Juniper Networks | Engineering Simplicity

Inline Static NAT over Layer 3 VPNs for Business Edge

Published
2023-08-27

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Inline Static NAT over Layer 3 VPNs for Business Edge

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this example to learn how to set up inline NAT so that a service provider can give enterprise employees on different networks access to cloud services from their customers' LANs through the service provider MPLS core to cloud services.

1
CHAPTER

Inline Static NAT over Layer 3 VPNs for Business Edge

Inline Static NAT over Layer 3 VPNs for Business Edge

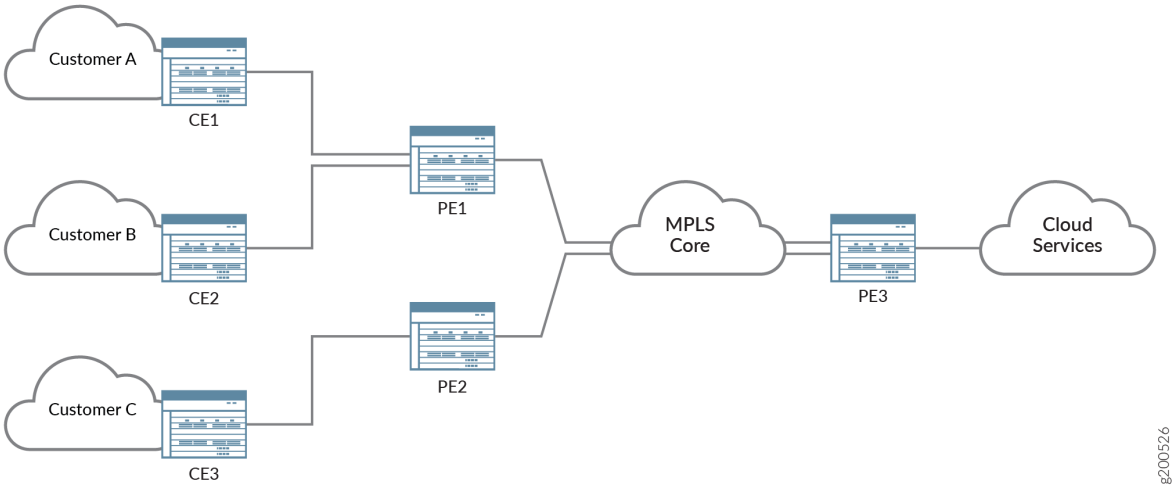
IN THIS SECTION

About This Example

This example shows how a service provider can give enterprise employees on different networks access to cloud services by using inline NAT from their customers' LANs through the service provider MPLS core to cloud services. The example consists of the following:

- Three Customer Edge (CE) routers that originate traffic from the customer LANs to cloud services.
- Three Provider Edge (PE) routers.
- Cloud Services that could belong to the enterprise or to the service provider.

