set vlans CUST-VLANS vlan-id-list [100 200 300]
set interfaces ge-0/0/5 unit 0 family ethernet-switching vlan members CUST-VLANS
```

Scenario 3: Inter-VLAN Routing Not Working

Symptom: Cannot ping between VLANs despite IRB configuration

Diagnostic Commands:

```
user@PE-1> show interfaces irb.300 terse
Interface        Admin Link Proto    Local
irb.300          up    down inet     192.168.30.1/24

user@PE-1> show evpn instance CUSTOMER-A-BUNDLE extensive | find "VLAN  "
VLAN  Intfs/up  RTT        IRB intf   Mode
300       1/1   65000:1000            Extended    # No IRB binding!
```

Cause: IRB interface not bound to VLAN

Solution:

```
set routing-instances CUSTOMER-A-BUNDLE vlans VLAN300 l3-interface irb.300
commit

# Verify IRB is up:
show interfaces irb.300
show route table CUSTOMER-A-BUNDLE.evpn.0 match-prefix "2:*:*:*:*:*:192.168.30.1"
```

Scenario 4: Bundle RT Conflicts with VLAN-Based EVIs

Symptom: Receiving routes from unrelated EVPN instances

Diagnostic Commands:

```
user@PE-1> show route receive-protocol bgp 10.0.0.2 table CUSTOMER-A-BUNDLE.evpn.0 | match RD
     Route Distinguisher: 10.0.0.2:100     # Different customer!
     Route Distinguisher: 10.0.0.2:200     # Different customer!
     Route Distinguisher: 10.0.0.2:1000    # Correct

user@PE-1> show route receive-protocol bgp 10.0.0.2 table CUSTOMER-A-BUNDLE.evpn.0 detail | match Communities
     Communities: target:65000:100 target:65000:1000   # Multiple RTs!
```

Cause: Overlapping route targets between different services

Solution:

```
# Use unique route targets per service:
set routing-instances CUSTOMER-A-BUNDLE vrf-target target:65000:1000

# For very specific control, use import/export policies:
set policy-options policy-statement BUNDLE-IMPORT term 1 from community BUNDLE-RT
set policy-options policy-statement BUNDLE-IMPORT term 1 then accept
set policy-options policy-statement BUNDLE-IMPORT term 2 then reject

set policy-options community BUNDLE-RT members target:65000:1000

set routing-instances CUSTOMER-A-BUNDLE vrf-import BUNDLE-IMPORT
commit
```

Module 23: EVPN—Multihoming Configuration and Type 4 Routes

Part 1: The Conceptual Lecture (The Why)

The Need for Multihoming

So far, we've only connected each customer site to a single PE router. But what happens when that PE fails? The entire site loses connectivity. Multihoming solves this by connecting each site to multiple PE routers.

```
Single-Homed (Previous Modules):        Multihomed (This Module):
        Site A                                  Site A
       [Hosts]                                 [Hosts]
          |                                       |
       [CE-A] ← Single point of failure        [CE-A]
          |                                     /    \
       [PE-1]                                [PE-1]  [PE-2] ← Redundancy!
          |                                    |        |
    [MPLS Core]                             [MPLS Core]
```

The Multihoming Challenge

Connecting a CE to multiple PEs creates immediate challenges:

1. **Layer 2 Loops**: Both PEs could forward the same broadcast, creating loops
2. **Duplicate Packets**: Unicast traffic might be duplicated
3. **MAC Flapping**: Remote PEs see the same MAC from multiple next-hops
4. **Load Balancing**: How to use both links efficiently?

```
The Loop Problem:
Step 1: CE sends broadcast frame
Step 2: PE-1 receives it, floods to PE-2 and PE-3
Step 3: PE-2 receives it from CE AND from PE-1
Step 4: PE-2 forwards PE-1's copy back to CE
Step 5: CE forwards it to PE-1 again... LOOP!
```

EVPN's Multihoming Solution

EVPN solves these challenges using three key mechanisms:

1. **Ethernet Segment Identifier (ESI)**: Identifies which PEs connect to the same CE
2. **Designated Forwarder Election**: Only one PE forwards BUM traffic
3. **Split Horizon**: PEs don't forward traffic between each other for the same ES

Understanding Ethernet Segments (ES)

An Ethernet Segment represents the connection between a CE and multiple PEs. It's identified by a 10-byte ESI (Ethernet Segment Identifier).

```
Ethernet Segment Concept:
                    ESI: 00:11:22:33:44:55:66:77:88:99
                    +---------------------------+
                    |    Ethernet Segment       |
                    |  ┌─────┐       ┌─────┐    |
                    |  │ PE-1│       │ PE-2│    |
                    |  └──┬──┘       └──┬──┘    |
                    |     │     ┌─┐     │       |
                    |     └─────┤CE├─────┘       |
                    |           └─┘             |
                    +---------------------------+

All connections to the same CE share the same ESI
```

ESI Assignment Methods:

1. **Type 0 (Manual)**: Administrator configures identical ESI on all PEs
2. **Type 1 (Auto-LACP)**: Derived from LACP system ID
3. **Type 3 (Auto-MAC)**: Derived from CE's MAC address

Enter Type 4 Routes: Ethernet Segment Routes

Type 4 routes are the cornerstone of EVPN multihoming. They serve two critical functions:

1. **ES Discovery**: PEs discover who else is connected to the same segment
2. **DF Election**: PEs elect a designated forwarder for BUM traffic

```
Type 4 Route Contents:
Route Type: 4 (Ethernet Segment)
Route Distinguisher: 10.0.0.1:32767    # Special RD format
ESI: 00:11:22:33:44:55:66:77:88:99
Originating Router IP: 10.0.0.1        # Who sent this
ES-Import RT: target:es:00:11:22:33:44:55:66:77:88:99
```

Designated Forwarder (DF) Election

For each VLAN on an Ethernet Segment, one PE is elected as the Designated Forwarder. Only the DF forwards BUM traffic to the CE.

DF Election Algorithm:

```
# Simplified DF election logic
def calculate_df(vlan_id, pe_list):
    # Sort PE list by IP (lowest to highest)
    sorted_pes = sorted(pe_list)

    # Use modulo to distribute VLANs across PEs
    df_index = vlan_id % len(sorted_pes)

    return sorted_pes[df_index]

# Example:
PEs = [10.0.0.1, 10.0.0.2]  # PE-1 and PE-2
VLAN 100: 100 % 2 = 0 → PE-1 (10.0.0.1) is DF
VLAN 101: 101 % 2 = 1 → PE-2 (10.0.0.2) is DF
```

This provides automatic load balancing of BUM traffic across VLANs!

Split Horizon and Loop Prevention

EVPN uses split horizon with Type 4 routes to prevent loops:

```
Split Horizon Rule:
"If a PE receives a packet from the core with an ESI label
 matching a local Ethernet Segment, drop the packet"

Example Flow:
1. CE sends frame to PE-1
2. PE-1 floods to core with ESI label 1000
3. PE-2 receives frame with ESI label 1000
4. PE-2 checks: "Do I have ES with label 1000?" → Yes
5. PE-2 drops frame (split horizon)
```

Part 2: The Junos CLI Masterclass (The How)

Configuring Multihomed EVPN

Let's configure a complete multihomed EVPN setup:

Step 1: Configure CE for Multihoming (LACP)

```
# CE Configuration (Device connecting to multiple PEs)

# Create aggregated Ethernet interface
set interfaces ae0 description "To PE-1 and PE-2"
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:11:22:33:44:55
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
```

```
set interfaces ae0 unit 0 family ethernet-switching vlan members 100-200

# Add physical interfaces to bundle
set interfaces ge-0/0/0 description "To PE-1"
set interfaces ge-0/0/0 ether-options 802.3ad ae0
set interfaces ge-0/0/1 description "To PE-2"
set interfaces ge-0/0/1 ether-options 802.3ad ae0
```

Step 2: Configure PE-1 for Multihoming

```
# PE-1 Configuration

# Configure interface to CE with ESI
set interfaces ge-0/0/5 description "To CE-A (multihomed with PE-2)"
set interfaces ge-0/0/5 flexible-vlan-tagging
set interfaces ge-0/0/5 encapsulation flexible-ethernet-services

# KEY: Configure ESI (manual method)
set interfaces ge-0/0/5 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ge-0/0/5 esi all-active

# Configure logical interface
set interfaces ge-0/0/5 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-0/0/5 unit 0 family ethernet-switching vlan members 100-200

# EVPN Instance configuration (similar to before)
set routing-instances CUSTOMER-A instance-type evpn
set routing-instances CUSTOMER-A service-type vlan-aware
set routing-instances CUSTOMER-A interface ge-0/0/5.0
set routing-instances CUSTOMER-A route-distinguisher 10.0.0.1:1000
set routing-instances CUSTOMER-A vrf-target target:65000:1000
set routing-instances CUSTOMER-A protocols evpn

# Configure VLANs
set routing-instances CUSTOMER-A vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A vlans VLAN200 vlan-id 200
```

Step 3: Configure PE-2 (Identical ESI)

```
# PE-2 Configuration (mirror of PE-1 with same ESI)

set interfaces ge-0/0/7 description "To CE-A (multihomed with PE-1)"
set interfaces ge-0/0/7 flexible-vlan-tagging
set interfaces ge-0/0/7 encapsulation flexible-ethernet-services

# CRITICAL: Same ESI as PE-1
set interfaces ge-0/0/7 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ge-0/0/7 esi all-active

set interfaces ge-0/0/7 unit 0 family ethernet-switching interface-mode trunk
set interfaces ge-0/0/7 unit 0 family ethernet-switching vlan members 100-200

# EVPN Instance (same RT as PE-1)
set routing-instances CUSTOMER-A instance-type evpn
set routing-instances CUSTOMER-A service-type vlan-aware
set routing-instances CUSTOMER-A interface ge-0/0/7.0
set routing-instances CUSTOMER-A route-distinguisher 10.0.0.2:1000
set routing-instances CUSTOMER-A vrf-target target:65000:1000
set routing-instances CUSTOMER-A protocols evpn
```

```
set routing-instances CUSTOMER-A vlans VLAN100 vlan-id 100
set routing-instances CUSTOMER-A vlans VLAN200 vlan-id 200
```

Automatic ESI Generation

Instead of manual ESI, you can use automatic generation:

```
# LACP-based ESI (Type 1)
set interfaces ae0 esi auto-generate
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:11:22:33:44:55

# Single-homed CE with LAG to PE
set interfaces ae0 esi single-active      # For active-standby
set interfaces ae0 esi all-active         # For active-active
```

Advanced Multihoming Features

```
# DF Election Preference (influence which PE becomes DF)
set interfaces ge-0/0/5 designated-forwarder-preference value 100

# Fast Convergence
set routing-instances CUSTOMER-A protocols evpn fast-reroute

# MAC Flush on Topology Change
set routing-instances CUSTOMER-A protocols evpn mac-flush-on-topology-change

# Enhanced DF Election Algorithm
set routing-instances CUSTOMER-A protocols evpn designated-forwarder-election algorithm preference-based

# Source MAC Protection (prevent CE from using PE MACs)
set interfaces ge-0/0/5 source-mac-validation
```

Part 3: Verification & Troubleshooting (The What-If)

Verifying Multihoming Operation

1. **Check Ethernet Segment Status:**

```
user@PE-1> show evpn ethernet-segment
ESI: 00:11:22:33:44:55:66:77:88:99
  Mode: all-active
  Status: Up
  Local interface: ge-0/0/5.0, Status: Up
  Number of remote PEs connected: 1
    Remote PE         MAC label  Aliasing label  Mode
    10.0.0.2          300000     300001          all-active
  Designated Forwarder: 10.0.0.1
  DF Candidates: 10.0.0.1, 10.0.0.2
```

2. **Verify Type 4 Route Advertisement:**

```
user@PE-1> show route advertising-protocol bgp 10.0.0.100 match-prefix "4:*" detail

bgp.evpn.0: 25 destinations, 25 routes (25 active, 0 holddown, 0 hidden)
* 4:10.0.0.1:32767::00112233445566778899::10.0.0.1/304 (1 entry, 1 announced)
  BGP group EVPN-CORE type Internal
    Route Distinguisher: 10.0.0.1:32767
    Nexthop: Self
    Communities: target:es:00:11:22:33:44:55:66:77:88:99
    ES-Import: target:es:00:11:22:33:44:55:66:77:88:99
```

3. **Check DF Election Results:**

```
user@PE-1> show evpn ethernet-segment designated-forwarder
ESI: 00:11:22:33:44:55:66:77:88:99
  VLAN     DF Election   DF             Backup         Preference
  100      Complete      10.0.0.1       10.0.0.2       100
  200      Complete      10.0.0.2       10.0.0.1       100
```

4. **Verify Split Horizon Labels:**

```
user@PE-1> show route table mpls.0 label 300000

mpls.0: 50 destinations, 50 routes (50 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

300000             *[EVPN/7] 00:45:12
                      to table __SPLIT_HORIZON__.mpls.0
```

Common Troubleshooting Scenarios

Scenario 1: DF Election Not Completing

Symptom: No designated forwarder elected, BUM traffic dropped

Diagnostic Commands:

```
user@PE-1> show evpn ethernet-segment designated-forwarder
ESI: 00:11:22:33:44:55:66:77:88:99
  VLAN     DF Election   DF             Backup
  100      In-Progress   None           None

user@PE-1> show route receive-protocol bgp 10.0.0.2 match-prefix "4:*"
# No Type 4 routes from PE-2!
```

Cause: PE-2 not advertising Type 4 route (ESI mismatch or BGP issue)

Solution:

```
# On PE-2, verify ESI configuration:
show configuration interfaces ge-0/0/7 | match esi

# Ensure ESI matches exactly:
set interfaces ge-0/0/7 esi 00:11:22:33:44:55:66:77:88:99
commit

# Check BGP EVPN is enabled:
set protocols bgp group EVPN-CORE family evpn signaling
```

Scenario 2: Both PEs Forwarding BUM Traffic

Symptom: Duplicate broadcasts received at CE

Diagnostic Commands:

```
user@PE-1> show evpn ethernet-segment designated-forwarder
ESI: 00:11:22:33:44:55:66:77:88:99
  VLAN     DF Election   DF             Backup
  100      Complete      10.0.0.1       10.0.0.2    # PE-1 is DF

user@PE-1> monitor traffic interface ge-0/0/5.0 extensive matching "broadcast"
# Shows BUM traffic being sent
```

```
user@PE-2> monitor traffic interface ge-0/0/7.0 extensive matching "broadcast"
# Also shows BUM traffic being sent! (Should be blocked)
```

Cause: DF filter not working or split horizon disabled

Solution:

```
# Verify EVPN is handling the interface:
user@PE-2> show evpn instance CUSTOMER-A | match ge-0/0/7
    ge-0/0/7.0      00:11:22:33:44:55:66:77:88:99  all-active   Up

# Check for manual override:
show configuration routing-instances CUSTOMER-A | match designated-forwarder

# Ensure no "no-df-election" is configured:
delete routing-instances CUSTOMER-A protocols evpn no-df-election
commit
```

Scenario 3: Traffic Blackholed After Link Failure

Symptom: When one PE-CE link fails, traffic is lost

Diagnostic Commands:

```
user@PE-1> show interfaces ge-0/0/5 terse
Interface        Admin Link Proto
ge-0/0/5         up    down        # Link failed!

user@PE-2> show evpn ethernet-segment
ESI: 00:11:22:33:44:55:66:77:88:99
  Status: Up
  Number of remote PEs connected: 1    # Still thinks PE-1 is up!
```

Cause: Slow EVPN ES failure detection

Solution:

```
# Enable fast failure detection:
set protocols evpn fast-reroute

# Configure BFD on core links:
set protocols bgp group EVPN-CORE neighbor 10.0.0.1 bfd-liveness-detection minimum-interval 300
set protocols bgp group EVPN-CORE neighbor 10.0.0.1 bfd-liveness-detection multiplier 3

# Use Ethernet OAM for CE link monitoring:
set interfaces ge-0/0/5 oam ethernet link-fault-management
```

Scenario 4: MAC Mobility Issues in Multihomed Setup

Symptom: Remote PEs see constant MAC moves between PE-1 and PE-2

Diagnostic Commands:

```
user@PE-3> show evpn mac-table instance CUSTOMER-A
MAC address        Active source     Timestamp
00:aa:bb:cc:dd:ee  10.0.0.1         10:15:01
00:aa:bb:cc:dd:ee  10.0.0.2         10:15:02  # Flapping!

user@PE-3> show evpn mac-table extensive mac-address 00:aa:bb:cc:dd:ee
  Sequence number: 10 (from PE-1)
  Sequence number: 11 (from PE-2)
  Sequence number: 12 (from PE-1)  # Constant updates
```

Cause: Both PEs learning and advertising the same MAC

Solution:

```
# This is normal for all-active multihoming
# Remote PEs should see both and load-balance

# Verify aliasing is working:
user@PE-3> show route table CUSTOMER-A.evpn.0 match-prefix "1:*" detail
# Should see Type 1 per-EVI routes from both PEs

# If causing issues, can rate-limit MAC advertisements:
set routing-instances CUSTOMER-A protocols evpn mac-mobility
set routing-instances CUSTOMER-A protocols evpn mac-mobility detection-threshold 50
set routing-instances CUSTOMER-A protocols evpn mac-mobility detection-window 180
```

Module 24: EVPN—Multihoming Features Using Type 1 Routes

Part 1: The Conceptual Lecture (The Why)

Beyond Type 4: The Need for Type 1 Routes

In Module 23, we learned how Type 4 routes enable multihoming through ES discovery and DF election. However, Type 4 routes alone don't solve all multihoming challenges:

1. **Fast Convergence**: When a PE-CE link fails, how quickly can traffic shift?
2. **Load Balancing**: How can remote PEs load-balance to multiple PEs?
3. **Mass Withdrawal**: If a PE loses all its links, how to withdraw all routes quickly?
4. **Aliasing**: How to forward known unicast to all PEs in an ES?

Type 1 routes address these advanced requirements.

Two Flavors of Type 1 Routes

EVPN defines two distinct Type 1 routes, each serving different purposes:

```
Type 1 Route Variants:

1. Per-ES Route (Ethernet Auto-Discovery per Ethernet Segment)
   - One per Ethernet Segment
   - Advertised when ES is configured
   - Used for fast convergence and mass withdrawal

2. Per-EVI Route (Ethernet Auto-Discovery per EVPN Instance)
   - One per ES per EVI
   - Advertised when ES has active VLANs
   - Used for aliasing and load balancing
```

Type 1 Per-ES Route: Fast Convergence

The Per-ES route provides a single advertisement representing all services on an Ethernet Segment:

```
Type 1 Per-ES Route Purpose:
"I (PE-1) am connected to ES X. If this route disappears,
 immediately flush all MACs learned from me on this ES"

Route Structure:
Route Type: 1 (Ethernet Auto-Discovery)
Route Distinguisher: 10.0.0.1:32767
ESI: 00:11:22:33:44:55:66:77:88:99
```

```
Ethernet Tag: MAX-ET (0xFFFFFFFF)    ← Special value
MPLS Label: 0
```

Type 1 Per-ES Route: Fast Convergence (continued)

```
Fast Convergence Example:

Normal Operation:
PE-1 advertises: Type 1 Per-ES for ESI-A
PE-2 advertises: Type 1 Per-ES for ESI-A
PE-3 learns: "Both PE-1 and PE-2 connect to ESI-A"

Link Failure:
1. PE-1 loses link to CE
2. PE-1 withdraws Type 1 Per-ES route
3. PE-3 receives withdrawal
4. PE-3 immediately flushes ALL MACs learned via PE-1 for ESI-A
5. Traffic instantly shifts to PE-2

Without Type 1: Wait for individual MAC timeouts (minutes)
With Type 1: Immediate flush (milliseconds)
```

Type 1 Per-EVI Route: The Aliasing Function

The Per-EVI route enables traffic load balancing to all PEs in an active-active setup:

```
Aliasing Concept:
"If you want to reach MAC X, you can send to ANY PE in the ES"

Without Aliasing:                    With Aliasing:
Remote PE learns MAC from PE-1       Remote PE learns MAC from PE-1
Remote PE sends all traffic to PE-1  Remote PE can send to PE-1 OR PE-2

Result: Link underutilization        Result: Full link utilization
```

How Aliasing Works:

```
Step-by-Step Aliasing:
1. CE (with ESI-A) sends frame with source MAC 00:11:22:33:44:55
2. PE-1 learns MAC and advertises Type 2 route
3. PE-1 also advertises Type 1 Per-EVI: "I have ESI-A in EVI 100"
4. PE-2 also advertises Type 1 Per-EVI: "I have ESI-A in EVI 100"
5. Remote PE-3 builds forwarding state:
   - MAC 00:11:22:33:44:55 → Next-hops: PE-1, PE-2
   - Can load-balance per-flow
```

Split Horizon Labels in Type 1 Routes

Both Type 1 variants carry split horizon labels to prevent loops:

```
Type 1 Route Labels:
Per-ES Route:
  ESI Label: 300000   ← Used for split horizon

Per-EVI Route:
  ESI Label: 300000   ← Same split horizon label
  Aliasing Label: 300100  ← Used for forwarding

Split Horizon in Action:
1. PE-1 receives frame from CE on ESI-A
2. PE-1 floods to core with ESI label 300000
```

```
3. PE-2 receives frame with label 300000
4. PE-2 has same ESI-A with label 300000
5. PE-2 drops frame (split horizon match)
```

Mass Withdrawal Mechanism

Type 1 Per-ES routes enable efficient mass withdrawal:

```
Scenario: PE-1 loses power (all routes need withdrawal)

Traditional Approach:
- Withdraw 1000 Type 2 routes (one per MAC)
- BGP session timeout (90-180 seconds)
- Massive BGP processing load

Type 1 Approach:
- BGP session fails
- Single Type 1 Per-ES withdrawal
- Remote PEs flush all related MACs
- Orders of magnitude faster
```

Part 2: The Junos CLI Masterclass (The How)

Enabling Type 1 Route Advertisement

Type 1 routes are automatically generated when multihoming is configured:

```
# Basic multihoming configuration (automatically generates Type 1)
set interfaces ge-0/0/5 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ge-0/0/5 esi all-active

# Verify Type 1 route generation is enabled (default)
user@PE-1> show configuration routing-instances CUSTOMER-A protocols evpn | display inheritance
##
## '...' inherited from group 'evpn-defaults'
##
    advertise-ethernet-segment-routes;  ## Generates Type 1 Per-ES
    advertise-aliasing-routes;          ## Generates Type 1 Per-EVI
```

Advanced Type 1 Configuration

```
# Configure split horizon label allocation
set routing-instances CUSTOMER-A protocols evpn label-allocation per-instance

# Configure aliasing behavior
set routing-instances CUSTOMER-A protocols evpn aliasing

# Mass withdrawal timer (how long to wait before flushing)
set routing-instances CUSTOMER-A protocols evpn mass-withdrawal-timer 5

# Configure load balancing for aliased routes
set routing-options forwarding-table export LOAD-BALANCE-ALIASED

set policy-options policy-statement LOAD-BALANCE-ALIASED term 1 from protocol evpn
set policy-options policy-statement LOAD-BALANCE-ALIASED term 1 from route-type ethernet-auto-discovery
set policy-options policy-statement LOAD-BALANCE-ALIASED term 1 then load-balance per-packet
```

Comprehensive Multihoming Configuration

Here's a complete configuration showcasing Type 1 features:

```
# PE-1 Complete Configuration for Advanced Multihoming

# Interface configuration with ES
interfaces {
    ge-0/0/5 {
        description "To CE-A (Multihomed with PE-2)";
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        esi {
            00:11:22:33:44:55:66:77:88:99;
            all-active;
        }
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members [ 100 200 ];
                }
            }
        }
    }
}

# EVPN Instance with advanced features
routing-instances {
    CUSTOMER-A {
        instance-type evpn;
        service-type vlan-aware;
        interface ge-0/0/5.0;
        route-distinguisher 10.0.0.1:1000;
        vrf-target target:65000:1000;
        protocols {
            evpn {
                ## Type 1 related configurations
                advertise-ethernet-segment-routes;    ## Default: on
                advertise-aliasing-routes;            ## Default: on
                label-allocation per-instance;        ## Optimizes label usage

                ## Fast convergence features
                fast-reroute;
                mass-withdrawal-timer 2;              ## Faster than default

                ## DF election enhancement
                designated-forwarder-election {
                    algorithm preference-based;
                }
            }
        }
        vlans {
            VLAN100 {
                vlan-id 100;
            }
            VLAN200 {
                vlan-id 200;
            }
        }
    }
}
```

Controlling Type 1 Advertisement

Sometimes you may want fine-grained control:

```
# Disable Type 1 Per-ES routes (not recommended)
set routing-instances CUSTOMER-A protocols evpn no-advertise-ethernet-segment-routes

# Disable Type 1 Per-EVI routes (disables aliasing)
set routing-instances CUSTOMER-A protocols evpn no-advertise-aliasing-routes

# Advertise Type 1 routes even for single-active mode
set interfaces ge-0/0/5 esi single-active
set interfaces ge-0/0/5 esi advertise-evi-routes
```

Policy Control for Type 1 Routes

```
# Create policy to tag Type 1 routes
set policy-options policy-statement TAG-TYPE1 term match-per-es from route-type ethernet-auto-discovery
set policy-options policy-statement TAG-TYPE1 term match-per-es from ethernet-tag 4294967295  ## MAX-ET
set policy-options policy-statement TAG-TYPE1 term match-per-es then community add TYPE1-PER-ES

set policy-options policy-statement TAG-TYPE1 term match-per-evi from route-type ethernet-auto-discovery
set policy-options policy-statement TAG-TYPE1 term match-per-evi from ethernet-tag 100-200
set policy-options policy-statement TAG-TYPE1 term match-per-evi then community add TYPE1-PER-EVI

set policy-options community TYPE1-PER-ES members 65000:11111
set policy-options community TYPE1-PER-EVI members 65000:11112

# Apply to BGP export
set protocols bgp group EVPN-CORE export TAG-TYPE1
```

Part 3: Verification & Troubleshooting (The What-If)

Verifying Type 1 Route Operation

1. **Check Type 1 Per-ES Route Advertisement:**

```
user@PE-1> show route advertising-protocol bgp 10.0.0.100 match-prefix "1:*" detail ethernet-tag 4294967295

bgp.evpn.0: 30 destinations, 30 routes (30 active, 0 holddown, 0 hidden)
* 1:10.0.0.1:32767::00112233445566778899::4294967295/304 (1 entry, 1 announced)
  BGP group EVPN-CORE type Internal
    Route Distinguisher: 10.0.0.1:32767
    Route Label: 0
    ESI: 00:11:22:33:44:55:66:77:88:99
    Ethernet Tag: 4294967295              ## MAX-ET indicates Per-ES
    ESI Label: 300000                      ## Split horizon label
    Nexthop: Self
    Communities: target:65000:1000 target:es:00:11:22:33:44:55:66:77:88:99
```

2. **Check Type 1 Per-EVI Route Advertisement:**

```
user@PE-1> show route advertising-protocol bgp 10.0.0.100 match-prefix "1:*" detail ethernet-tag 100

bgp.evpn.0: 30 destinations, 30 routes (30 active, 0 holddown, 0 hidden)
* 1:10.0.0.1:1000::00112233445566778899::100/304 (1 entry, 1 announced)
  BGP group EVPN-CORE type Internal
    Route Distinguisher: 10.0.0.1:1000     ## RD from EVI
    Route Label: 299776                    ## Aliasing label
    ESI: 00:11:22:33:44:55:66:77:88:99
    Ethernet Tag: 100                      ## VLAN ID
    ESI Label: 300000                      ## Split horizon label
    Nexthop: Self
    Communities: target:65000:1000
```

3. **Verify Aliasing Function:**

```
user@PE-3> show evpn mac-table instance CUSTOMER-A mac-address 00:11:22:33:44:55 extensive

MAC address: 00:11:22:33:44:55
  Ethernet segment: 00:11:22:33:44:55:66:77:88:99
  Active source: 10.0.0.1                ## Learned from PE-1
  Aliased to: 10.0.0.2                   ## But can forward to PE-2!
  Aliasing label: 299800

user@PE-3> show route forwarding-table destination 00:11:22:33:44:55/48 extensive
Route: 00:11:22:33:44:55/48
  Next hop type: composite                ## Multiple next-hops
  Next hop: via indirect 1048574
    Load balance label: 299776            ## To PE-1
    Load balance label: 299800            ## To PE-2 (aliased)
```

4. **Check Mass Withdrawal Timer:**

```
user@PE-1> show evpn instance CUSTOMER-A extensive | match withdrawal
  Mass withdrawal timer: 2 seconds
  Mass withdrawal routes advertised: 2
    Type 1 Per-ES routes: 1
    Type 1 Per-EVI routes: 1
```

Common Troubleshooting Scenarios

Scenario 1: Aliasing Not Working

Symptom: Remote PEs send all traffic to one PE despite multihoming

Diagnostic Commands:

```
user@PE-3> show route table CUSTOMER-A.evpn.0 match-prefix "1:*"
# Missing Type 1 Per-EVI routes!

user@PE-1> show route advertising-protocol bgp 10.0.0.100 match-prefix "1:*" | count
Count: 1 lines    ## Only Per-ES, no Per-EVI
```

Cause: Type 1 Per-EVI routes not being advertised

Solution:

```
# Check if aliasing is disabled:
show configuration routing-instances CUSTOMER-A protocols evpn | match aliasing
    no-advertise-aliasing-routes;    ## Problem found!

# Fix:
delete routing-instances CUSTOMER-A protocols evpn no-advertise-aliasing-routes
commit
```

Scenario 2: MACs Not Flushed on Link Failure

Symptom: After PE-CE link failure, remote PEs continue sending traffic to failed PE

Diagnostic Commands:

```
## Simulate link failure on PE-1
user@PE-1> request interface ge-0/0/5 disable

## Check on remote PE-3
user@PE-3> show evpn mac-table instance CUSTOMER-A | match "10.0.0.1"
00:11:22:33:44:55  10.0.0.1      ## Still pointing to failed PE!
```

```
user@PE-3> show route receive-protocol bgp 10.0.0.1 match-prefix "1:*:*:*:4294967295"
# Type 1 Per-ES route still present
```

Cause: BGP session still up, Type 1 route not withdrawn

Solution:

```
# Configure fast failure detection:
set interfaces ge-0/0/5 link-fault-management
set protocols bgp group EVPN-CORE neighbor 10.0.0.3 bfd-liveness-detection minimum-interval 300

# Or use EVPN OAM:
set routing-instances CUSTOMER-A protocols evpn oam
set routing-instances CUSTOMER-A protocols evpn oam ethernet-segment 00:11:22:33:44:55:66:77:88:99 cfm
```

Scenario 3: Split Horizon Not Working

Symptom: Broadcast storms in multihomed setup

Diagnostic Commands:

```
user@PE-2> monitor traffic interface ge-0/0/7.0 extensive
## Shows broadcast received from CE AND from PE-1!

user@PE-2> show route table mpls.0 label 300000 detail
300000  *[EVPN/7] 00:05:00
           Split Horizon Forwarding table
           ## Should show split horizon action

user@PE-2> show evpn instance CUSTOMER-A split-horizon-label
ESI: 00:11:22:33:44:55:66:77:88:99
  Split horizon label: None     ## Problem!
```

Cause: Split horizon label not allocated

Solution:

```
# Force label reallocation:
clear evpn mac-table instance CUSTOMER-A

# Check ESI configuration:
show configuration interfaces ge-0/0/7 | display inheritance | match esi
    esi {
        00:11:22:33:44:55:66:77:88:99;
        all-active;
        ## df-election-disable;    ## This would break split horizon!
    }

# If found, remove:
delete interfaces ge-0/0/7 esi df-election-disable
commit
```

Scenario 4: Load Balancing Not Optimal

Symptom: All flows going to same PE despite aliasing

Diagnostic Commands:

```
user@PE-3> show route forwarding-table destination 00:11:22:33:44:55/48
Destination        Type RtRef Next hop           Type Index    NhRef Netif
00:11:22:33:44:55/48 user     0                 ulst  1048574     2
                              10.1.1.2           Push 299776   573     1 ge-0/0/0.0
```

```
                            10.1.1.6           Push 299800   574     1 ge-0/0/1.0

user@PE-3> show interfaces statistics interface ge-0/0/0 | match "Output packets"
  Output packets:                           1000000
user@PE-3> show interfaces statistics interface ge-0/0/1 | match "Output packets"
  Output packets:                           1000000    ## Equal distribution expected

## But customer reports all traffic on one link!
```

Cause: Load balancing hash not optimal for traffic pattern

Solution:

```
# Configure enhanced load balancing:
set forwarding-options enhanced-hash-key family evpn function symmetric-hash
set forwarding-options enhanced-hash-key family evpn esi-lag

# Use per-flow instead of per-packet:
set policy-options policy-statement LB-EVPN from protocol evpn
set policy-options policy-statement LB-EVPN then load-balance per-flow
set routing-options forwarding-table export LB-EVPN

# Include Layer 4 in hash:
set forwarding-options enhanced-hash-key family evpn layer-4
commit
```

This completes our comprehensive coverage of EVPN Type 1 routes and their role in enabling advanced multihoming features. Type 1 routes, combined with Type 4 routes from the previous module, provide a complete solution for resilient, efficient, and scalable Layer 2 services across MPLS networks.

Module 25: EVPN—MAC Mobility and IRB Interfaces

Part 1: The Conceptual Lecture (The Why)

The Problem MAC Mobility Solves

Imagine you work in a large corporate campus with thousands of employees carrying laptops. These employees move between buildings, floors, and conference rooms throughout the day. Each time they move, their laptop needs to maintain network connectivity. In traditional Layer 2 networks, this creates a significant challenge.

The Core Problem: When a device (identified by its MAC address) moves from one switch to another in a network, the entire network needs to learn about this move quickly. Without proper mechanisms, you might experience:

- Traffic being sent to the old location (black-holing)
- Duplicate MAC addresses appearing in different locations
- Network loops as switches try to figure out where the device really is
- Slow convergence as the network slowly learns the new location

Understanding MAC Mobility in EVPN

EVPN (Ethernet VPN) introduces an elegant solution called **MAC Mobility** that tracks when devices move between different PE (Provider Edge) routers in the network. Think of it like a postal forwarding service that instantly updates when you move houses.

How MAC Mobility Works

When a MAC address moves:

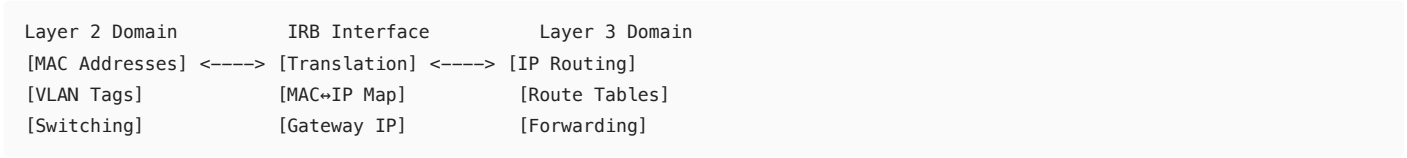
1. **Detection**: PE2 learns MAC address A locally
2. **Sequence Number**: PE2 advertises the MAC with a sequence number (higher than PE1's)
3. **BGP Advertisement**: PE2 sends a BGP EVPN Type-2 route with the MAC mobility extended community
4. **Network Update**: All PEs update their forwarding tables to point to PE2
5. **Old Location Cleanup**: PE1 withdraws its advertisement or marks it as inactive

IRB Interfaces: Bridging Layer 2 and Layer 3

IRB (Integrated Routing and Bridging) interfaces solve another fundamental problem: How do you connect Layer 2 EVPN domains to Layer 3 routing domains?

Think of an IRB interface as a bilingual translator sitting between two groups of people who speak different languages:

- One group speaks "Layer 2" (MAC addresses, VLANs)
- Another group speaks "Layer 3" (IP addresses, routing)



Gateway MAC-IP Synchronization

In EVPN deployments with multiple PE routers acting as gateways, you face a challenge: How do all gateways present the same virtual gateway to hosts?

The Problem:

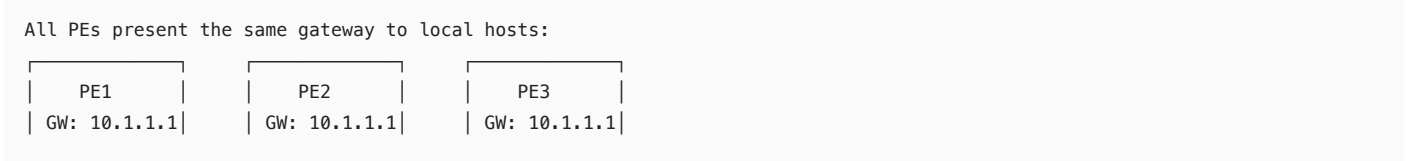
- Host A connected to PE1 thinks the gateway is at MAC address X
- Host B connected to PE2 needs to see the same gateway at the same MAC address X
- But PE1 and PE2 are different physical devices!

Two Solutions:

1. **Automatic Gateway MAC-IP Synchronization**
   - PEs automatically share their IRB interface MAC-IP bindings
   - Uses BGP EVPN Type-2 routes with both MAC and IP information
   - Each PE learns all other PEs' gateway information
2. **Manual Gateway MAC-IP Synchronization**
   - Administrator configures the same virtual MAC on all PEs
   - Simpler but requires careful planning
   - Used when you want explicit control

Virtual Gateway Addresses

Virtual Gateway Addresses provide a single, consistent gateway IP and MAC across all PEs in an EVPN instance. It's like having one phone number that rings at multiple locations - callers don't need to know which specific location will answer.