Figure 1: Inline NAT Network Overview

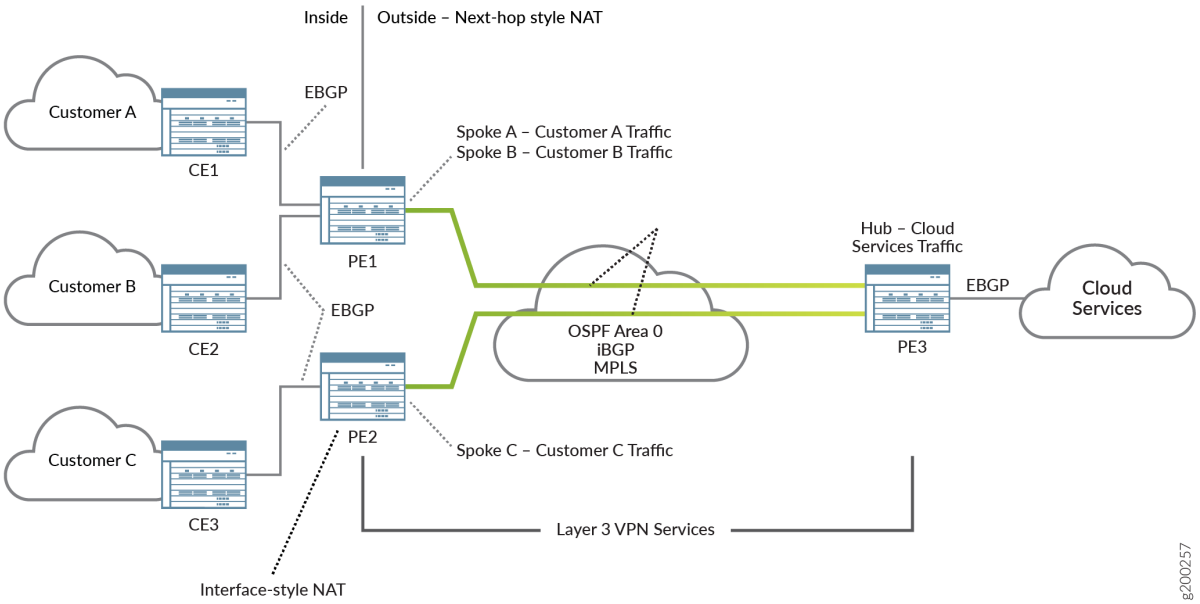


Technology Overview

IN THIS SECTION

Figure 2 on page 4 gives an overview of the technology used in this example.

Figure 2: Inline NAT Example Network Overview



Routing Overview

The core is an MPLS core that uses:

- RSVP as the signaling protocol that sets up end-to-end paths.
- Label-switched path (LSP) tunnels between the PE routers.
- EBGP to distribute routes from the CE routers and cloud services to PE routers.
- Multiprotocol BGP (MP–BGP) to exchange routing information among the PE routers.
- OSPF to provide reachability information in the core to allow BGP to resolve its next-hops.

Layer 3 VPN

A Layer 3 VPN is a set of sites that share common routing information and whose connectivity is controlled by policies. Layer 3 VPNs allow service providers to use their IP core to provide VPN services to their customers.

The type of Layer 3 VPN in this example is called BGP/MPLS VPN because BGP distributes VPN routing information across the provider's core, and MPLS forwards VPN traffic across the core to the VPN sites.

There are four sites attached to the Layer 3 VPN in this example—three customer sites and one cloud services site. The Layer 3 VPN has a hub-and-spoke configuration. Routers PE1 and PE2 are the spokes, and they connect to the customer networks. PE3 is the hub, and it connects to cloud services.

Inline NAT

In an MX Series device, you can use inline NAT on MPC line cards. You do not need a dedicated services interface, such as an MS-MPC. Inline NAT is applied in the forwarding plane, similar to the way firewalls and policers are handled in the Junos OS. Inline NAT runs on services inline (si) interfaces that are based on the FPC and PIC.

Because packets do not need to be sent to a services card for processing, the MX Series router can achieve line rate, low latency NAT translations. While Inline NAT services provide better performance than using a services card, their functionality is more basic; inline NAT supports only static NAT.

There are two types of inline NAT:

- Interface-style—an interface-based method, where packets arriving at an interface are sent through a service set. You use interface-style NAT to apply NAT to all traffic on an interface.
- Next-hop-style—a route-based method that is typically used when routing instances forward packets from a specific network or destined for a specific destination. Routing instances move customer traffic to a service interface where NAT is applied to traffic that matches the route.

Both methods are used in this example.

Requirements

This example uses the following hardware and software components:

- MX Series routers with Modular Port Concentrators (MPCs)
- Junos OS Release 17.1R1 or higher

Configuring the Core

IN THIS SECTION

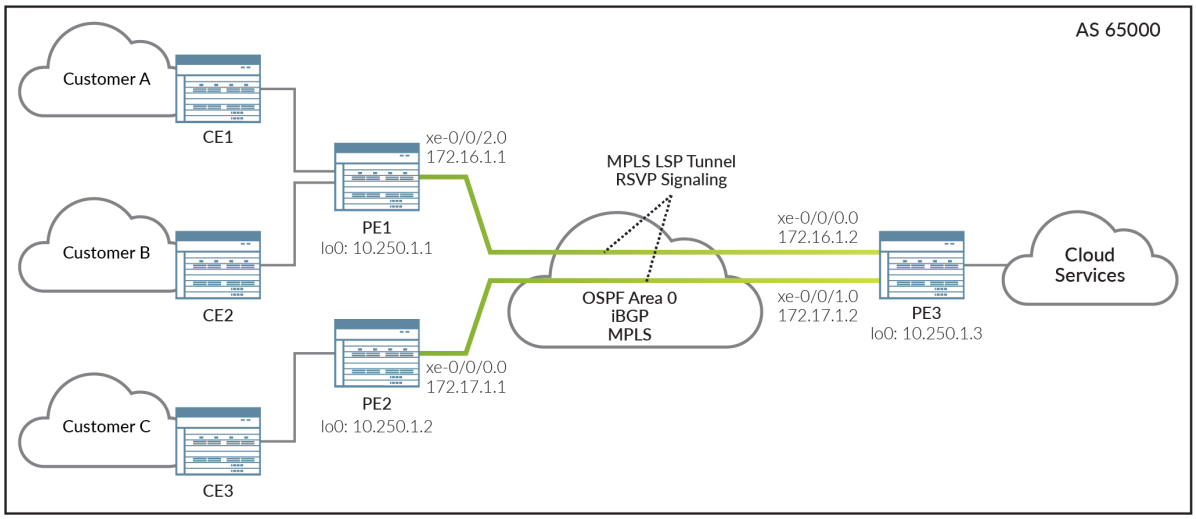
Core Overview

IN THIS SECTION

The core configuration consists of the physical and loopback interfaces and routing protocols. The routing protocol design includes:

- RSVP is the signaling protocol that sets up end-to-end paths between PE1 and PE3 and between PE2 and PE3.
- MPLS LSPs provide tunnels between PE1 and PE3 and between PE2 and PE3.
- OSPF provides reachability information in the core to allow BGP to resolve its next-hops.
- MP-BGP supports Layer 3 VPNs by allowing the PE routers to exchange information about routes originating and terminating in the VPNs.

Figure 3: • Core Interfaces and Routing



Core Transport Signaling Design Considerations

The PE devices use LSPs between them to send customer traffic over the MPLS core. In this design, we considered the two most common signaling types to set up the end-to-end LSP paths—LDP and RSVP. We are using RSVP as the signaling protocol that sets up end-to-end paths.

In this example, MP-BGP distributes VPN routing information across the provider's core, and MPLS forwards VPN traffic across the core to remote VPN sites.

Interior Gateway Protocol (IGP) Design Considerations

An IGP exchanges routing information within an autonomous system (AS). We are using OSPF as the IGP for the core network. We chose OSPF because it is easy to configure, does not require a large amount of planning, has flexible summarization and filtering, and can scale to large networks.

Configuring PE1

1. Configure the core-facing physical interface and the loopback interface.

```
interfaces {
    xe-0/0/2 {
        description "Outside to PE3";
        unit 0 {
            family inet {
                address 172.16.1.1/24;
```

```
                }
                family mpls; ## allows interface to support MPLS protocol traffic
            }
        }
        lo0 {
            unit 0 {
                family inet {
                    address 10.250.1.1/32;
                }
            }
        }
    }
```

2. Configure the core routing protocols on the core-facing interface (xe-0/0/2.0).
   - Enable RSVP.
   - Enable MPLS on the core-facing interface to allow MPLS to create an MPLS label for the interface.
   - Configure an MPLS LSP tunnel from PE1 to PE3.
   - Configure IBGP, and add PE3's loopback address as a neighbor.
   - Configure OSPF, and add the core-facing interface and the loopback interface to area 0.

   We recommend adding the `no-cspf` statement to the MPLS configuration to disable constrained-path LSP computation. CSPF is enabled by default, but it is a best practice to turn off when it is not needed.

```
protocols {
    rsvp {
        interface xe-0/0/2.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE1toPE3 {
            to 10.250.1.3; ## PE3 loopback address
        }
        interface xe-0/0/2.0; ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
```

```
            local-address 10.250.1.1;
            neighbor 10.250.1.3; ## PE3 loopback address
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/2.0;  ## core-facing interface
            interface lo0.0;
        }
    }
}
```