```
| MAC: AA:BB  |     | MAC: AA:BB  |     | MAC: AA:BB  |
└──────┬──────┘     └──────┬──────┘     └──────┬──────┘
       |                   |                   |
   Host A              Host B              Host C
   Sees same GW       Sees same GW       Sees same GW
```

Part 2: The Junos CLI Masterclass (The How)

Configuration Hierarchy Overview

In Junos, EVPN MAC mobility and IRB configurations live in several interconnected places:

```
[edit]
├── interfaces {           # Physical and IRB interfaces
├── routing-instances {    # EVPN instances and IRB integration
├── protocols {           # BGP for EVPN signaling
└── switch-options {      # EVPN-specific parameters
```

Step-by-Step Configuration

1. Configure IRB Interface

First, create the IRB interface that will serve as the gateway:

```
[edit interfaces]
set irb unit 100 family inet address 10.1.1.1/24
set irb unit 100 mac 00:11:22:33:44:55
```

Why: The IRB interface needs both an IP address (for routing) and a MAC address (for bridging). Unit 100 typically corresponds to VLAN 100.

2. Configure EVPN Instance with MAC Mobility

```
[edit routing-instances EVPN-1]
set instance-type evpn
set vlan-id 100
set interface ge-0/0/0.100
set routing-interface irb.100
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list 5100
set protocols evpn vni-options vni 5100 vrf-target target:65000:100
```

Why: This creates the EVPN instance and ties it to both the Layer 2 domain (VLAN 100) and Layer 3 domain (IRB interface).

3. Enable MAC Mobility

```
[edit routing-instances EVPN-1 protocols evpn]
set mac-mobility
set mac-mobility sticky-mac
```

Why:

- `mac-mobility` enables the feature
- `sticky-mac` prevents certain MACs from moving (useful for servers)

4. Configure Automatic Gateway MAC-IP Synchronization

```
[edit routing-instances EVPN-1 protocols evpn]
set default-gateway auto-gw-mac-advertise
```

Why: This tells the PE to automatically advertise its IRB interface MAC-IP binding to other PEs.

5. Configure Manual Gateway MAC-IP Synchronization (Alternative)

```
[edit interfaces irb unit 100]
set virtual-gateway-address 10.1.1.254
set virtual-gateway-v4-mac 00:00:5e:00:01:01
```

Why: All PEs will use this same virtual MAC/IP combination, ensuring consistency.

6. Configure Virtual Gateway Addresses

```
[edit routing-instances EVPN-1 protocols evpn]
set default-gateway advertise
set default-gateway virtual-gateway-address 10.1.1.254
set default-gateway virtual-gateway-v4-mac 00:00:5e:00:01:01
```

Why: This configures a virtual gateway that's consistent across all PEs.

Complete Reference Configuration

```
## Physical Interface
set interfaces ge-0/0/0 flexible-vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 100 vlan-id 100
set interfaces ge-0/0/0 unit 100 family ethernet-switching

## IRB Interface
set interfaces irb unit 100 family inet address 10.1.1.1/24
set interfaces irb unit 100 mac 00:11:22:33:44:55
set interfaces irb unit 100 virtual-gateway-address 10.1.1.254
set interfaces irb unit 100 virtual-gateway-v4-mac 00:00:5e:00:01:01

## Loopback for VTEP
set interfaces lo0 unit 0 family inet address 192.168.1.1/32

## EVPN Routing Instance
set routing-instances EVPN-1 instance-type evpn
set routing-instances EVPN-1 vlan-id 100
set routing-instances EVPN-1 interface ge-0/0/0.100
set routing-instances EVPN-1 routing-interface irb.100
set routing-instances EVPN-1 vrf-target target:65000:100
set routing-instances EVPN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-1 protocols evpn extended-vni-list 5100
set routing-instances EVPN-1 protocols evpn vni-options vni 5100 vrf-target target:65000:100
set routing-instances EVPN-1 protocols evpn mac-mobility
set routing-instances EVPN-1 protocols evpn default-gateway advertise
set routing-instances EVPN-1 protocols evpn default-gateway virtual-gateway-address 10.1.1.254
set routing-instances EVPN-1 protocols evpn default-gateway virtual-gateway-v4-mac 00:00:5e:00:01:01

## BGP Configuration for EVPN
set protocols bgp group EVPN-PEERS type internal
set protocols bgp group EVPN-PEERS local-address 192.168.1.1
set protocols bgp group EVPN-PEERS family evpn signaling
set protocols bgp group EVPN-PEERS neighbor 192.168.1.2
set protocols bgp group EVPN-PEERS neighbor 192.168.1.3
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify MAC Mobility Status

```
user@PE1> show evpn mac-table instance EVPN-1 extensive
MAC address: 00:50:56:84:3b:6c
  Ethernet segment: 00:00:00:00:00:00:00:00:00:00
  VLAN ID: 100
  Source: ge-0/0/0.100
  Active source: ge-0/0/0.100
  State: Active
  Mobility sequence number: 3        <-- Indicates MAC has moved 3 times
  Local MAC advertisement: Type 2
```

```
  Remote MAC advertisement: Not advertised
  History db:
    Time                 Event
    Nov 10 10:15:23 2023  Learned from ge-0/0/0.100
    Nov 10 10:20:45 2023  Moved to remote PE 192.168.1.2
    Nov 10 10:25:12 2023  Moved back to local
```

2. Verify IRB Interface and Gateway

```
user@PE1> show interfaces irb.100 extensive
Logical interface irb.100 (Index 334) (SNMP ifIndex 540)
  Flags: Up SNMP-Traps 0x4004000 VLAN-Tag [ 0x8100.100 ]
  Ethernet-switching parameters:
    Virtual-gateway-address: 10.1.1.254
    Virtual-gateway-v4-mac: 00:00:5e:00:01:01
  Local statistics:
    Input packets : 15234
    Output packets: 18976
```

3. Verify EVPN Database

```
user@PE1> show evpn database instance EVPN-1 mac-ip-table
MAC address        IP address         Logical interface  Active source
00:50:56:84:3b:6c  10.1.1.100        ge-0/0/0.100       00:50:56:84:3b:6c
00:00:5e:00:01:01  10.1.1.254        irb.100            00:00:5e:00:01:01
```

Common Troubleshooting Scenarios

Scenario 1: MAC Not Moving Properly

Symptom: When a host moves from PE1 to PE2, traffic continues going to PE1

Diagnostic Commands:

```
user@PE2> show evpn mac-table instance EVPN-1
## No entry for the MAC that should have moved

user@PE2> show log messages | match "EVPN|MAC"
Nov 10 10:30:12 PE2 EVPN: MAC mobility detected but sequence number not incremented
```

Cause: MAC mobility sequence number not incrementing properly

Solution:

```
[edit routing-instances EVPN-1 protocols evpn]
set mac-mobility window-size 10
set mac-mobility detection-threshold 5
commit
```

Scenario 2: Virtual Gateway Not Working

Symptom: Hosts cannot reach their default gateway

Diagnostic Commands:

```
user@PE1> show arp no-resolve | match 10.1.1.254
## No ARP entry for virtual gateway

user@PE1> show evpn instance EVPN-1 extensive | match "gateway"
Default gateway: Not advertised
```

Cause: Virtual gateway not properly advertised

Solution:

```
[edit routing-instances EVPN-1 protocols evpn]
set default-gateway advertise
commit

## Verify after commit:
user@PE1> show route table EVPN-1.evpn.0 match-prefix *10.1.1.254*
```

Scenario 3: IRB Interface Not Learning MAC-IP Bindings

Symptom: Hosts in EVPN cannot communicate with external networks

Diagnostic Commands:

```
user@PE1> show ethernet-switching table instance EVPN-1
## Shows MAC addresses but no IP correlation

user@PE1> show route forwarding-table family ethernet-switching
## Missing IRB interface entries
```

Cause: IRB interface not properly associated with EVPN instance

Solution:

```
[edit routing-instances EVPN-1]
set routing-interface irb.100
commit

## Also ensure VLAN ID matches:
[edit routing-instances EVPN-1]
set vlan-id 100
```

Scenario 4: Duplicate MAC Detection

Symptom: MAC address appears to be flapping between PEs rapidly

Diagnostic Commands:

```
user@PE1> show evpn mac-table duplicate-macs
MAC address: 00:50:56:84:3b:6c
  Detection time: Nov 10 10:35:00 2023
  Number of moves: 15
  Move threshold: 5
  Detection window: 180 seconds
  Action: Block
```

Cause: Loop in the network or misconfigured host

Solution:

```
## Temporarily increase threshold while investigating:
[edit routing-instances EVPN-1 protocols evpn]
set duplicate-mac-detection detection-threshold 20
set duplicate-mac-detection detection-window 300
commit

## Find and fix the root cause, then restore normal values
```

Best Practices for MAC Mobility and IRB

1. **Always configure consistent virtual gateway addresses** across all PEs
2. **Monitor MAC mobility counters** - excessive moves indicate network problems
3. **Use sticky MAC** for devices that should never move (servers, printers)
4. **Set appropriate duplicate MAC detection thresholds** based on your environment
5. **Ensure IRB interface MAC addresses are unique** unless using virtual gateway

Module 26: EVPN—Integration with L3VPNs

Part 1: The Conceptual Lecture (The Why)

Understanding the Need for EVPN-L3VPN Integration

Imagine a large enterprise with multiple data centers and branch offices. Some locations need Layer 2 connectivity (servers in the same VLAN across sites), while others need Layer 3 routing (efficient WAN connectivity). Traditionally, you'd deploy separate networks:

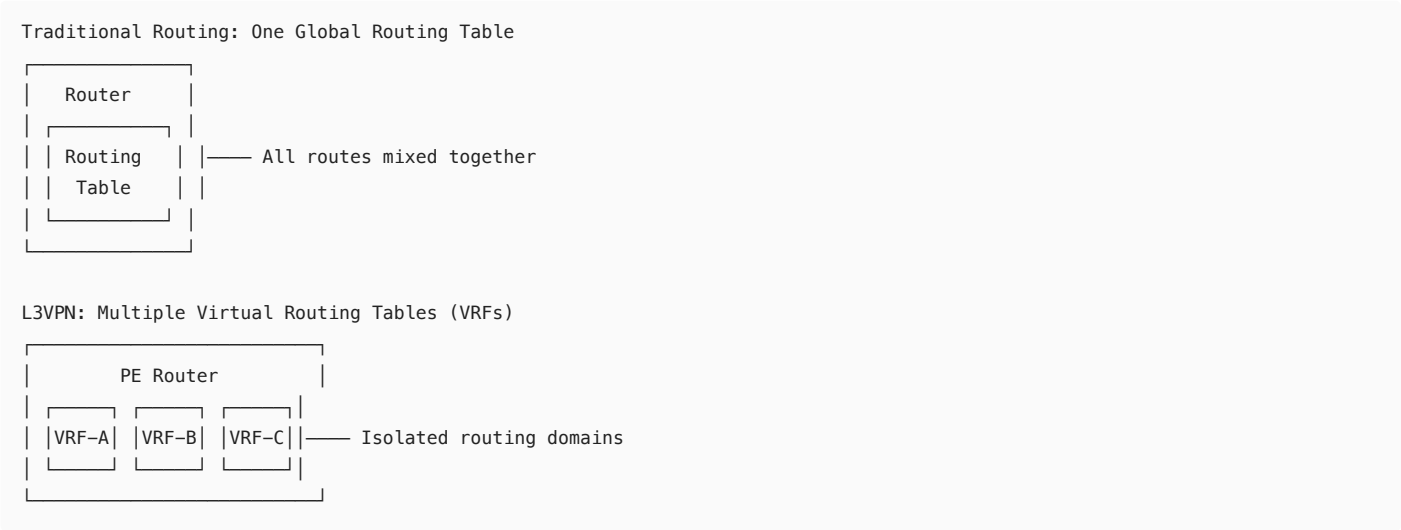
- EVPN for Layer 2 extension
- L3VPN for Layer 3 routing

But what happens when a server in an EVPN domain needs to communicate with a branch office connected via L3VPN? Without integration, traffic takes inefficient paths, requires multiple devices, and becomes complex to manage.

The Basic Functionality of L3VPN

Before diving into integration, let's understand L3VPN fundamentals.

L3VPN (Layer 3 Virtual Private Network) creates isolated routing domains across a shared MPLS infrastructure. Think of it as creating multiple virtual routers within physical routers.



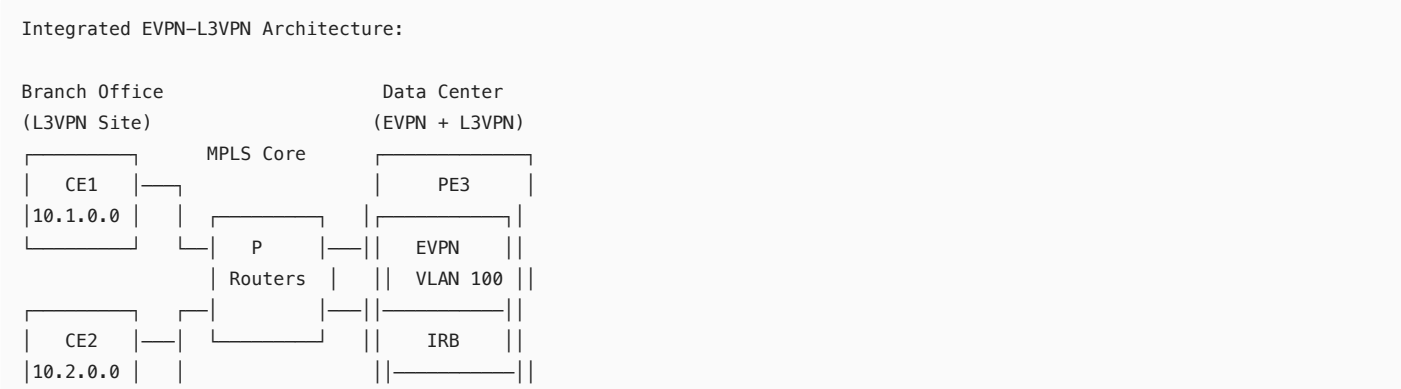
Key L3VPN Concepts:

- **VRF (Virtual Routing and Forwarding)**: Separate routing table for each customer
- **Route Distinguisher (RD)**: Makes routes globally unique (like area codes for phone numbers)
- **Route Target (RT)**: Controls which routes go into which VRFs (like mailing lists)
- **MP-BGP**: Carries VPN routes with labels across the MPLS core

How EVPNs and L3VPNs Integrate

EVPN can seamlessly integrate with L3VPN through IRB interfaces. This creates a unified architecture where:

- Layer 2 traffic stays within EVPN
- Layer 3 traffic leverages L3VPN routing
- Optimal forwarding paths for both types of traffic



```
└──────────┘   |                ||  L3VPN    ||
               |                ||   VRF     ||
              PE1               |└──────────┘|
                                └────────────┘
```

Integration Benefits:

1. **Single Service Model**: One PE router handles both L2 and L3 services
2. **Optimal Routing**: Traffic takes the best path based on destination
3. **Seamless Mobility**: Hosts can move between L2 domains while maintaining L3 connectivity
4. **Simplified Operations**: Unified configuration and troubleshooting

Chained Composite Next Hop

The Packet Forwarding Engine (PFE) in Junos uses an optimization called **Chained Composite Next Hop** for EVPN-L3VPN integration.

The Problem: Without optimization, each MAC/IP route would need separate forwarding entries:

- One entry for Layer 2 forwarding (MAC destination)
- Another entry for Layer 3 forwarding (IP destination)
- Massive forwarding table growth in large deployments

The Solution: Chained Composite Next Hop creates a single forwarding structure that handles both L2 and L3 forwarding decisions:

```
Traditional Forwarding (Inefficient):
MAC Table: 00:11:22:33:44:55 → Interface ge-0/0/1.100
IP Table:  10.1.1.100/32 → Next-hop 192.168.1.1

Chained Composite Next Hop (Efficient):
Composite Entry:
├── L2 Component: MAC 00:11:22:33:44:55
├── L3 Component: IP 10.1.1.100/32
└── Forwarding: Single lookup → Interface ge-0/0/1.100 + MPLS labels
```

Benefits:

- Reduces forwarding entries by up to 50%
- Faster packet processing (single lookup vs. multiple)
- Lower memory usage in PFE
- Scales to larger deployments

Part 2: The Junos CLI Masterclass (The How)

Configuration Hierarchy for EVPN-L3VPN Integration

```
[edit]
├── interfaces {          # IRB interfaces for EVPN-L3VPN binding
├── routing-instances {   # Both EVPN and L3VPN instances
│   ├── EVPN-1 {         # EVPN instance
│   └── L3VPN-1 {        # L3VPN VRF instance
├── routing-options {     # Route distinguishers and autonomous-system
└── protocols {          # BGP for both EVPN and L3VPN signaling
```

Step-by-Step Configuration

1. Configure the IRB Interface

```
[edit interfaces]
set irb unit 100 family inet address 10.100.1.1/24
set irb unit 100 mac 00:11:22:33:44:01
```

Why: The IRB interface serves as the bridge between EVPN (L2) and L3VPN (L3).

2. Configure the EVPN Instance

```
[edit routing-instances EVPN-1]
set instance-type evpn
set vlan-id 100
set interface ge-0/0/0.100
set routing-interface irb.100
set vrf-target target:65000:100
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list 5100
set protocols evpn vni-options vni 5100 vrf-target target:65000:100
```

Why: Standard EVPN configuration with IRB interface for L3 gateway functionality.

3. Configure the L3VPN Instance

```
[edit routing-instances L3VPN-1]
set instance-type vrf
set interface irb.100
set route-distinguisher 192.168.1.1:100
set vrf-target target:65000:1000
set vrf-table-label
```

Why:

- `instance-type vrf` creates the L3VPN instance
- `interface irb.100` binds the same IRB to the VRF
- `vrf-table-label` enables MPLS label allocation for the VRF

4. Enable EVPN-L3VPN Route Leaking

```
[edit routing-instances EVPN-1]
set protocols evpn ip-prefix-routes advertise direct-nexthop
set protocols evpn ip-prefix-routes encapsulation vxlan
set protocols evpn ip-prefix-routes vni 5100
set protocols evpn ip-prefix-routes export l3vpn-export-policy
```

Why: This enables EVPN to advertise IP prefixes (Type-5 routes) for L3VPN integration.