3. Configure the autonomous system.

```
routing-options {
    autonomous-system 65000;
}
```

4. Configure and apply per flow load balancing.

```
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet;  ## actually applied per flow
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
```

Configuring PE2

1. Configure the core-facing physical interface and the loopback interface.

```
interfaces {
    xe-0/0/0 {
        description "Outside to PE3";
        unit 0 {
            family inet {
                address 172.17.1.1/24;
```

```
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.1.2/32;
            }
        }
    }
}
```

2. Configure the core routing protocols on the core-facing interface (xe-0/0/0.0).
   - Enable RSVP.
   - Configure an MPLS LSP tunnel from PE2 to PE3.
   - Enable MPLS on the core-facing interface to allow MPLS to create an MPLS label for the interface.
   - Configure IBGP, and add PE3's loopback address as a neighbor.
   - Configure OSPF, and add the core-facing interface and the loopback interface to area 0.

   We recommend adding the `no-cspf` statement to the MPLS configuration to disable constrained-path LSP computation. CSPF is enabled by default, but it is a best practice to turn off when it is not needed.

```
protocols {
    rsvp {
        interface xe-0/0/0.0; ## core-facing interface
    }
    mpls {
        no-cspf;
        label-switched-path PE2toPE3 {
            to 10.250.1.3;  ## PE3 loopback address
        }
        interface xe-0/0/0.0 ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
```

```
            local-address 10.250.1.2; ## local loopback address
            family inet {
                unicast;
            }
            neighbor 10.250.1.3; ## lo0 address of PE3
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/0.0;
            interface lo0.0;
        }
    }
}
```

3. Configure the autonomous system.

```
routing-options {
    autonomous-system 65000;
}
```

4. Configure and apply per flow load balancing.

```
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet;  ## actually applied per flow
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
```

Configuring PE3

1. Configure the core-facing physical interfaces and the loopback interface.

```
interfaces {
    xe-0/0/0 {
        description "to PE1";
```

```
        unit 0 {
            family inet {
                address 172.16.1.2/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    xe-0/0/1 {
        description "to PE2";
        unit 0 {
            family inet {
                address 172.17.1.2/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.1.3/32;
            }
        }
    }
}
```

2. Configure the core routing protocols on the core-facing interfaces (xe-0/0/0.0 and xe-0/0/1.0).

   - Enable RSVP.
   - Enable MPLS on the core-facing interface to allow MPLS to create an MPLS label for the interface.
   - Configure an MPLS LSP tunnel from PE3 to PE1 and from PE3 to PE2.
   - Configure IBGP, and add the PE1 and PE3 loopback addresses as neighbors.
   - Configure OSPF, and add the core-facing interface and the loopback interface to area 0.

   We recommend adding the `no-cspf` statement to the MPLS configuration to disable constrained-path LSP computation. CSPF is enabled by default, but it is a best practice to turn off when it is not needed.

```
protocols {
    rsvp {
```

```
        interface xe-0/0/0.0;
        interface xe-0/0/1.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE3toPE1 {
            to 10.250.1.1;  ## PE1 loopback address
        }
        label-switched-path PE3toPE2 {
            to 10.250.1.2;  ## PE2 loopback address
        }
        interface xe-0/0/0.0; ## core-facing interface
        interface xe-0/0/1.0; ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
            local-address 10.250.1.3; ## local loopback address
            family inet {
                unicast;
            }
            neighbor 10.250.1.1; ## lo0 address of spoke router PE1
            neighbor 10.250.1.2; ## lo0 address of spoke router PE2
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/0.0;
            interface xe-0/0/1.0;
            interface lo0.0;
        }
    }
}
```

3. Configure the autonomous system.

```
routing-options {
    autonomous-system 65000;
}
```

4. Configure and apply per flow load balancing.

```
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet;  ## actually applied per session
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
```

Verifying Your Configuration

Commit and then verify the core configuration. The examples below show output from PE3.

1. Verify that your physical interfaces are up.

```
host@PE3> show interfaces xe-0/0/0 terse
Interface               Admin Link Proto    Local                 Remote
xe-0/0/0                up    up
xe-0/0/0.0              up    up   inet     172.16.1.2/24
                                  mpls
                                  multiservice
```

```
host@PE3> show interfaces xe-0/0/1 terse
Interface               Admin Link Proto    Local                 Remote
xe-0/0/1                up    up
xe-0/0/1.0              up    up   inet     172.17.1.2/24
                                  mpls
                                  multiservice
```

2. Verify OSPF neighbors.

```
host@PE3> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
```

```
172.16.1.1       xe-0/0/0.0            Full      10.250.1.1       128    39
172.17.1.1       xe-0/0/1.0            Full      10.250.1.2       128    34
```

3. Verify BGP peers on PE1 and PE2.

```
user@PE3> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                      0          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active
Received/Accepted/Damped...
10.250.1.1            65000       2586       2575       0       0    19:21:36
0/0/0/0              0/0/0/0
10.250.1.2            65000       2586       2577       0       0    19:21:38
0/0/0/0              0/0/0/0
```

4. Show that neighbors are established in the IBGP group.

```
user@PE3> show bgp group IBGP
Group Type: Internal    AS: 65000                 Local AS: 65000
  Name: IBGP            Index: 0                  Flags: <Export Eval>
  Holdtime: 0
  Total peers: 2        Established: 2
  10.250.1.1+179
  10.250.1.2+179
  inet.0: 0/0/0/0
```

5. Verify your RSVP sessions.

```
host@PE3> show rsvp session
Ingress RSVP: 2 sessions
To              From            State   Rt Style Labelin Labelout LSPname
10.250.1.1      10.250.1.3      Up       0  1 FF       -        3 PE3toPE1
10.250.1.2      10.250.1.3      Up       0  1 FF       -        3 PE3toPE2
Total 2 displayed, Up 2, Down 0

Egress RSVP: 2 sessions
To              From            State   Rt Style Labelin Labelout LSPname
10.250.1.3      10.250.1.2      Up       0  1 FF       3        - PE2toPE3
10.250.1.3      10.250.1.1      Up       0  1 FF       3        - PE1toPE3
```

```
Total 2 displayed, Up 2, Down 0

Transit RSVP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

6. Verify your MPLS LSP sessions.

```
host@PE3> show mpls lsp
Ingress LSP: 2 sessions
To              From            State Rt P     ActivePath       LSPname
10.250.1.1      10.250.1.3      Up     0 *                      PE3toPE1
10.250.1.2      10.250.1.3      Up     0 *                      PE3toPE2
Total 2 displayed, Up 2, Down 0

Egress LSP: 2 sessions
To              From            State   Rt Style Labelin Labelout LSPname
10.250.1.3      10.250.1.2      Up       0  1 FF       3        - PE2toPE3
10.250.1.3      10.250.1.1      Up       0  1 FF       3        - PE1toPE3
Total 2 displayed, Up 2, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

7. Check the status of MPLS label-switched paths (LSPs).

```
host@PE3> show mpls lsp name PE3toPE1 detail
Ingress LSP: 2 sessions

10.250.1.1
  From: 10.250.1.3, State: Up, ActiveRoute: 0, LSPname: PE3toPE1
  ActivePath:  (primary)
  LSPtype: Static Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  LSP Self-ping Status : Enabled
 *Primary                    State: Up
    Priorities: 7 0
    SmartOptimizeTimer: 180
    Flap Count: 0
    MBB Count: 0
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt 20=Node-ID):
          172.16.1.1(Label=3)
```

```
Total 1 displayed, Up 1, Down 0

Egress LSP: 2 sessions
Total 0 displayed, Up 0, Down 0

Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

8. To validate OSPF-level reachability in the core, from PE3, ping PE1.