5. Configure Route Leaking Policies

```
[edit policy-options]
set policy-statement l3vpn-export-policy term from-evpn from protocol evpn
set policy-statement l3vpn-export-policy term from-evpn then accept

set policy-statement evpn-import-policy term from-l3vpn from protocol bgp
set policy-statement evpn-import-policy term from-l3vpn from community target:65000:1000
set policy-statement evpn-import-policy term from-l3vpn then accept
```

Why: Controls which routes move between EVPN and L3VPN domains.

6. Configure BGP for Both Address Families

```
[edit protocols bgp group MPLS-CORE]
set type internal
set local-address 192.168.1.1
set family inet-vpn unicast
set family evpn signaling
set neighbor 192.168.1.2
set neighbor 192.168.1.3
```

Why: BGP needs to signal both L3VPN routes (inet-vpn) and EVPN routes.

7. Enable Chained Composite Next Hop

```
[edit routing-options]
set forwarding-table chained-composite-next-hop ingress evpn
```

Why: Optimizes PFE forwarding for integrated EVPN-L3VPN deployments.

Complete Reference Configuration

```
## Interfaces
set interfaces ge-0/0/0 flexible-vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 100 vlan-id 100
set interfaces ge-0/0/0 unit 100 family ethernet-switching

set interfaces irb unit 100 family inet address 10.100.1.1/24
set interfaces irb unit 100 mac 00:11:22:33:44:01

set interfaces lo0 unit 0 family inet address 192.168.1.1/32

## EVPN Instance
set routing-instances EVPN-1 instance-type evpn
set routing-instances EVPN-1 vlan-id 100
set routing-instances EVPN-1 interface ge-0/0/0.100
set routing-instances EVPN-1 routing-interface irb.100
set routing-instances EVPN-1 vrf-target target:65000:100
set routing-instances EVPN-1 protocols evpn encapsulation vxlan
set routing-instances EVPN-1 protocols evpn extended-vni-list 5100
set routing-instances EVPN-1 protocols evpn vni-options vni 5100 vrf-target target:65000:100
set routing-instances EVPN-1 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances EVPN-1 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances EVPN-1 protocols evpn ip-prefix-routes vni 5100

## L3VPN Instance
set routing-instances L3VPN-1 instance-type vrf
set routing-instances L3VPN-1 interface irb.100
set routing-instances L3VPN-1 interface ge-0/0/1.0
set routing-instances L3VPN-1 route-distinguisher 192.168.1.1:100
set routing-instances L3VPN-1 vrf-target target:65000:1000
set routing-instances L3VPN-1 vrf-table-label

## Routing Options
set routing-options router-id 192.168.1.1
set routing-options autonomous-system 65000
set routing-options forwarding-table chained-composite-next-hop ingress evpn

## BGP Configuration
set protocols bgp group MPLS-CORE type internal
set protocols bgp group MPLS-CORE local-address 192.168.1.1
set protocols bgp group MPLS-CORE family inet-vpn unicast
set protocols bgp group MPLS-CORE family evpn signaling
set protocols bgp group MPLS-CORE neighbor 192.168.1.2
set protocols bgp group MPLS-CORE neighbor 192.168.1.3

## Policy Configuration
set policy-options policy-statement l3vpn-export-policy term from-evpn from protocol evpn
set policy-options policy-statement l3vpn-export-policy term from-evpn then accept
set policy-options policy-statement evpn-import-policy term from-l3vpn from protocol bgp
set policy-options policy-statement evpn-import-policy term from-l3vpn from community target:65000:1000
set policy-options policy-statement evpn-import-policy term from-l3vpn then accept

set routing-instances EVPN-1 protocols evpn ip-prefix-routes export l3vpn-export-policy
set routing-instances L3VPN-1 vrf-import evpn-import-policy
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify EVPN-L3VPN Integration Status

```
user@PE1> show evpn l3-context instance EVPN-1
L3 context information for EVPN instance EVPN-1:
  IPv4 Router ID: 192.168.1.1
  IPv6 Router ID: ::
  L3 interfaces:
    irb.100
  L3VPN instances connected:
    L3VPN-1
```

2. Verify Route Exchange Between EVPN and L3VPN

```
user@PE1> show route table L3VPN-1.inet.0 protocol evpn
L3VPN-1.inet.0: 15 destinations, 20 routes (15 active)
+ = Active Route, * = Last Active

10.100.1.0/24      *[EVPN/7] 00:10:23
                    > via irb.100
10.100.1.100/32    *[EVPN/7] 00:05:45
                    > to 10.100.1.100 via irb.100
```

3. Verify Chained Composite Next Hop

```
user@PE1> show route forwarding-table family evpn table EVPN-1 detail
Destination: 00:50:56:84:3b:6c::10.100.1.100/304
  Next-hop type: composite
  Next-hop index: 1048589
  L2 flags: forward-to-kernel
  L3 flags: vrf-lookup L3VPN-1
```

Common Troubleshooting Scenarios

Scenario 1: Routes Not Leaking Between EVPN and L3VPN

Symptom: L3VPN sites cannot reach hosts in EVPN domain

Diagnostic Commands:

```
user@PE1> show route table EVPN-1.evpn.0 route-type 5
## No Type-5 (IP prefix) routes

user@PE1> show evpn ip-prefix-database
## Empty or missing entries
```

Cause: IP prefix routes not configured or advertised

Solution:

```
[edit routing-instances EVPN-1 protocols evpn]
set ip-prefix-routes advertise direct-nexthop
set ip-prefix-routes encapsulation vxlan
set ip-prefix-routes vni 5100
commit
```

Scenario 2: Suboptimal Routing Path

Symptom: Traffic between EVPN and L3VPN sites takes inefficient path

Diagnostic Commands:

```
user@PE1> show route forwarding-table destination 10.200.1.100
## Shows next-hop pointing to wrong PE

user@PE1> show route table L3VPN-1.inet.0 10.200.1.100 detail
## Multiple paths with wrong preference
```

Cause: Chained composite next-hop not enabled

Solution:

```
[edit routing-options]
set forwarding-table chained-composite-next-hop ingress evpn
commit

## Verify optimization:
user@PE1> show pfe route ip table-index 0 | match composite
```

Scenario 3: IRB Interface Not Available in Both Instances

Symptom: Ping from L3VPN to EVPN fails

Diagnostic Commands:

```
user@PE1> show interfaces irb.100 routing-instance all
  Routing Instance: EVPN-1
  ## Missing from L3VPN-1

user@PE1> show route table L3VPN-1.inet.0 | match "10.100.1.1"
## No local route for IRB interface
```

Cause: IRB interface not added to L3VPN instance

Solution:

```
[edit routing-instances L3VPN-1]
set interface irb.100
commit

## Verify:
user@PE1> show interfaces irb.100 routing-instance all
  Routing Instance: EVPN-1, L3VPN-1
```

Scenario 4: BGP Session Not Exchanging Both Address Families

Symptom: Either EVPN or L3VPN routes missing on remote PEs

Diagnostic Commands:

```
user@PE1> show bgp neighbor 192.168.1.2 | match "NLRI"
  NLRI configured: inet-vpn-unicast
  ## Missing evpn

user@PE1> show bgp summary
## Shows session up but 0 received prefixes for evpn
```

Cause: BGP not configured for both address families

Solution:

```
[edit protocols bgp group MPLS-CORE]
set family evpn signaling
set family inet-vpn unicast
commit

## Verify both families active:
user@PE1> show bgp neighbor 192.168.1.2 | match "NLRI|Address families"
```

Best Practices for EVPN-L3VPN Integration

1. **Use consistent route targets** carefully - overlap can cause routing loops
2. **Enable chained composite next-hop** for all EVPN deployments with L3 services
3. **Monitor PFE memory usage** - composite entries use more memory than simple entries
4. **Use Type-5 routes** for efficient IP prefix advertisement between domains
5. **Implement proper filtering** to prevent route leaking between unrelated VPNs

Module 27: Inter-AS MPLS VPNs

Part 1: The Conceptual Lecture (The Why)

The Inter-AS Challenge

Imagine two large service providers want to offer seamless VPN services to a multinational corporation. The customer has offices in countries served by different providers:

- Provider A serves North America (AS 65001)
- Provider B serves Europe (AS 65002)

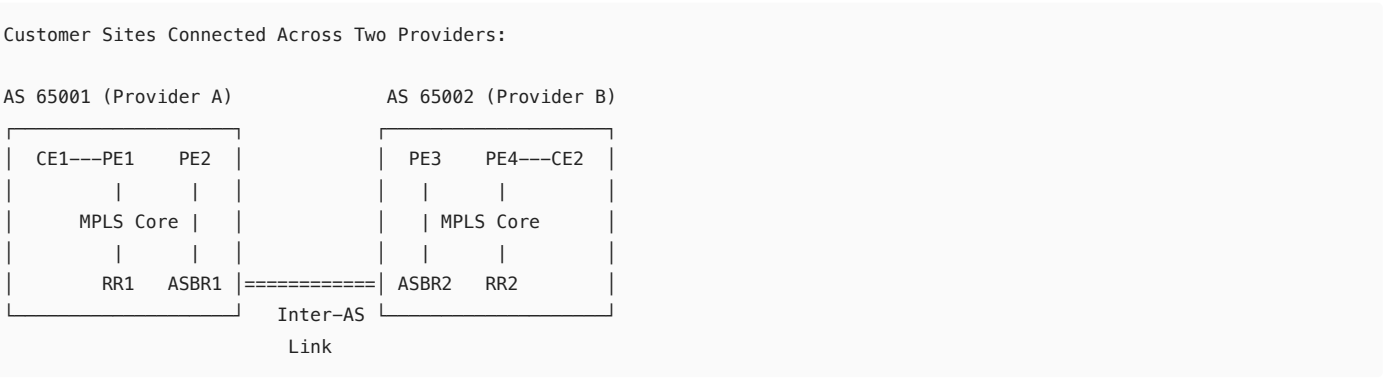
The customer expects their VPN to work seamlessly across both providers. This creates a fundamental challenge: How do you extend MPLS VPNs across autonomous system boundaries while maintaining:

- Security and isolation between customers
- Scalability for thousands of VPNs
- Operational independence between providers

Understanding Autonomous Systems in MPLS Context

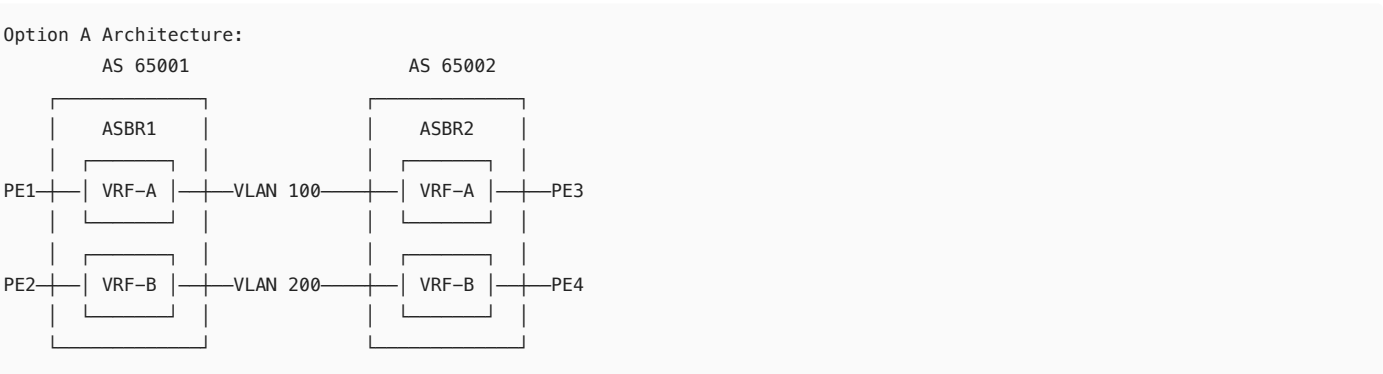
An **Autonomous System (AS)** is a collection of networks under a single administrative control. In MPLS VPNs:

- Each AS has its own IGP (OSPF/IS-IS)
- Each AS has its own MPLS label space
- BGP sessions between ASes are typically eBGP



Interprovider Option A: Back-to-Back VRFs

Option A is the simplest approach: Create VRF connections between ASBRs.



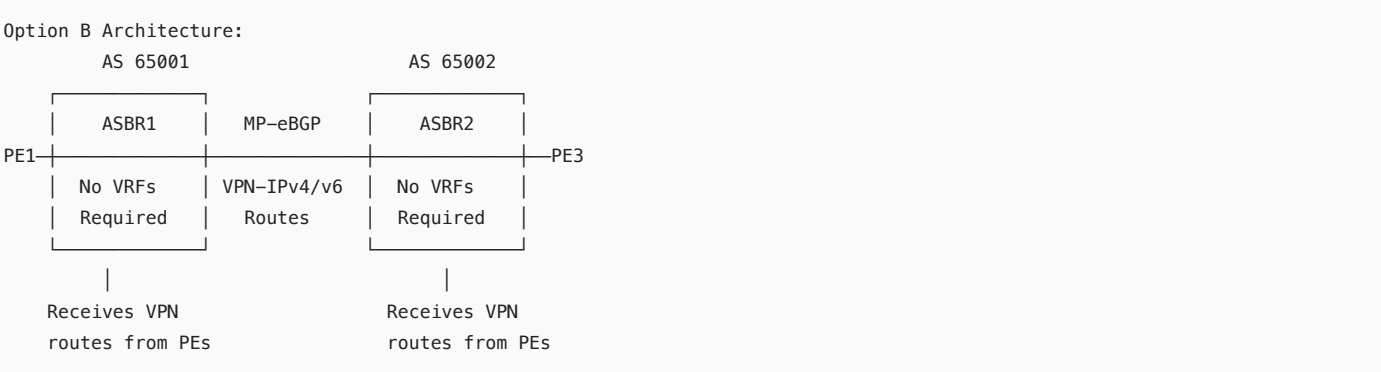
How it works:

1. Each ASBR maintains VRFs for each Inter-AS VPN
2. ASBRs connect VRFs using VLANs or sub-interfaces
3. Each VRF connection is like a PE-CE connection
4. No MPLS labels exchanged between ASBRs

Pros: Simple, secure (complete isolation)
Cons: Not scalable (ASBR needs VRF per VPN), ASBR becomes bottleneck

Interprovider Option B: ASBR-to-ASBR MP-eBGP

Option B uses MP-eBGP to exchange VPN routes directly between ASBRs.



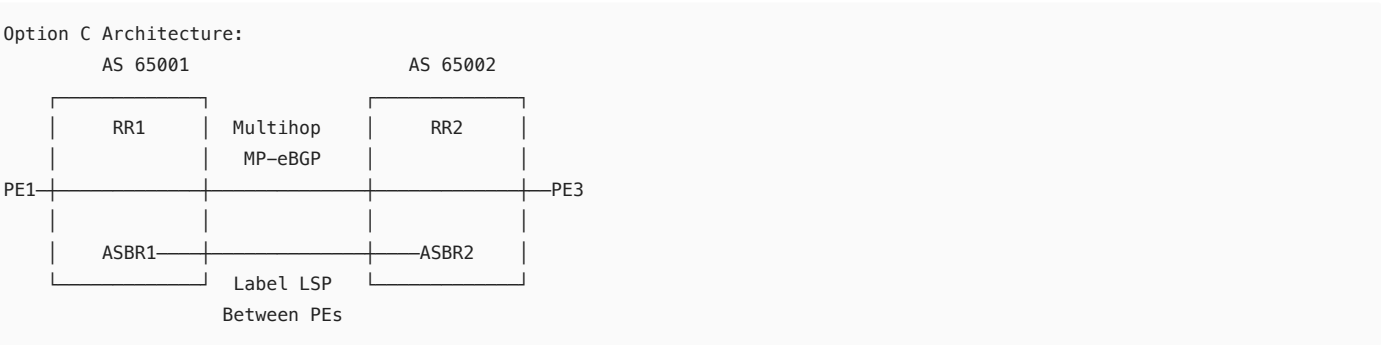
How it works:

1. PEs advertise VPN routes to ASBRs via MP-iBGP
2. ASBRs exchange VPN routes via MP-eBGP
3. ASBRs allocate new labels for VPN routes
4. No VRFs needed on ASBRs

Pros: More scalable than Option A
Cons: ASBRs must process all VPN routes, potential security concerns

Interprovider Option C: Multihop MP-eBGP

Option C establishes MP-eBGP sessions directly between Route Reflectors or PEs across AS boundaries.



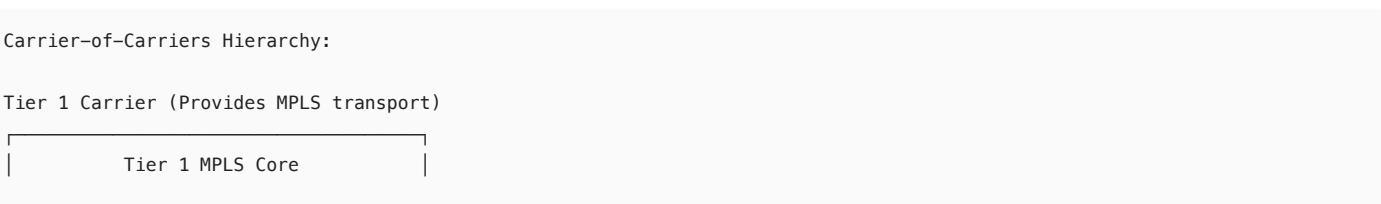
How it works:

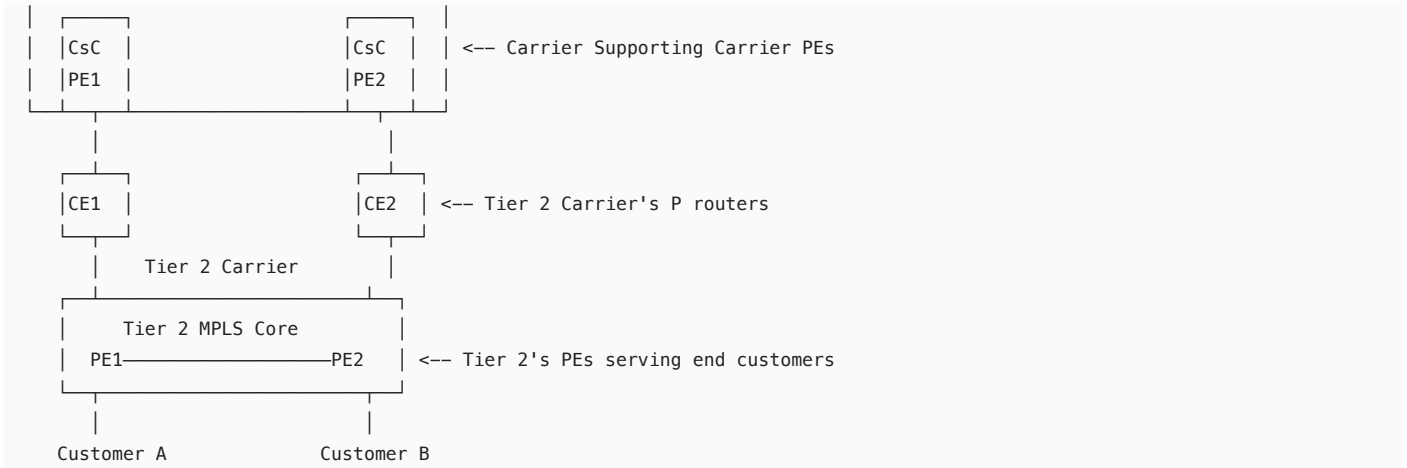
1. RRs exchange VPN routes via multihop MP-eBGP
2. ASBRs exchange PE loopbacks via labeled BGP (BGP-LU)
3. End-to-end LSP established between PEs
4. ASBRs don't process VPN routes

Pros: Most scalable, ASBRs only handle transport
Cons: Most complex, requires coordination between providers

Carrier-of-Carriers VPNs

Carrier-of-Carriers (CoC) allows one service provider to offer VPN services to another service provider.





Key Concepts:

- Tier 1 carrier provides MPLS VPN service to Tier 2
- Tier 2's P routers appear as CEs to Tier 1
- Tier 2 runs its own MPLS and can offer VPNs
- Labels are stacked: Tier 1 label + Tier 2 label

Part 2: The Junos CLI Masterclass (The How)

Configuration Focus: Interprovider Option C

We'll focus on Option C as it's the most complex and scalable solution.

1. Configure ASBR for Option C - BGP Labeled Unicast

```
## On ASBR1 in AS 65001
[edit interfaces ge-0/0/0]
set description "To ASBR2 AS 65002"
set unit 0 family inet address 10.0.0.1/30
set unit 0 family mpls

[edit protocols mpls]
set interface ge-0/0/0.0

[edit protocols bgp group INTER-AS]
set type external
set peer-as 65002
set neighbor 10.0.0.2
set family inet labeled-unicast
set export advertise-pe-loopbacks
```

Why: BGP-LU advertises PE loopback addresses with MPLS labels, creating an LSP between ASes.

2. Configure Policy to Advertise PE Loopbacks

```
[edit policy-options]
set prefix-list pe-loopbacks 192.168.1.0/24

set policy-statement advertise-pe-loopbacks term pe-loopbacks from prefix-list pe-loopbacks
set policy-statement advertise-pe-loopbacks term pe-loopbacks then accept
set policy-statement advertise-pe-loopbacks term default then reject
```

Why: Only PE loopback addresses should be advertised between ASes for Option C.

3. Configure Route Reflector for Multihop MP-eBGP

```
## On RR1 in AS 65001
[edit protocols bgp group INTER-AS-VPN]
set type external
```

```
set multihop ttl 10
set local-address 192.168.1.100
set peer-as 65002
set neighbor 192.168.2.100
set family inet-vpn unicast
set family inet6-vpn unicast
```

Why: Multihop eBGP session between RRs exchanges VPN routes across AS boundaries.

4. Configure Next-Hop Resolution

```
[edit routing-options]
set resolution rib inet.3 resolution-ribs inet.0
```

Why: Allows BGP to resolve next-hops using labeled routes in inet.3 table.

5. Configure PE for Option C

```
## On PE1 in AS 65001
[edit routing-instances VPN-A]
set instance-type vrf
set interface ge-0/0/1.0
set route-distinguisher 192.168.1.1:100
set vrf-target target:1:100
set vrf-table-label

[edit protocols bgp group RR]
set type internal
set local-address 192.168.1.1
set family inet-vpn unicast
set neighbor 192.168.1.100
```

Why: Standard L3VPN configuration with routes sent to local RR.

Complete Option C Reference Configuration

```
## ASBR1 Configuration (AS 65001)
## Physical Interfaces
set interfaces ge-0/0/0 description "To ASBR2 AS 65002"
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces ge-0/0/0 unit 0 family mpls

set interfaces ge-0/0/1 description "To P router"
set interfaces ge-0/0/1 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/1 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.1.254/32

## MPLS
set protocols mpls interface ge-0/0/0.0
set protocols mpls interface ge-0/0/1.0

## RSVP
set protocols rsvp interface ge-0/0/1.0

## BGP Configuration
set protocols bgp group INTER-AS type external
set protocols bgp group INTER-AS peer-as 65002
set protocols bgp group INTER-AS neighbor 10.0.0.2
set protocols bgp group INTER-AS family inet labeled-unicast
set protocols bgp group INTER-AS export advertise-pe-loopbacks

set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 192.168.1.254
set protocols bgp group IBGP family inet labeled-unicast
set protocols bgp group IBGP neighbor 192.168.1.100

## Policy Configuration
set policy-options prefix-list pe-loopbacks 192.168.1.0/24
```

```
set policy-options policy-statement advertise-pe-loopbacks term pe-loopbacks from prefix-list pe-loopbacks
set policy-options policy-statement advertise-pe-loopbacks term pe-loopbacks then accept
set policy-options policy-statement advertise-pe-loopbacks term default then reject

## Routing Options
set routing-options autonomous-system 65001
set routing-options resolution rib inet.3 resolution-ribs inet.0

## RR1 Configuration (AS 65001)
set interfaces lo0 unit 0 family inet address 192.168.1.100/32

set protocols bgp group PE-CLIENTS type internal
set protocols bgp group PE-CLIENTS local-address 192.168.1.100
set protocols bgp group PE-CLIENTS family inet-vpn unicast
set protocols bgp group PE-CLIENTS family inet6-vpn unicast
set protocols bgp group PE-CLIENTS cluster 192.168.1.100
set protocols bgp group PE-CLIENTS neighbor 192.168.1.1
set protocols bgp group PE-CLIENTS neighbor 192.168.1.2

set protocols bgp group INTER-AS-VPN type external
set protocols bgp group INTER-AS-VPN multihop ttl 10
set protocols bgp group INTER-AS-VPN local-address 192.168.1.100
set protocols bgp group INTER-AS-VPN peer-as 65002
set protocols bgp group INTER-AS-VPN neighbor 192.168.2.100
set protocols bgp group INTER-AS-VPN family inet-vpn unicast
set protocols bgp group INTER-AS-VPN family inet6-vpn unicast

## PE1 Configuration (AS 65001)
set interfaces ge-0/0/0 unit 0 family inet address 10.10.1.1/30
set interfaces ge-0/0/0 unit 0 family mpls
set interfaces ge-0/0/1 unit 0 family inet address 172.16.1.1/30
set interfaces lo0 unit 0 family inet address 192.168.1.1/32

set routing-instances VPN-A instance-type vrf
set routing-instances VPN-A interface ge-0/0/1.0
set routing-instances VPN-A route-distinguisher 192.168.1.1:100
set routing-instances VPN-A vrf-target target:1:100
set routing-instances VPN-A vrf-table-label

set protocols bgp group RR type internal
set protocols bgp group RR local-address 192.168.1.1
set protocols bgp group RR family inet-vpn unicast
set protocols bgp group RR neighbor 192.168.1.100

set protocols mpls interface ge-0/0/0.0
set protocols rsvp interface ge-0/0/0.0
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify BGP Labeled Unicast on ASBRs

```
user@ASBR1> show route receive-protocol bgp 10.0.0.2 table inet.3
inet.3: 10 destinations, 10 routes
  Prefix                  Nexthop         MED     Lclpref AS path
  192.168.2.1/32          10.0.0.2                        65002 I
  192.168.2.2/32          10.0.0.2                        65002 I
```

2. Verify End-to-End LSP

```
user@PE1> show route table inet.3 192.168.2.1 detail
inet.3: 15 destinations, 15 routes
192.168.2.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop: 10.1.1.2 via ge-0/0/0.0
                Label operation: Push 299840, Push 299776(top)
                ## Two labels: top for reaching ASBR, bottom for remote PE
```

3. Verify Inter-AS VPN Routes

```
user@RR1> show route receive-protocol bgp 192.168.2.100 table bgp.l3vpn.0
bgp.l3vpn.0: 50 destinations, 50 routes
  Prefix                              Nexthop
  1:100:172.16.2.0/30                192.168.2.1
  1:100:172.16.2.4/30                192.168.2.2
```

Common Troubleshooting Scenarios

Scenario 1: No Labeled Routes Between ASBRs

Symptom: PEs in different ASes cannot reach each other

Diagnostic Commands:

```
user@ASBR1> show route advertising-protocol bgp 10.0.0.2 table inet.3
## No routes advertised

user@ASBR1> show bgp neighbor 10.0.0.2 | match "NLRI"
  NLRI configured: inet-unicast
  ## Should show inet-labeled-unicast
```

Cause: BGP not configured for labeled-unicast family

Solution:

```
[edit protocols bgp group INTER-AS]
set family inet labeled-unicast
delete family inet unicast
commit
```

Scenario 2: VPN Routes Not Exchanged Between RRs

Symptom: VPN routes from AS1 not visible in AS2

Diagnostic Commands:

```
user@RR1> show bgp neighbor 192.168.2.100
## State: Active (not Established)

user@RR1> ping 192.168.2.100 source 192.168.1.100
## No route to host
```

Cause: No IP connectivity between RRs (multihop eBGP requirement)

Solution:

```
## Ensure RR loopbacks are in IGP or static routes
[edit routing-options static]
set route 192.168.2.100/32 next-hop 10.0.0.2

## Or ensure BGP-LU is advertising RR loopbacks
[edit policy-options prefix-list pe-loopbacks]
set 192.168.1.100/32
set 192.168.2.100/32
```

Scenario 3: Carrier-of-Carriers Label Stack Issues

Symptom: Customer traffic dropped at CsC-PE

Diagnostic Commands:

```
user@CsC-PE1> show route table mpls.0 label 300100
## No route
```

```
user@CsC-PE1> show ldp neighbor
## No LDP session with CE
```

Cause: LDP not enabled on PE-CE link for label distribution

Solution:

```
[edit protocols ldp]
set interface ge-0/0/1.0
commit

[edit routing-instances CARRIER-B protocols mpls]
set interface ge-0/0/1.0
```

Scenario 4: Option B Scaling Issues

Symptom: ASBR CPU high, BGP sessions flapping

Diagnostic Commands:

```
user@ASBR1> show bgp summary
## 50,000+ VPN prefixes

user@ASBR1> show system processes extensive | match rpd
## CPU usage > 90%
```

Cause: Option B ASBR processing too many VPN routes

Solution: Migrate to Option C:

```
## Remove VPN families from ASBR eBGP
[edit protocols bgp group INTER-AS]
delete family inet-vpn
delete family inet6-vpn
set family inet labeled-unicast

## Configure RR for multihop eBGP instead
```

Best Practices for Inter-AS MPLS VPNs

1. **Choose the right option**:
   - Option A: Small scale, high security requirements
   - Option B: Medium scale, simpler than Option C
   - Option C: Large scale, complex but most efficient
2. **Security considerations**:
   - Filter routes between ASes using policies
   - Use BGP TTL security (GTSM) for eBGP sessions
   - Implement route validation (RPKI/ROV)
3. **Scaling guidelines**:
   - Option C: ASBRs should only handle transport routes
   - Use route reflectors to reduce full mesh requirements
   - Implement proper route filtering and aggregation
4. **Carrier-of-Carriers specifics**:
   - Always use labeled interfaces between CsC-PE and CE
   - Consider using segment routing for simpler operations
   - Monitor label stack depth (hardware limitations)
5. **Monitoring and troubleshooting**:
   - Monitor inter-AS link utilization
   - Track BGP prefix counts on ASBRs
   - Use BMP (BGP Monitoring Protocol) for visibility

Module 28: Circuit Cross-Connect

Part 1: The Conceptual Lecture (The Why)

Understanding the Need for Circuit Cross-Connect

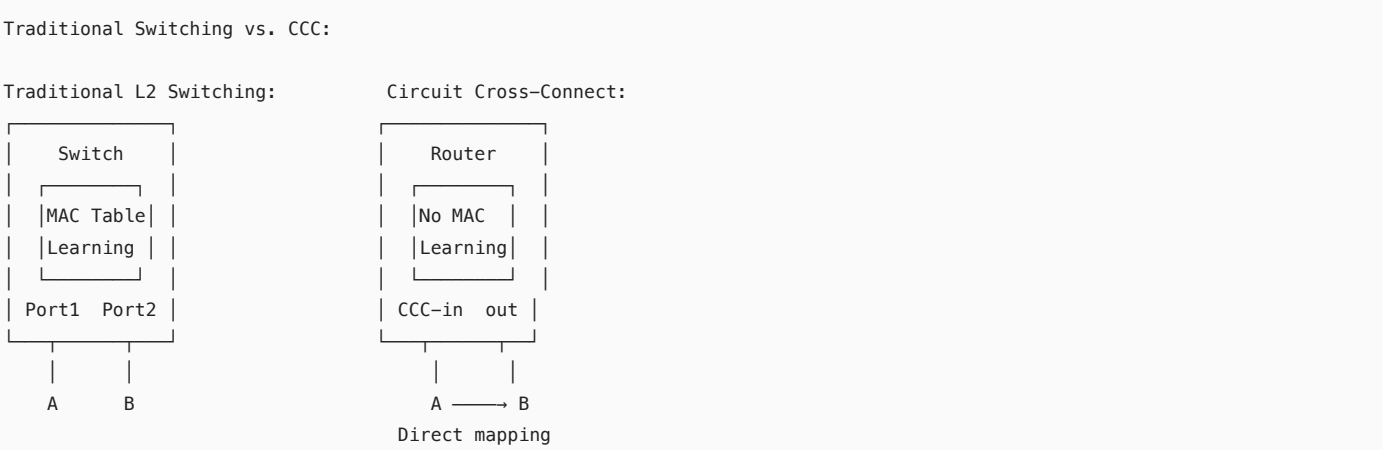
Imagine you're a service provider with a complex, multi-region network. A large enterprise customer wants a point-to-point Layer 2 connection between two sites, but these sites are:

- Located in different regions of your network
- Separated by multiple autonomous systems
- Requiring transit through a third-party provider's network

Traditional L2VPN solutions hit limitations here. You can't establish a single pseudowire across all these boundaries. This is where **Circuit Cross-Connect (CCC)** comes in - it allows you to "stitch" multiple pseudowires together or create specialized transport paths.

What is Circuit Cross-Connect?

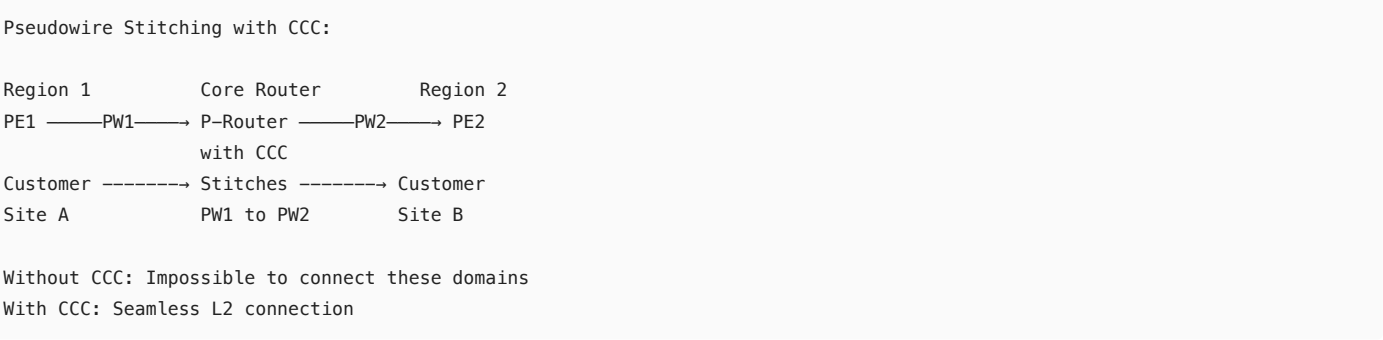
Circuit Cross-Connect creates a local switching relationship between two circuits on a router. Think of it as creating a "patch cable" inside the router that connects two interfaces at Layer 2.