```
user@PE3> ping 10.250.1.1
PING 10.250.1.1 (10.250.1.1): 56 data bytes
64 bytes from 10.250.1.1: icmp_seq=0 ttl=64 time=1.571 ms
64 bytes from 10.250.1.1: icmp_seq=1 ttl=64 time=1.829 ms
64 bytes from 10.250.1.1: icmp_seq=2 ttl=64 time=0.947 ms
64 bytes from 10.250.1.1: icmp_seq=3 ttl=64 time=2.022 ms
64 bytes from 10.250.1.1: icmp_seq=4 ttl=64 time=0.948 ms
--- 10.250.1.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.947/1.463/2.022/0.445 ms
```

Configuring the Layer 3 VPN on PE Routers

IN THIS SECTION

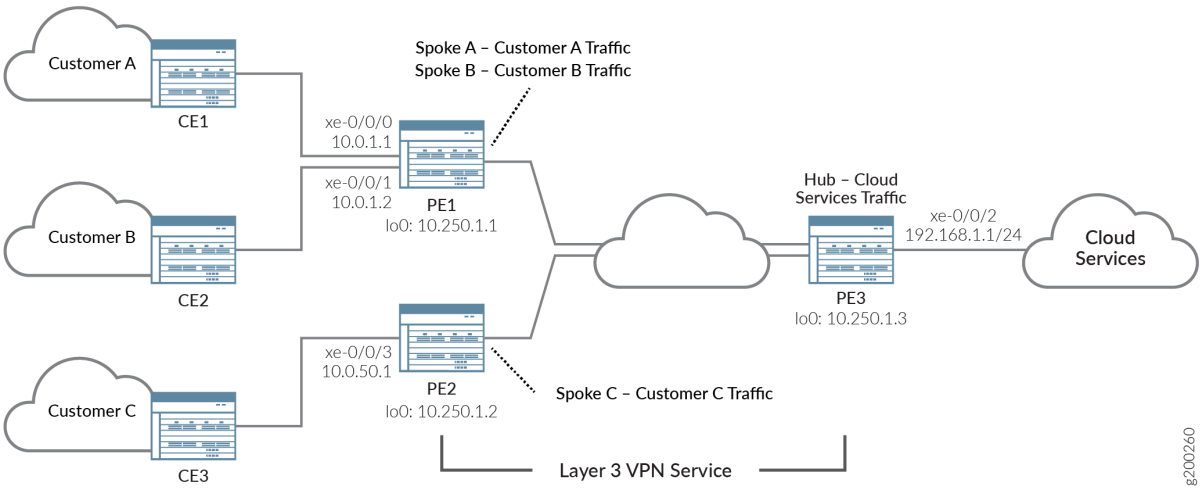
Layer 3 VPN Overview

We are using a Layer 3 VPN to separate and route traffic from each of the customer LANs and cloud services over the core. There are four sites in the VPN—the three customer LANs and cloud services.

To distinguish routes for the customer LANs and cloud services on the PE routers, we are using virtual routing and forwarding (VRF) routing instance . A VRF routing instance has one or more routing tables, a forwarding table, the interfaces that use the forwarding table, and the policies and routing protocols that control what goes into the forwarding table. The VRF tables are populated with routes received from the CE sites and cloud services, and with routes received from other PE routers in the VPN. Because each site has its own routing instance, each site has separate tables, rules, and policies.

This example uses a hub-and-spoke VPN configuration. Routers PE1 and PE2 are the spokes, and they represent the customer networks. PE3 is the hub, and it represents the cloud services. Policies mark traffic as either a hub or spoke, and the marking is used to direct traffic to the correct VRF routing instance.

Figure 4: Layer 3 VPN with Hub and Spokes



Configuring PE1

1. Configure the physical interfaces to Customer A and Customer B.

```
interfaces {
    xe-0/0/0 {
        description "Inside to CE1_Cust-A";
        unit 0 {
```

```
            family inet {
                address 10.0.1.1/24;
            }
        }
    }
    xe-0/0/1 {
        description "Inside to CE1_Cust-B";
        unit 0 {
            family inet {
                address 10.0.1.2/24;
            }
        }
    }
}
```

2. Configure policies that we will use as VPN import and export policies in the router's VRF routing instances.

   - `CustA-to-CloudSvcs` and `CustB-to-CloudSvcs`—These are export policies that add the Spoke tag when BGP exports routes that match the policies.
   - `from-CloudSvcs`—This is an import policy that adds received routes with the Hub tag to the VRF routing table.

```
policy-options {
    policy-statement CustA-to-CloudSvcs {
        term 1 {
            from protocol static;
            then {
                community add SpokeA; ## Add SpokeA tag when BGP exports route
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement CustB-to-CloudSvcs {
        term 1 {
            from protocol static;
            then {
                community add SpokeB; ## Add SpokeB tag when BGP exports route
                accept;
```

```
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement from-CloudSvcs {  ## add routes received with Hub tag to routing table
    term 1 {
        from community Hub;
        then accept;
    }
    term 2 {
        then reject;
    }
}
```

3. Configure VRF routing instances for Customer A and B. These routing instances create the following routing tables on PE1:

   - For the Customer A, the VRF table is Cust-A-VRF.inet.0.
   - For the customer B, the VRF table is Cust-B-VRF.inet.0.

   Each routing instance must contain:

   - Instance type of VRF, which creates the VRF routing table for the VPN on the PE router.
   - Interface connected to the customer CE device.
   - Route distinguisher, which must be unique for each routing instance on the PE router. It is used to distinguish the addresses in one VPN from those in another VPN.
   - VRF import policy that adds received routes with the Hub tag to the VRF routing table.
   - VRF export policy that adds the Spoke tag when BGP exports the route.
   - VRF table label that maps the inner label of a packet to a specific VRF table. This allows the examination of the encapsulated IP header. All routes in the VRF configured with this option are advertised with the label allocated per VRF.

```
routing-instances {
    Cust-A-VRF {
        instance-type vrf;
        interface xe-0/0/0.0;
        route-distinguisher 10.250.1.1:1;
        vrf-import from-CloudSvcs;
```

```
        vrf-export CustA-to-CloudSvcs;
        vrf-table-label;
    }
    Cust-B-VRF {
        instance-type vrf;
        interface xe-0/0/1.0;
        route-distinguisher 10.250.1.1:2;
        vrf-import from-CloudSvcs;
        vrf-export CustB-to-CloudSvcs;
        vrf-table-label;
    }
}
```

4. Add Layer 3 VPN support to the IBGP group.

```
protocols  {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast;
        }
    }
}
```

Configuring PE2

1. Configure the interface to Customer C.

```
interfaces {
    xe-0/0/3 {
        description "Inside to CE1_Cust-C";
        unit 0 {
            family inet {
                address 10.0.50.1/24;
            }
        }
    }
}
```

2. Configure policies that we will use as VPN import and export policies in the router's VRF routing instance.

- `CustC-to-CloudSvcs`—This is an export policy that adds the Spoke tag when BGP exports routes that match the policy.
- `from-CloudSvcs`—This is an import policy that adds received routes with the Hub tag to the VRF routing table.

```
policy-options {
    policy-statement CustC-to-CloudSvcs { ## Add SpokeC tag when BGP exports route
    term 1 {
        from protocol static;
        then {
            community add SpokeC;
            accept;
        }
    }
    term 2 {
        then reject;
    }
}
policy-statement from-CloudSvcs { ## add routes received with Hub tag to routing table
    term 1 {
        from community Hub;
        then accept;
    }
    term 2 {
        then reject;
    }
}
```

3. Configure a VRF routing instance for Customer C that will create a routing table to forward packets within the VPN.

   For Cust-C, the VRF table is Cust-C.inet.0.

   The routing instance must contain:

   - Route distinguisher, which must be unique for each routing instance on the PE router. It is used to distinguish the addresses in one VPN from those in another VPN.
   - Instance type of VRF, which creates the VRF routing table for the VPN on the PE router.
   - Interface connected to CE3.
   - VRF import policy that adds received routes with the Hub tag to the VRF routing table.

- VRF export policy that adds the Spoke tag when BGP exports the route.
- VRF table label that maps the inner label of a packet to a specific VRF table. This allows the examination of the encapsulated IP header. All routes in the VRF configured with this option are advertised with the label allocated per VRF.

```
routing-instances {
    Cust-C {
        instance-type vrf;
        interface xe-0/0/3.0;
        route-distinguisher 10.250.1.2:3;
        vrf-import from-CloudSvcs;
        vrf-export CustC-to-CloudSvcs;
        vrf-table-label;
    }
}
```

4. Add Layer 3 VPN support to the IBGP group.

```
protocols  {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast
        }
    }
}
```

Configuring PE3

1. Configure the physical interface to cloud services.