CCC Use Cases

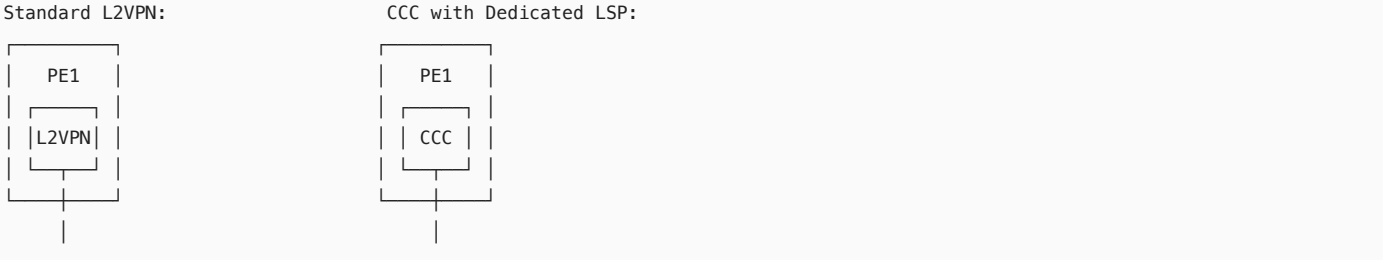
1. Pseudowire Stitching

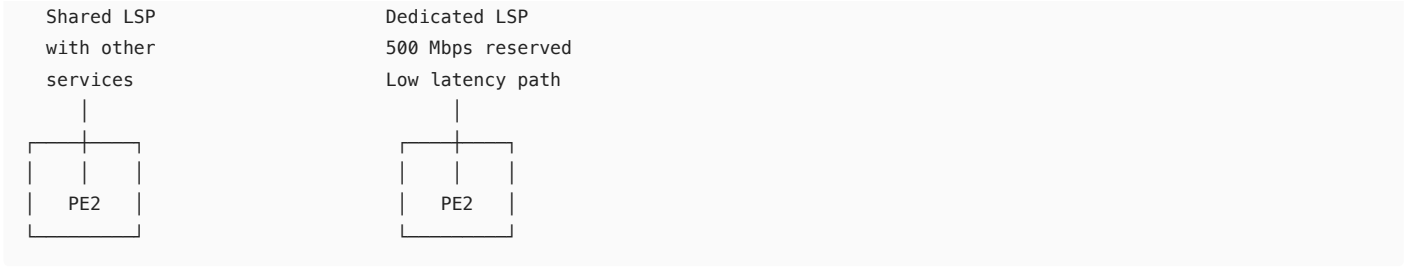
When you need to connect pseudowires from different L2VPN domains:



2. Dedicated RSVP LSPs for Pseudowires

Some pseudowires require guaranteed bandwidth or specific paths:





How CCC Works

CCC operates at the circuit level, meaning it works with the raw frames without any MAC learning or processing:

1. **Ingress Processing**:
   - Frames arrive on ingress interface
   - No MAC table lookup
   - No VLAN processing (unless specifically configured)
2. **Cross-Connect**:
   - Direct mapping to egress interface
   - Can map to physical interface, logical interface, or LSP
3. **Egress Processing**:
   - Frames transmitted exactly as received
   - No modifications (by default)

CCC Modes

1. Local Cross-Connect

Connects two local interfaces on the same router:

```
Router Internal Cross-Connect:
ge-0/0/1 ←──CCC──→ ge-0/0/2
```

2. Remote Cross-Connect

Connects local interface to remote PE via MPLS:

```
Local Interface ←──CCC──→ MPLS LSP → Remote PE
```

3. LSP Stitching

Connects two LSPs together:

```
Incoming LSP ←──CCC──→ Outgoing LSP
```

Translational Cross-Connect (TCC)

Sometimes you need to modify frames during cross-connect:

- Add/remove VLAN tags
- Swap VLAN IDs
- Change encapsulation types

TCC extends CCC with these capabilities:

```
TCC Operation Example:
Ingress: Frame with VLAN 100 → TCC → Egress: Frame with VLAN 200

Or:
Ingress: Ethernet frame → TCC → Egress: Frame over MPLS
```

Part 2: The Junos CLI Masterclass (The How)

CCC Configuration Types

1. Local Circuit Cross-Connect

Basic configuration connecting two local interfaces:

```
[edit interfaces]
set ge-0/0/1 encapsulation ethernet-ccc
set ge-0/0/1 unit 0

set ge-0/0/2 encapsulation ethernet-ccc
set ge-0/0/2 unit 0

[edit protocols connections]
set local-switching-interface interface-1 ge-0/0/1.0
set local-switching-interface interface-1 connection interface-2
set local-switching-interface interface-2 ge-0/0/2.0
```

Why:

- `ethernet-ccc` encapsulation enables CCC on the interface
- `local-switching-interface` creates the cross-connect

2. Remote Circuit Cross-Connect via RSVP

Connecting local interface to remote PE:

```
## Local interface configuration
[edit interfaces]
set ge-0/0/1 encapsulation ethernet-ccc
set ge-0/0/1 unit 0

## RSVP LSP configuration
[edit protocols mpls]
set label-switched-path TO-PE2 to 192.168.1.2
set label-switched-path TO-PE2 bandwidth 100m
set label-switched-path TO-PE2 primary STRICT-PATH

[edit protocols mpls path STRICT-PATH]
set 10.1.1.2 strict
set 10.1.2.2 strict
set 192.168.1.2 strict

## CCC configuration
[edit protocols connections]
set remote-interface-switch CCC-1 interface ge-0/0/1.0
set remote-interface-switch CCC-1 transmit-lsp TO-PE2
set remote-interface-switch CCC-1 receive-lsp FROM-PE2
```

Why:

- Dedicated RSVP LSP ensures bandwidth and path
- Separate transmit/receive LSPs for bidirectional traffic

3. Pseudowire Stitching

Connecting two L2VPN pseudowires:

```
## First, configure L2VPN interfaces for stitching
[edit interfaces]
set ge-0/0/1 encapsulation ethernet-ccc
set ge-0/0/1 unit 0 description "From Region 1 L2VPN"

set ge-0/0/2 encapsulation ethernet-ccc
set ge-0/0/2 unit 0 description "To Region 2 L2VPN"
```

```
## Configure L2VPN instances
[edit routing-instances]
set REGION1-L2VPN instance-type l2vpn
set REGION1-L2VPN interface ge-0/0/1.0
set REGION1-L2VPN route-distinguisher 192.168.1.1:100
set REGION1-L2VPN vrf-target target:65000:100
set REGION1-L2VPN protocols l2vpn encapsulation-type ethernet
set REGION1-L2VPN protocols l2vpn site STITCH site-identifier 1
set REGION1-L2VPN protocols l2vpn site STITCH interface ge-0/0/1.0

set REGION2-L2VPN instance-type l2vpn
set REGION2-L2VPN interface ge-0/0/2.0
set REGION2-L2VPN route-distinguisher 192.168.1.1:200
set REGION2-L2VPN vrf-target target:65000:200
set REGION2-L2VPN protocols l2vpn encapsulation-type ethernet
set REGION2-L2VPN protocols l2vpn site STITCH site-identifier 1
set REGION2-L2VPN protocols l2vpn site STITCH interface ge-0/0/2.0

## Create CCC between the two
[edit protocols connections]
set local-switching-interface interface-1 ge-0/0/1.0
set local-switching-interface interface-1 connection interface-2
set local-switching-interface interface-2 ge-0/0/2.0
```

Why: This creates a local stitch between two separate L2VPN domains.

4. Translational Cross-Connect (TCC)

VLAN translation example:

```
[edit interfaces]
set ge-0/0/1 flexible-vlan-tagging
set ge-0/0/1 encapsulation flexible-ethernet-services
set ge-0/0/1 unit 100 encapsulation vlan-ccc
set ge-0/0/1 unit 100 vlan-id 100

set ge-0/0/2 flexible-vlan-tagging
set ge-0/0/2 encapsulation flexible-ethernet-services
set ge-0/0/2 unit 200 encapsulation vlan-ccc
set ge-0/0/2 unit 200 vlan-id 200

[edit protocols connections]
set local-switching-interface interface-1 ge-0/0/1.100
set local-switching-interface interface-1 connection interface-2
set local-switching-interface interface-2 ge-0/0/2.200
```

Why: Frames entering on VLAN 100 exit on VLAN 200.

Complete Reference Configuration

```
## Physical Interfaces
set interfaces ge-0/0/1 description "Customer A - Site 1"
set interfaces ge-0/0/1 encapsulation ethernet-ccc
set interfaces ge-0/0/1 unit 0

set interfaces ge-0/0/2 description "To Core Network"
set interfaces ge-0/0/2 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/2 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.1.1/32

## MPLS Configuration
set protocols mpls interface ge-0/0/2.0
set protocols mpls label-switched-path CCC-LSP-TO-PE2 to 192.168.1.2
set protocols mpls label-switched-path CCC-LSP-TO-PE2 bandwidth 500m
set protocols mpls label-switched-path CCC-LSP-TO-PE2 priority 1 1
set protocols mpls label-switched-path CCC-LSP-TO-PE2 primary DEDICATED-PATH

set protocols mpls label-switched-path CCC-LSP-FROM-PE2 from 192.168.1.2
set protocols mpls path DEDICATED-PATH 10.1.1.2 strict
```

```
set protocols mpls path DEDICATED-PATH 10.1.2.2 strict
set protocols mpls path DEDICATED-PATH 192.168.1.2 strict

## RSVP Configuration
set protocols rsvp interface ge-0/0/2.0 bandwidth 1g

## CCC Configuration for Remote Connection
set protocols connections remote-interface-switch CUSTOMER-A-CCC interface ge-0/0/1.0
set protocols connections remote-interface-switch CUSTOMER-A-CCC transmit-lsp CCC-LSP-TO-PE2
set protocols connections remote-interface-switch CUSTOMER-A-CCC receive-lsp CCC-LSP-FROM-PE2

## BGP Configuration (for signaling CCC)
set protocols bgp group IBGP type internal
set protocols bgp group IBGP local-address 192.168.1.1
set protocols bgp group IBGP family l2vpn signaling
set protocols bgp group IBGP neighbor 192.168.1.2

## For Pseudowire Stitching Scenario
set interfaces ge-0/0/3 encapsulation ethernet-ccc
set interfaces ge-0/0/3 unit 0 description "From West Region"

set interfaces ge-0/0/4 encapsulation ethernet-ccc
set interfaces ge-0/0/4 unit 0 description "To East Region"

## L2VPN Instances for Stitching
set routing-instances WEST-L2VPN instance-type l2vpn
set routing-instances WEST-L2VPN interface ge-0/0/3.0
set routing-instances WEST-L2VPN route-distinguisher 192.168.1.1:300
set routing-instances WEST-L2VPN vrf-target target:65000:300
set routing-instances WEST-L2VPN protocols l2vpn encapsulation-type ethernet
set routing-instances WEST-L2VPN protocols l2vpn site PE1 site-identifier 1
set routing-instances WEST-L2VPN protocols l2vpn site PE1 interface ge-0/0/3.0 remote-site-id 2

set routing-instances EAST-L2VPN instance-type l2vpn
set routing-instances EAST-L2VPN interface ge-0/0/4.0
set routing-instances EAST-L2VPN route-distinguisher 192.168.1.1:400
set routing-instances EAST-L2VPN vrf-target target:65000:400
set routing-instances EAST-L2VPN protocols l2vpn encapsulation-type ethernet
set routing-instances EAST-L2VPN protocols l2vpn site PE1 site-identifier 1
set routing-instances EAST-L2VPN protocols l2vpn site PE1 interface ge-0/0/4.0 remote-site-id 3

## Local CCC Stitching
set protocols connections local-switching-interface west-interface ge-0/0/3.0
set protocols connections local-switching-interface west-interface connection east-interface
set protocols connections local-switching-interface east-interface ge-0/0/4.0
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify CCC Status

```
user@PE1> show connections
Connection/Circuit                    State
Local switching:
  ge-0/0/1.0 <-> ge-0/0/2.0           Up

Remote switching:
  CUSTOMER-A-CCC
    Interface: ge-0/0/1.0             Up
    Transmit LSP: CCC-LSP-TO-PE2      Up
    Receive LSP: CCC-LSP-FROM-PE2     Up
```

2. Verify CCC LSPs

```
user@PE1> show mpls lsp name CCC-LSP-TO-PE2 extensive
Ingress LSP: 1 sessions
192.168.1.2
  From: 192.168.1.1, State: Up, ActiveRoute: 0
```

```
  LSPname: CCC-LSP-TO-PE2, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: -
  Resv style: 1 FF, Label in: -, Label out: 299776
  Tspec: rate 500Mbps size 500Mbps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 45088 protocol 0
  PATH rcvfresh: 34928 sent: 35072 refresh: 30 secs
  Adspec: received MTU 1500 sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 10.1.1.2 (ge-0/0/2.0) 4 pkts
  RESV rcvfrom: 10.1.1.2 (ge-0/0/2.0) 4 pkts
```

3. Verify Interface Statistics

```
user@PE1> show interfaces ge-0/0/1.0 statistics
Logical interface ge-0/0/1.0 (Index 334)
  Packets: 1849284
  Bytes: 2182849284
  Tail-dropped packets: 0
  Packet error count: 0
```

Common Troubleshooting Scenarios

Scenario 1: CCC Connection Down

Symptom: Customer traffic not passing through CCC

Diagnostic Commands:

```
user@PE1> show connections
Connection/Circuit                    State
Local switching:
  ge-0/0/1.0 <-> ge-0/0/2.0           Down

user@PE1> show interfaces ge-0/0/1.0 | match CCC
  Encapsulation: Ethernet-CCC, missing remote binding
```

Cause: Interface not configured for CCC encapsulation

Solution:

```
[edit interfaces ge-0/0/1]
set encapsulation ethernet-ccc
delete unit 0 family ethernet-switching
commit
```

Scenario 2: Dedicated LSP Not Established

Symptom: CCC configured but using wrong path

Diagnostic Commands:

```
user@PE1> show mpls lsp name CCC-LSP-TO-PE2
CCC-LSP-TO-PE2        192.168.1.2      Down

user@PE1> show rsvp session
No RSVP sessions

user@PE1> show route protocol rsvp
No RSVP routes
```

Cause: RSVP not enabled on interfaces

Solution:

```
[edit protocols rsvp]
set interface ge-0/0/2.0
```

```
set interface all disable
commit

## Verify RSVP neighbors
user@PE1> show rsvp neighbor
```

Scenario 3: Pseudowire Stitching Not Working

Symptom: Both L2VPNs up but traffic not passing

Diagnostic Commands:

```
user@PE1> show l2vpn connections instance WEST-L2VPN
Instance: WEST-L2VPN
  Local site: PE1 (1)
    Interface: ge-0/0/3.0
    Status: Up

user@PE1> show connections | match ge-0/0/3
## No output - CCC not configured
```

Cause: CCC connection between interfaces missing

Solution:

```
[edit protocols connections]
set local-switching-interface west-interface ge-0/0/3.0
set local-switching-interface west-interface connection east-interface
set local-switching-interface east-interface ge-0/0/4.0
commit
```

Scenario 4: VLAN Translation Not Working

Symptom: Frames not being translated between VLANs

Diagnostic Commands:

```
user@PE1> show interfaces ge-0/0/1.100 | match Encap
  Encapsulation: VLAN-CCC

user@PE1> monitor interface traffic
## Shows input on ge-0/0/1.100 but no output on ge-0/0/2.200
```

Cause: Incorrect VLAN configuration or encapsulation mismatch

Solution:

```
[edit interfaces]
set ge-0/0/1 flexible-vlan-tagging
set ge-0/0/1 encapsulation flexible-ethernet-services
set ge-0/0/1 unit 100 encapsulation vlan-ccc
set ge-0/0/1 unit 100 vlan-id 100

## Same for other interface
set ge-0/0/2 flexible-vlan-tagging
set ge-0/0/2 encapsulation flexible-ethernet-services
set ge-0/0/2 unit 200 encapsulation vlan-ccc
set ge-0/0/2 unit 200 vlan-id 200
commit
```

Best Practices for Circuit Cross-Connect

1. **Choose the right CCC type**:
   - Local CCC for same-router connections
   - Remote CCC for dedicated bandwidth requirements
   - TCC when frame modification is needed
2. **Monitor bandwidth utilization**:

- CCC has no inherent QoS
- Use policers if rate limiting needed
- Monitor interface statistics regularly

3. **Plan for redundancy**:
   - CCC connections are stateless
   - Use link aggregation for local redundancy
   - Configure backup LSPs for remote CCC
4. **Documentation is critical**:
   - CCC connections are not visible in routing tables
   - Document all cross-connects thoroughly
   - Use meaningful descriptions on interfaces
5. **Testing considerations**:
   - Test with various frame sizes
   - Verify MTU consistency end-to-end
   - Check for frame loss during LSP changes

Module 29: Multisegment Pseudowires

Part 1: The Conceptual Lecture (The Why)

The Challenge of End-to-End Layer 2 Services

Imagine a multinational corporation with offices in New York, London, and Tokyo. They want a Layer 2 connection between all sites - essentially making their WAN feel like a giant switch. The challenge? These sites are served by different service providers:

- New York: Provider A (AS 65001)
- London: Provider B (AS 65002)
- Tokyo: Provider C (AS 65003)

Traditional pseudowires require a direct signaling relationship between endpoints. But PE routers in different providers can't directly signal to each other. This is where **Multisegment Pseudowires (MS-PW)** come in.

Understanding Pseudowire Segments

A multisegment pseudowire is like a relay race - multiple runners (segments) work together to complete the journey:

```
Traditional Single-Segment Pseudowire:
PE1 ←──────────Single Pseudowire──────────→ PE2
     Direct LDP/BGP signaling session

Multisegment Pseudowire:
PE1 ←──PW Seg1──→ S-PE1 ←──PW Seg2──→ S-PE2 ←──PW Seg3──→ PE2
NYC               London              Tokyo              Singapore
(AS 65001)        (AS 65002)          (AS 65003)         (AS 65003)

S-PE = Switching PE (Pseudowire Switching)
```

Key Concepts

1. Pseudowire Switching PE (S-PE)

An S-PE is a special PE router that:

- Terminates one pseudowire segment
- Initiates another pseudowire segment
- Switches Layer 2 traffic between segments
- Does NOT perform MAC learning

2. Terminating PE (T-PE)

The endpoints of the multisegment pseudowire:

- Connects to customer equipment

- Initiates/terminates the MS-PW
- Performs normal L2VPN functions

3. Pseudowire Switching

The process of connecting two pseudowire segments:

```
Pseudowire Switching Operation:

Segment 1 Label Stack:          Segment 2 Label Stack:
┌─────────────────┐               ┌─────────────────┐
│ Transport       │               │ Transport       │
│ Label: 1000     │               │ Label: 2000     │
├─────────────────┤               ├─────────────────┤
│ PW Label:       │   Switch at   │ PW Label:       │
│ 5000            │     S-PE      │ 6000            │
├─────────────────┤               ├─────────────────┤
│ L2 Payload      │ ──────────→   │ L2 Payload      │
└─────────────────┘               └─────────────────┘
```

How MS-PW Signaling Works

MS-PW can be signaled in two ways:

1. Static Configuration

Each segment manually configured:

- Administrator configures each segment separately
- No end-to-end signaling
- Simple but doesn't scale

2. Dynamic Signaling (T-LDP)

Targeted LDP with MS-PW extensions:

- Uses FEC 129 (Generalized PWid FEC)
- Includes S-PE information in signaling
- Automatic end-to-end path setup

```
Dynamic MS-PW Signaling Flow:

1. T-PE1 initiates: "I need PW to T-PE2"
   └→ Sends T-LDP Label Mapping with:
       - AGI (Attachment Group ID)
       - SAII/TAII (Source/Target identifiers)
       - Optional S-PE path

2. S-PE1 receives and processes:
   └→ Creates local switching entry
   └→ Initiates new segment to next hop

3. Process continues until reaching T-PE2

4. Return path established similarly
```

MS-PW Path Selection

How does the MS-PW know which S-PEs to traverse?

1. Explicit Path

Administrator specifies exact S-PE sequence:

```
Path: T-PE1 → S-PE1 → S-PE2 → T-PE2
```

2. Dynamic Path

S-PEs use routing information to find next hop:

- BGP-based autodiscovery
- IGP-based computation
- Policy-based selection

Advantages of Multisegment Pseudowires

1. **Provider Independence**: Each provider manages their segment
2. **Scalability**: T-PEs don't need full mesh connectivity
3. **Policy Control**: Each segment can have different characteristics
4. **Troubleshooting**: Issues isolated to specific segments

MS-PW Control Word and Sequencing

The control word becomes crucial in MS-PW:

```
Control Word Format (4 bytes):
┌────┬────┬──────────────────┐
│0000│Flags│ Sequence Number│
└────┴────┴──────────────────┘

Purpose:
- Maintains packet order across segments
- Detects packet loss
- Provides OAM capabilities
```

Part 2: The Junos CLI Masterclass (The How)

Configuration Approaches

We'll cover both static and dynamic MS-PW configurations.

1. Static MS-PW Configuration on T-PE

First, configure the T-PE (edge router):

```
## Interface facing customer
[edit interfaces]
set ge-0/0/0 description "Customer Site A"
set ge-0/0/0 encapsulation ethernet-ccc
set ge-0/0/0 unit 0

## L2VPN instance with static MS-PW
[edit routing-instances]
set MS-PW-STATIC instance-type l2vpn
set MS-PW-STATIC interface ge-0/0/0.0
set MS-PW-STATIC route-distinguisher 192.168.1.1:100
set MS-PW-STATIC vrf-target target:65001:100
set MS-PW-STATIC protocols l2vpn encapsulation-type ethernet
set MS-PW-STATIC protocols l2vpn control-word
set MS-PW-STATIC protocols l2vpn site SITE-A site-identifier 1
set MS-PW-STATIC protocols l2vpn site SITE-A interface ge-0/0/0.0
```

Why: Standard L2VPN configuration but with control-word mandatory for MS-PW.

2. S-PE Configuration for Static MS-PW

Configure the S-PE for pseudowire switching:

```
## Enable pseudowire switching globally
[edit protocols l2vpn]
set pseudowire-switching

## Configure incoming and outgoing segments
[edit protocols l2vpn pseudowire-switching]
set instance MS-PW-SWITCH
set instance MS-PW-SWITCH neighbor 10.1.1.1 psn-tunnel-endpoint 10.1.1.1
set instance MS-PW-SWITCH neighbor 10.1.1.1 vc-id 100
set instance MS-PW-SWITCH neighbor 10.1.1.1 control-word

set instance MS-PW-SWITCH neighbor 10.2.2.2 psn-tunnel-endpoint 10.2.2.2
set instance MS-PW-SWITCH neighbor 10.2.2.2 vc-id 200
set instance MS-PW-SWITCH neighbor 10.2.2.2 control-word
```

Why: Creates switching relationship between two pseudowire segments.

3. Dynamic MS-PW with T-LDP

Configure dynamic signaling using FEC 129:

```
## On T-PE1
[edit routing-instances MS-PW-DYNAMIC]
set instance-type l2vpn
set interface ge-0/0/0.0
set route-distinguisher 192.168.1.1:200
set vrf-target target:65001:200
set protocols l2vpn encapsulation-type ethernet
set protocols l2vpn control-word
set protocols l2vpn site SITE-A site-identifier 1
set protocols l2vpn site SITE-A interface ge-0/0/0.0
set protocols l2vpn site SITE-A remote-site-id 2

## Configure MS-PW parameters
set protocols l2vpn multisegment-pseudowire AGI 65001:100
set protocols l2vpn multisegment-pseudowire SAII 192.168.1.1:1
set protocols l2vpn multisegment-pseudowire TAII 192.168.3.3:2
```

Why:

- AGI (Attachment Group Identifier) identifies the VPN
- SAII/TAII identify source and target attachment circuits

4. Explicit Path Configuration

Specify exact S-PE path:

```
[edit protocols l2vpn]
set multisegment-pseudowire explicit-path PATH1
set multisegment-pseudowire explicit-path PATH1 s-pe 192.168.2.1
set multisegment-pseudowire explicit-path PATH1 s-pe 192.168.2.2

[edit routing-instances MS-PW-DYNAMIC protocols l2vpn]
set multisegment-pseudowire explicit-path PATH1
```

Why: Forces MS-PW through specific S-PEs for policy or SLA requirements.

5. S-PE with BGP Auto-Discovery

Enable automatic S-PE discovery:

```
## BGP configuration for MS-PW autodiscovery
[edit protocols bgp]
set group MS-PW-AD type internal
set group MS-PW-AD local-address 192.168.2.1
set group MS-PW-AD family l2vpn auto-discovery-mspw
set group MS-PW-AD neighbor 192.168.2.2
set group MS-PW-AD neighbor 192.168.2.3
```

```
## Enable on S-PE
[edit protocols l2vpn]
set auto-discovery-mspw
set route-distinguisher 192.168.2.1:1
set vrf-target target:65001:999
```

Why: S-PEs advertise their capability and learn about other S-PEs dynamically.

Complete MS-PW Reference Configuration

```
## T-PE1 Configuration (New York)
## Physical Interfaces
set interfaces ge-0/0/0 description "Customer NYC Office"
set interfaces ge-0/0/0 encapsulation ethernet-ccc
set interfaces ge-0/0/0 unit 0

set interfaces ge-0/0/1 description "To MPLS Core"
set interfaces ge-0/0/1 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/1 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.1.1/32

## MPLS and LDP
set protocols mpls interface ge-0/0/1.0
set protocols ldp interface ge-0/0/1.0
set protocols ldp interface lo0.0

## Dynamic MS-PW Instance
set routing-instances MS-PW-CUSTOMER-A instance-type l2vpn
set routing-instances MS-PW-CUSTOMER-A interface ge-0/0/0.0
set routing-instances MS-PW-CUSTOMER-A route-distinguisher 192.168.1.1:300
set routing-instances MS-PW-CUSTOMER-A vrf-target target:65001:300
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn encapsulation-type ethernet
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn control-word
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn mtu 1500
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn site NYC site-identifier 1
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn site NYC interface ge-0/0/0.0

## MS-PW Signaling Parameters
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment enable
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment signaling-protocol t-ldp
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment attachment-group-identifier 65001:300
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment source-attachment-identifier 192.168.1.1:1
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment target-attachment-identifier 192.168.3.3:1
set routing-instances MS-PW-CUSTOMER-A protocols l2vpn multisegment explicit-path PREFERRED-PATH

## Explicit Path Definition
set protocols l2vpn multisegment-pseudowire-path PREFERRED-PATH
set protocols l2vpn multisegment-pseudowire-path PREFERRED-PATH s-pe 192.168.2.1
set protocols l2vpn multisegment-pseudowire-path PREFERRED-PATH s-pe 192.168.2.2

## S-PE1 Configuration (London)
set interfaces ge-0/0/1 description "From AS 65001"
set interfaces ge-0/0/1 unit 0 family inet address 10.1.1.2/30
set interfaces ge-0/0/1 unit 0 family mpls

set interfaces ge-0/0/2 description "To AS 65003"
set interfaces ge-0/0/2 unit 0 family inet address 10.2.1.1/30
set interfaces ge-0/0/2 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.2.1/32

## Enable MS-PW Switching
set protocols l2vpn pseudowire-switching

## Configure PW Switching Instance
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH description "NYC to Singapore MS-PW"
```

```
## Segment from T-PE1
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment1
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment1 neighbor 192.168.1.1
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment1 psn-tunnel-endpoint 192.168.1.1
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment1 control-word
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment1 attachment-group-identifier 65001:300

## Segment to next S-PE
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment2
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment2 neighbor 192.168.2.2
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment2 psn-tunnel-endpoint 192.168.2.2
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment2 control-word
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH segment segment2 attachment-group-identifier 65001:300

## Connect the segments
set protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH local-switching-instance segment1 segment2

## BGP for MS-PW Auto-discovery
set protocols bgp group MS-PW-AD type internal
set protocols bgp group MS-PW-AD local-address 192.168.2.1
set protocols bgp group MS-PW-AD family l2vpn auto-discovery-mspw
set protocols bgp group MS-PW-AD neighbor 192.168.1.100
set protocols bg
```

set protocols bgp group MS-PW-AD neighbor 192.168.2.2

LDP Configuration with MS-PW Extensions

set protocols ldp interface ge-0/0/1.0 set protocols ldp interface ge-0/0/2.0
set protocols ldp interface lo0.0 set protocols ldp targeted-hello accept-from 0.0.0.0/0 set protocols ldp session 192.168.1.1 authentication-key "mspw-key" set protocols ldp session 192.168.2.2 authentication-key "mspw-key"

MPLS

set protocols mpls interface ge-0/0/1.0 set protocols mpls interface ge-0/0/2.0

```


## Part 3: Verification & Troubleshooting (The What-If)

### Essential Verification Commands

#### 1. Verify MS-PW Status on T-PE

```bash
user@T-PE1> show l2vpn connections instance MS-PW-CUSTOMER-A extensive
Instance: MS-PW-CUSTOMER-A
  Local site: NYC (1)
    Interface: ge-0/0/0.0
    Status: Up
    Encapsulation: ETHERNET
    Description: Customer NYC Office

  Remote PE: 192.168.3.3, Negotiated control-word: Yes
  Incoming label: 800100, Outgoing label: 800200
  Negotiated PW status TLV: Yes
  Local PW status code: 0x00000000, Remote PW status code: 0x00000000

  MS-PW Information:
    AGI: 65001:300
    SAII: 192.168.1.1:1, TAII: 192.168.3.3:1
    Path: 192.168.1.1 -> 192.168.2.1 -> 192.168.2.2 -> 192.168.3.3
    Segment count: 3
```

2. Verify S-PE Switching Status
```

```
user@S-PE1> show l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH
Instance: CUSTOMER-A-SWITCH
  Description: NYC to Singapore MS-PW

  Segment: segment1
    Neighbor: 192.168.1.1
    VC ID: Dynamic
    Status: Up
    Local label: 800100, Remote label: 800200
    Control word: Yes

  Segment: segment2
    Neighbor: 192.168.2.2
    VC ID: Dynamic
    Status: Up
    Local label: 900100, Remote label: 900200
    Control word: Yes

  Switching: segment1 <-> segment2
    Status: Active
    Packets switched: 1849284
```

3. Verify MS-PW Path

```
user@T-PE1> show l2vpn connections path instance MS-PW-CUSTOMER-A
MS-PW Path for instance MS-PW-CUSTOMER-A:
  192.168.1.1 (Local T-PE)
    -> 192.168.2.1 (S-PE) [AS 65002]
    -> 192.168.2.2 (S-PE) [AS 65003]
    -> 192.168.3.3 (Remote T-PE) [AS 65003]

  Total segments: 3
  Path status: Complete
```

4. Verify LDP Sessions for MS-PW

```
user@S-PE1> show ldp session detail
Address: 192.168.2.1, State: Operational, Connection: Open
  Session ID: 192.168.1.1:0--192.168.2.1:0
  Next keepalive in 8 seconds
  Passive, Maximum PDU: 4096, Hold time: 30, Neighbor count: 1
  Capabilities advertised: MS-PW
  Capabilities received: MS-PW
  Neighbor types: targeted
  Received LDP Identifier: 192.168.1.1:0
  Negotiated hold time: 30
  MS-PW TLVs supported: Yes
```

Common Troubleshooting Scenarios

Scenario 1: MS-PW Not Establishing End-to-End

Symptom: T-PE shows pseudowire down

Diagnostic Commands:

```
user@T-PE1> show l2vpn connections instance MS-PW-CUSTOMER-A
Instance: MS-PW-CUSTOMER-A
  Local site: NYC (1)
    Interface: ge-0/0/0.0
    Status: Down (No remote connection)

user@T-PE1> show log messages | match "L2VPN|MS-PW"
Nov 10 14:30:12 T-PE1 L2VPN: MS-PW path setup failed: No route to TAII
```

Cause: S-PEs not advertising MS-PW capability or path broken

Solution:

```
## On each S-PE, ensure MS-PW is enabled:
[edit protocols l2vpn]
set pseudowire-switching
set auto-discovery-mspw

## Verify BGP sessions for MS-PW autodiscovery:
user@S-PE1> show bgp summary family l2vpn-auto-discovery-mspw
```

Scenario 2: Control Word Negotiation Failure

Symptom: Segments up but traffic not passing

Diagnostic Commands:

```
user@S-PE1> show l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH detail
Segment: segment1
  Control word: No (Mismatch)
  Local capability: Yes, Remote capability: No

user@S-PE1> show interfaces statistics | match error
  Frame errors: 28491 (incrementing)
```

Cause: Control word mismatch between segments

Solution:

```
## Ensure control word is configured on all segments:
[edit protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH]
set segment segment1 control-word
set segment segment2 control-word
commit

## On T-PEs:
[edit routing-instances MS-PW-CUSTOMER-A protocols l2vpn]
set control-word
```

Scenario 3: S-PE Not Switching Between Segments

Symptom: Individual segments up but no end-to-end connectivity

Diagnostic Commands:

```
user@S-PE1> show l2vpn pseudowire-switching statistics
Instance: CUSTOMER-A-SWITCH
  Segment1 -> Segment2: 0 packets
  Segment2 -> Segment1: 0 packets

user@S-PE1> show configuration protocols l2vpn pseudowire-switching
## Missing local-switching-instance configuration
```

Cause: Segments not connected for switching

Solution:

```
[edit protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH]
set local-switching-instance segment1 segment2
commit

## Verify switching active:
user@S-PE1> show l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH | match Switching
  Switching: segment1 <-> segment2, Status: Active
```

Scenario 4: MTU Mismatch Across Segments

Symptom: Small packets work, large packets dropped

Diagnostic Commands:

```
user@T-PE1> ping mpls l2vpn instance MS-PW-CUSTOMER-A size 1400
Success rate: 100%

user@T-PE1> ping mpls l2vpn instance MS-PW-CUSTOMER-A size 1500
Success rate: 0%

user@S-PE1> show interfaces ge-0/0/1 | match MTU
  Link-level type: Ethernet, MTU: 1514
user@S-PE1> show interfaces ge-0/0/2 | match MTU
  Link-level type: Ethernet, MTU: 1500
```

Cause: Different MTU values on S-PE interfaces

Solution:

```
## Set consistent MTU across all interfaces in the path:
[edit interfaces ge-0/0/2]
set mtu 1514

## Configure MS-PW MTU signaling:
[edit protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH]
set mtu 1500
```

Advanced MS-PW Features

1. MS-PW OAM Configuration

```
## Enable VCCV (Virtual Circuit Connectivity Verification)
[edit routing-instances MS-PW-CUSTOMER-A protocols l2vpn]
set oam ping-interval 10
set oam bfd-liveness-detection minimum-interval 100
set oam bfd-liveness-detection multiplier 3
```

2. MS-PW Load Balancing

```
## Configure multiple paths
[edit protocols l2vpn]
set multisegment-pseudowire-path PATH1 s-pe 192.168.2.1
set multisegment-pseudowire-path PATH1 s-pe 192.168.2.2

set multisegment-pseudowire-path PATH2 s-pe 192.168.2.3
set multisegment-pseudowire-path PATH2 s-pe 192.168.2.4

[edit routing-instances MS-PW-CUSTOMER-A protocols l2vpn]
set multisegment primary-path PATH1
set multisegment backup-path PATH2
```

3. MS-PW Bandwidth Reservation

```
## Configure bandwidth parameters
[edit protocols l2vpn pseudowire-switching instance CUSTOMER-A-SWITCH]
set segment segment1 bandwidth 100m
set segment segment2 bandwidth 100m
```

Best Practices for Multisegment Pseudowires

1. **Always use control words** - Essential for proper operation across multiple segments
2. **Plan S-PE placement carefully**:
   - Minimize hop count
   - Consider redundant S-PEs
   - Place S-PEs at AS boundaries
3. **Monitor segment health**:
   - Enable OAM on all segments

- Monitor packet loss and latency
- Set up alerts for segment failures

4. **Document the MS-PW path**:
   - Maintain accurate topology diagrams
   - Document AGI/SAII/TAII mappings
   - Track which S-PEs handle which services
5. **MTU considerations**:
   - Account for additional labels in MS-PW
   - Ensure consistent MTU across all segments
   - Test with maximum-sized frames
6. **Security considerations**:
   - Authenticate LDP sessions between S-PEs
   - Filter MS-PW signaling at AS boundaries
   - Consider encryption for sensitive traffic

Module 30: VPLS—Hub-and-Spoke Topologies

Part 1: The Conceptual Lecture (The Why)

Understanding the Need for Hub-and-Spoke VPLS

In standard VPLS deployments, all sites have full-mesh connectivity - every site can directly communicate with every other site. However, many organizations have hierarchical network requirements:

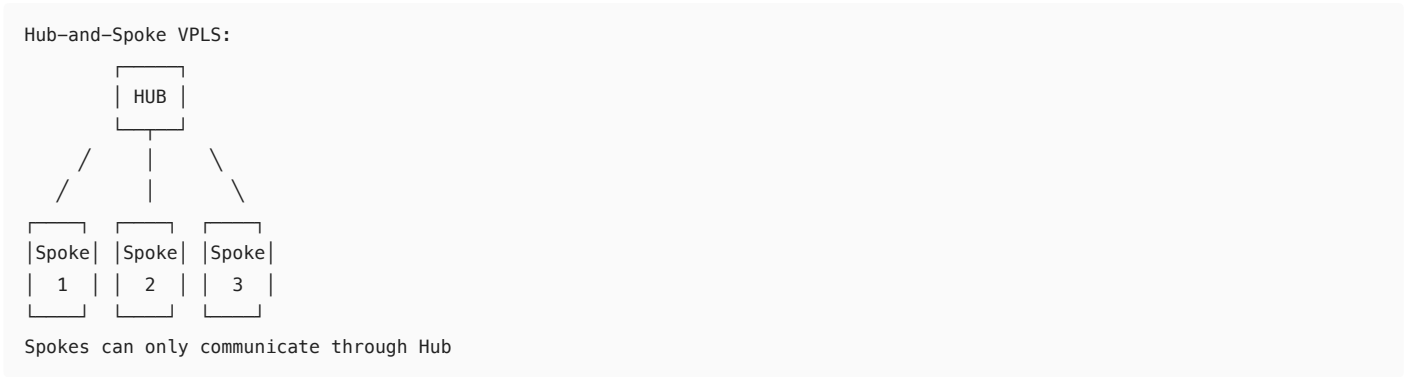
- **Retail chains**: Stores should only communicate with headquarters, not with each other
- **Financial institutions**: Branch offices connect through regional hubs for security and compliance
- **Educational networks**: Schools connect through district offices for content filtering and monitoring

This hierarchical requirement is where **Hub-and-Spoke VPLS** becomes essential.

The Challenge with Standard VPLS

Standard VPLS creates a full-mesh topology:

But what if you need this instead?



Three Methods for Hub-and-Spoke VPLS

Method 1: BGP VPLS with Route Targets

This method uses different route targets to control which sites can communicate:

How it works:

- Hub exports one route target and imports another
- Spokes export the RT that hub imports
- Spokes import the RT that hub exports
- Result: Asymmetric RT configuration creates hub-and-spoke topology

```
Route Target Flow:
Hub:    Export RT: 65000:100, Import RT: 65000:200
Spoke1: Export RT: 65000:200, Import RT: 65000:100
Spoke2: Export RT: 65000:200, Import RT: 65000:100

Spoke1 → (RT:65000:200) → Hub accepts
Hub → (RT:65000:100) → Spoke1 accepts
Spoke1 → (RT:65000:200) → Spoke2 rejects (doesn't import 65000:200)
```

Method 2: BGP VPLS with Site Ranges

Site ranges provide a more scalable approach using numeric ranges:

How it works:

- Sites are assigned numeric IDs
- Configure ranges to determine which sites can communicate
- Hub configured with range covering all spokes
- Spokes configured with range covering only hub

```
Site ID Assignment:
Hub:    Site ID = 1
Spoke1: Site ID = 101
Spoke2: Site ID = 102
Spoke3: Site ID = 103

Range Configuration:
Hub:    Accept sites 100-199 (all spokes)
Spokes: Accept sites 1-10 (only hub)
```

Method 3: Hierarchical VPLS (H-VPLS) with LDP

H-VPLS creates a two-tier architecture:

How it works:

- Spoke sites connect to hub via spoke pseudowires
- Hub PE performs VPLS switching
- Optional mesh group at hub level
- LDP signaling for pseudowires

```
H-VPLS Architecture:
        Core VPLS Mesh
     ┌────┐    ┌────┐
     │PE-H│    │PE-H│
     └──┬─┘    └──┬─┘
        │         │
    Spoke PWs  Spoke PWs
     /  |  \     |
  PE-S PE-S PE-S PE-S
```

```
   |    |    |    |
 Site  Site Site Site
```

Why Use Hub-and-Spoke?

1. **Security and Control**
   - Centralized security inspection
   - Content filtering at hub
   - Compliance monitoring
2. **Simplified Routing**
   - Spoke sites need minimal configuration
   - All routing decisions at hub
   - Easier troubleshooting
3. **Cost Optimization**
   - Spokes can have smaller, cheaper connections
   - Centralized services at hub
   - Reduced complexity at spoke sites
4. **Service Integration**
   - Firewall, IDS/IPS at hub
   - WAN optimization
   - Centralized internet breakout

Part 2: The Junos CLI Masterclass (The How)

Method 1: BGP VPLS with Route Targets Configuration

Hub PE Configuration

```
## Interface configuration
[edit interfaces]
set ge-0/0/0 description "To Hub Site"
set ge-0/0/0 unit 0 family vpls

set lo0 unit 0 family inet address 192.168.1.1/32

## VPLS Instance - Hub
[edit routing-instances VPLS-HUB]
set instance-type vpls
set interface ge-0/0/0.0
set route-distinguisher 192.168.1.1:100
set vrf-export HUB-EXPORT
set vrf-import HUB-IMPORT
set protocols vpls site-range 10
set protocols vpls site HUB site-identifier 1

## Policy for Hub RT handling
[edit policy-options]
set policy-statement HUB-EXPORT term 1 then community add HUB-COMM
set policy-statement HUB-EXPORT term 1 then accept

set policy-statement HUB-IMPORT term 1 from community SPOKE-COMM
set policy-statement HUB-IMPORT term 1 then accept

set community HUB-COMM members target:65000:100
set community SPOKE-COMM members target:65000:200
```

Why:

- Hub exports RT 65000:100 (which spokes will import)
- Hub imports RT 65000:200 (which spokes will export)
- This creates the asymmetric RT relationship

Spoke PE Configuration

```
## Interface configuration
[edit interfaces]
set ge-0/0/0 description "To Spoke Site 1"
set ge-0/0/0 unit 0 family vpls

set lo0 unit 0 family inet address 192.168.1.101/32

## VPLS Instance - Spoke
[edit routing-instances VPLS-SPOKE1]
set instance-type vpls
set interface ge-0/0/0.0
set route-distinguisher 192.168.1.101:100
set vrf-export SPOKE-EXPORT
set vrf-import SPOKE-IMPORT
set protocols vpls site-range 10
set protocols vpls site SPOKE1 site-identifier 101

## Policy for Spoke RT handling
[edit policy-options]
set policy-statement SPOKE-EXPORT term 1 then community add SPOKE-COMM
set policy-statement SPOKE-EXPORT term 1 then accept

set policy-statement SPOKE-IMPORT term 1 from community HUB-COMM
set policy-statement SPOKE-IMPORT term 1 then accept

set community HUB-COMM members target:65000:100
set community SPOKE-COMM members target:65000:200
```

Why:

- Spoke exports RT 65000:200 (which hub will import)
- Spoke imports RT 65000:100 (which hub exports)
- Spoke won't import RT 65000:200 from other spokes

Method 2: BGP VPLS with Site Ranges Configuration

Hub PE Configuration with Site Ranges

```
## VPLS Instance with Site Range
[edit routing-instances VPLS-HUB-RANGE]
set instance-type vpls
set interface ge-0/0/0.0
set route-distinguisher 192.168.1.1:200
set vrf-target target:65000:300
set protocols vpls site-range 200
set protocols vpls site HUB site-identifier 1
set protocols vpls site HUB interface ge-0/0/0.0
```

Why: Site range 200 means hub will accept sites 1-200

Spoke PE Configuration with Site Ranges

```
## VPLS Instance with Limited Site Range
[edit routing-instances VPLS-SPOKE-RANGE]
set instance-type vpls
set interface ge-0/0/0.0
set route-distinguisher 192.168.1.101:200
set vrf-target target:65000:300
set protocols vpls site-range 10
set protocols vpls site SPOKE1 site-identifier 101
set protocols vpls site SPOKE1 interface ge-0/0/0.0
```

Why: Site range 10 means spoke only accepts sites 1-10 (hub is site 1)

Method 3: Hierarchical VPLS (H-VPLS) Configuration

Hub PE Configuration for H-VPLS

```
## Configure main VPLS instance
[edit routing-instances H-VPLS-HUB]
set instance-type vpls
set interface ge-0/0/0.0
set protocols vpls vpls-id 100
set protocols vpls neighbor 192.168.1.2

## Configure spoke connections
set protocols vpls neighbor 192.168.1.101 static incoming-label 100101
set protocols vpls neighbor 192.168.1.101 static outgoing-label 101100
set protocols vpls neighbor 192.168.1.101 static interface ge-0/0/1.0

set protocols vpls neighbor 192.168.1.102 static incoming-label 100102
set protocols vpls neighbor 192.168.1.102 static outgoing-label 102100
set protocols vpls neighbor 192.168.1.102 static interface ge-0/0/2.0
```

Why: Static configuration creates spoke pseudowires without full mesh

Spoke PE Configuration for H-VPLS

```
## Simple spoke configuration
[edit routing-instances H-VPLS-SPOKE]
set instance-type l2vpn
set interface ge-0/0/0.0
set protocols l2vpn encapsulation-type ethernet
set protocols l2vpn site SPOKE1 site-identifier 1
set protocols l2vpn site SPOKE1 interface ge-0/0/0.0
set protocols l2vpn site SPOKE1 remote-site-id 100
set protocols l2vpn neighbor 192.168.1.1 static incoming-label 101100
set protocols l2vpn neighbor 192.168.1.1 static outgoing-label 100101
```

Why: Spoke uses simple L2VPN to connect to hub VPLS instance

Complete Reference Configuration

Here's a complete hub-and-spoke VPLS deployment using BGP with route targets:

```
## HUB PE Configuration (PE1)
## Physical Interfaces
set interfaces ge-0/0/0 description "To Hub Customer Site"
set interfaces ge-0/0/0 flexible-vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 100 vlan-id 100
set interfaces ge-0/0/0 unit 100 family vpls

set interfaces ge-0/0/1 description "To MPLS Core"
set interfaces ge-0/0/1 unit 0 family inet address 10.1.1.1/30
set interfaces ge-0/0/1 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.1.1/32

## MPLS and LDP
set protocols mpls interface ge-0/0/1.0
set protocols ldp interface ge-0/0/1.0
set protocols ldp interface lo0.0

## BGP Configuration
set protocols bgp group VPLS-PE type internal
set protocols bgp group VPLS-PE local-address 192.168.1.1
set protocols bgp group VPLS-PE family l2vpn signaling
set protocols bgp group VPLS-PE neighbor 192.168.1.101
set protocols bgp group VPLS-PE neighbor 192.168.1.102
set protocols bgp group VPLS-PE neighbor 192.168.1.103

## VPLS Instance Configuration
set routing-instances HUB-SPOKE-VPLS instance-type vpls
set routing-instances HUB-SPOKE-VPLS interface ge-0/0/0.100
set routing-instances HUB-SPOKE-VPLS route-distinguisher 192.168.1.1:1000
set routing-instances HUB-SPOKE-VPLS vrf-export HUB-EXPORT-POLICY
```

```
set routing-instances HUB-SPOKE-VPLS vrf-import HUB-IMPORT-POLICY
set routing-instances HUB-SPOKE-VPLS protocols vpls site-range 500
set routing-instances HUB-SPOKE-VPLS protocols vpls no-tunnel-services
set routing-instances HUB-SPOKE-VPLS protocols vpls site HUB-SITE site-identifier 1
set routing-instances HUB-SPOKE-VPLS protocols vpls site HUB-SITE interface ge-0/0/0.100

## Policy Configuration for Hub
set policy-options policy-statement HUB-EXPORT-POLICY term 1 then community add HUB-RT
set policy-options policy-statement HUB-EXPORT-POLICY term 1 then accept

set policy-options policy-statement HUB-IMPORT-POLICY term 1 from community SPOKE-RT
set policy-options policy-statement HUB-IMPORT-POLICY term 1 then accept
set policy-options policy-statement HUB-IMPORT-POLICY term 2 then reject

set policy-options community HUB-RT members target:65000:1000
set policy-options community SPOKE-RT members target:65000:2000

## Spoke PE Configuration (PE2)
## Physical Interfaces
set interfaces ge-0/0/0 description "To Spoke Site 1"
set interfaces ge-0/0/0 flexible-vlan-tagging
set interfaces ge-0/0/0 encapsulation flexible-ethernet-services
set interfaces ge-0/0/0 unit 200 vlan-id 200
set interfaces ge-0/0/0 unit 200 family vpls

set interfaces ge-0/0/1 description "To MPLS Core"
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.1/30
set interfaces ge-0/0/1 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.1.101/32

## VPLS Instance Configuration
set routing-instances HUB-SPOKE-VPLS instance-type vpls
set routing-instances HUB-SPOKE-VPLS interface ge-0/0/0.200
set routing-instances HUB-SPOKE-VPLS route-distinguisher 192.168.1.101:1000
set routing-instances HUB-SPOKE-VPLS vrf-export SPOKE-EXPORT-POLICY
set routing-instances HUB-SPOKE-VPLS vrf-import SPOKE-IMPORT-POLICY
set routing-instances HUB-SPOKE-VPLS protocols vpls site-range 500
set routing-instances HUB-SPOKE-VPLS protocols vpls no-tunnel-services
set routing-instances HUB-SPOKE-VPLS protocols vpls site SPOKE1 site-identifier 101
set routing-instances HUB-SPOKE-VPLS protocols vpls site SPOKE1 interface ge-0/0/0.200

## Policy Configuration for Spoke
set policy-options policy-statement SPOKE-EXPORT-POLICY term 1 then community add SPOKE-RT
set policy-options policy-statement SPOKE-EXPORT-POLICY term 1 then accept

set policy-options policy-statement SPOKE-IMPORT-POLICY term 1 from community HUB-RT
set policy-options policy-statement SPOKE-IMPORT-POLICY term 1 then accept
set policy-options policy-statement SPOKE-IMPORT-POLICY term 2 then reject

set policy-options community HUB-RT members target:65000:1000
set policy-options community SPOKE-RT members target:65000:2000

## Additional Features for Hub
## MAC limiting on spokes
set routing-instances HUB-SPOKE-VPLS protocols vpls site HUB-SITE mac-limit 1000
set routing-instances HUB-SPOKE-VPLS protocols vpls site HUB-SITE mac-limit packet-action drop

## Storm control
set routing-instances HUB-SPOKE-VPLS forwarding-options flooding broadcast bandwidth-percentage 10
set routing-instances HUB-SPOKE-VPLS forwarding-options flooding unknown-unicast bandwidth-percentage 10
```

Part 3: Verification & Troubleshooting (The What-If)

Essential Verification Commands

1. Verify Hub-and-Spoke Topology

```
user@HUB-PE> show vpls connections instance HUB-SPOKE-VPLS
Instance: HUB-SPOKE-VPLS
```

```
  VPLS-id: (none)
  Neighbor                Type  St     Time last up          # Up trans
  192.168.1.101(vpls-id 0)  rmt   Up     Nov 10 10:00:00 2023         1
    Remote PE: 192.168.1.101, Negotiated control-word: No
    Incoming label: 262145, Outgoing label: 262146
    Site-ID: 101, Site-Type: spoke
  192.168.1.102(vpls-id 0)  rmt   Up     Nov 10 10:00:00 2023         1
    Remote PE: 192.168.1.102, Negotiated control-word: No
    Incoming label: 262147, Outgoing label: 262148
    Site-ID: 102, Site-Type: spoke
```

2. Verify Route Target Import/Export

```
user@HUB-PE> show route instance HUB-SPOKE-VPLS detail | match "Import|Export|target"
  Import: [ target:65000:2000 ]
  Export: [ target:65000:1000 ]

user@SPOKE-PE> show route instance HUB-SPOKE-VPLS detail | match "Import|Export|target"
  Import: [ target:65000:1000 ]
  Export: [ target:65000:2000 ]
```

3. Verify MAC Learning

```
user@HUB-PE> show vpls mac-table instance HUB-SPOKE-VPLS
MAC address        Logical interface         Active source
00:11:22:33:44:55  ge-0/0/0.100             ge-0/0/0.100
00:11:22:33:44:66  192.168.1.101(vpls-id 0) 192.168.1.101
00:11:22:33:44:77  192.168.1.102(vpls-id 0) 192.168.1.102
```

Common Troubleshooting Scenarios

Scenario 1: Spoke-to-Spoke Communication Working

Symptom: Spokes can reach each other directly (breaking hub-and-spoke design)

Diagnostic Commands:

```
user@SPOKE1-PE> show route receive-protocol bgp 192.168.1.102 table HUB-SPOKE-VPLS.l2vpn.0
## Shows routes from SPOKE2 - shouldn't happen

user@SPOKE1-PE> show configuration policy-options policy-statement SPOKE-IMPORT-POLICY
## Check if accidentally importing spoke RT
```

Cause: Incorrect RT configuration allowing spoke-to-spoke import

Solution:

```
[edit policy-options policy-statement SPOKE-IMPORT-POLICY]
delete term 1 from community SPOKE-RT
set term 1 from community HUB-RT
set term 2 then reject
commit
```

Scenario 2: No Connectivity Between Hub and Spokes

Symptom: Complete isolation - no traffic flowing

Diagnostic Commands:

```
user@HUB-PE> show vpls connections instance HUB-SPOKE-VPLS
## No connections established

user@HUB-PE> show route advertising-protocol bgp 192.168.1.101 table HUB-SPOKE-VPLS.l2vpn.0
## No routes advertised
```

Cause: RT mismatch or BGP session issues

Solution:

```
## Verify RT configuration matches:
[edit routing-instances HUB-SPOKE-VPLS]
show | match "vrf-import|vrf-export"

## Ensure BGP session is up:
user@HUB-PE> show bgp summary
```

Scenario 3: Site Range Blocking Connections

Symptom: Using site ranges but some spokes can't connect

Diagnostic Commands:

```
user@HUB-PE> show vpls connections instance HUB-SPOKE-VPLS | match "Site-ID|Neighbor"
Neighbor: 192.168.1.103, Site-ID: 201  ## Out of range!

user@HUB-PE> show configuration routing-instances HUB-SPOKE-VPLS protocols vpls site-range
site-range 200;  ## Only accepts sites 1-200
```

Cause: Spoke site ID outside of hub's site range

Solution:

```
[edit routing-instances HUB-SPOKE-VPLS protocols vpls]
set site-range 300  ## Increase to cover all spokes
commit

## Or change spoke site ID:
[edit routing-instances HUB-SPOKE-VPLS protocols vpls site SPOKE3]
set site-identifier 103  ## Within range
```

Scenario 4: H-VPLS Spoke Pseudowire Down

Symptom: In H-VPLS setup, spoke can't establish PW to hub

Diagnostic Commands:

```
user@SPOKE-PE> show l2vpn connections
Instance: H-VPLS-SPOKE
  Remote-pe: 192.168.1.1 (static)
    Status: Down, Reason: No route to remote PE

user@SPOKE-PE> show route 192.168.1.1
## No route to hub PE loopback
```

Cause: Missing MPLS transport to hub

Solution:

```
## Ensure LDP/RSVP running:
[edit protocols]
set ldp interface ge-0/0/1.0
set mpls interface ge-0/0/1.0
commit

## Verify transport label exists:
user@SPOKE-PE> show route table inet.3 192.168.1.1
```

Advanced Hub-and-Spoke Features

1. Redundant Hub Sites

```
## Configure backup hub
[edit routing-instances HUB-SPOKE-VPLS]
set protocols vpls site HUB-PRIMARY site-identifier 1
```

```
set protocols vpls site HUB-PRIMARY preference primary

set protocols vpls site HUB-BACKUP site-identifier 2
set protocols vpls site HUB-BACKUP preference backup
```

2. Inter-Hub Communication

```
## Allow hubs to communicate
[edit policy-options policy-statement HUB-IMPORT-POLICY]
set term 2 from community HUB-RT
set term 2 then accept
```

3. Spoke Traffic Policing

```
## Limit spoke bandwidth at hub
[edit firewall policer SPOKE-POLICER]
set if-exceeding bandwidth-limit 100m
set if-exceeding burst-size-limit 10m
set then discard

[edit routing-instances HUB-SPOKE-VPLS protocols vpls site HUB-SITE]
set interface ge-0/0/0.100 input-policer SPOKE-POLICER
```

Best Practices for Hub-and-Spoke VPLS

1. **Choose the right method**:
   - RT-based: Most flexible, complex configuration
   - Site-range: Simpler, good for structured deployments
   - H-VPLS: Best for large-scale with simple spokes
2. **Plan site identifiers carefully**:
   - Reserve ranges for different regions
   - Document site ID assignments
   - Leave room for growth
3. **Monitor hub capacity**:
   - Hub handles all inter-spoke traffic
   - Size hub connection appropriately
   - Monitor CPU and memory usage
4. **Implement redundancy**:
   - Dual-homed hub sites
   - Multiple hub PEs
   - Fast convergence protocols
5. **Security considerations**:
   - Storm control at hub
   - MAC limiting on spoke connections
   - Consider firewall at hub site
6. **Operational simplicity**:
   - Standardize spoke configurations
   - Use templates for consistency
   - Automate provisioning where possible