```
interfaces {
    xe-0/0/2 {
        description "PE3 to CloudSvcs";
        unit 0 {
            family inet {
                address 192.168.1.1/24;
            }
        }
```

```
    }
}
```

2. Configure policies that we will use as VPN import and export policies in the router's VRF routing instance.

   - `to-Cust`—This is an export policy that adds the Spoke tag when BGP exports routes that match the policy.
   - `from-Cust`—This is an import policy that adds received routes that have a Spoke tag to the VRF routing table.

```
policy-options {
    policy-statement from-Cust {  ## add routes with hub tag to routing table
    term 1 {
        from community [ SpokeA SpokeB SpokeC ];
        then accept;
    }
}
policy-statement to-Cust {  ## add hub tag when BGP exports route
    term 1 {
        from protocol bgp;
        then {
            community add Hub;
            accept;
        }
    }
}
```

3. Configure a VRF routing instance that is used to create a routing table to forward packets within the VPN.

   For Cloud Services, the VRF table is CloudSvcs.inet.0.

   The routing instance must contain:

   - Route distinguisher, which must be unique for each routing instance on the PE router. It is used to distinguish the addresses in one VPN from those in another VPN.
   - Instance type of VRF, which creates the VRF table on the PE router.
   - Interfaces connected to the PE routers.
   - VRF import policy that adds received routes with a Spoke tag to the VRF routing table.

- VRF export policy that adds the Spoke tag when BGP exports routes that match the policy.

```
routing-instances {
    CloudSvcs {
        instance-type vrf;
        interface xe-0/0/2.0;
        route-distinguisher 10.250.1.3:100; ## PE3
        vrf-import from-Cust;
        vrf-export to-Cust;
    }
}
```

4. Add Layer 3 VPN support to the IBGP group that was configured previously.

```
protocols  {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast
        }
    }
}
```

Verifying Your Configuration

To verify your configuration, commit the configuration, and then do the following:

1. From PE3, show neighbors in the IBGP group. Notice the addition of the bgp.l3vpn and the CloudSvcs routing tables.

```
user@PE3> show bgp group IBGP
Group Type: Internal    AS: 65000                 Local AS: 65000
  Name: IBGP            Index: 0                  Flags: <Export Eval>
  Holdtime: 0
  Total peers: 2        Established: 2
  10.250.1.1+179
  10.250.1.2+179
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 0/0/0/0
  CloudSvcs.inet.0: 0/0/0/0
```

NOTE: When a PE router receives a route from another PE router, it checks it against the import policy on the IBGP session between the PE routers. If it is accepted, the router places the route into its bgp.l3vpn.0 table. At the same time, the router checks the route against the VRF import policy for the VPN. If it matches, the route distinguisher is removed from the route and the route is placed into the appropriate VRF table (the *routing-instance-name*.inet.0 table).

2. From PE3, verify BGP peers on PE1 and PE2. Again notice the addition of the bgp.l3vpn tables.

```
user@PE3> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       0          0          0          0          0          0
bgp.l3vpn.0
                       0          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.250.1.1            65000         41         39       0       0       17:16 Establ
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 0/0/0/0
10.250.1.2            65000         41         38       0       0       17:12 Establ
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 0/0/0/0
```

3. From PE1, verify that the Cust-A-VRF routing instance is active.

```
user@PE1> show route instance Cust-A-VRF detail
Cust-A-VRF:
  Router ID: 10.0.1.1
  Type: vrf               State: Active
  Interfaces:
    xe-0/0/0.0
    lsi.0
  Route-distinguisher: 10.250.1.1:1
  Vrf-import: [ from-CloudSvcs ]
  Vrf-export: [ CustA-to-CloudSvcs ]
  Fast-reroute-priority: low
  Tables:
    Cust-A-VRF.inet.0      : 3 routes (3 active, 0 holddown, 0 hidden)
```

4. From PE1, verify the Cust-A-VRF.inet.0 routing table.

```
user@PE1> show route table Cust-A-VRF.inet.0
Cust-A-VRF.inet.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.1.0/24        *[Direct/0] 00:10:12
                    > via xe-0/0/0.0
10.0.1.1/32        *[Local/0] 00:10:12
                      Local via xe-0/0/0.0
```

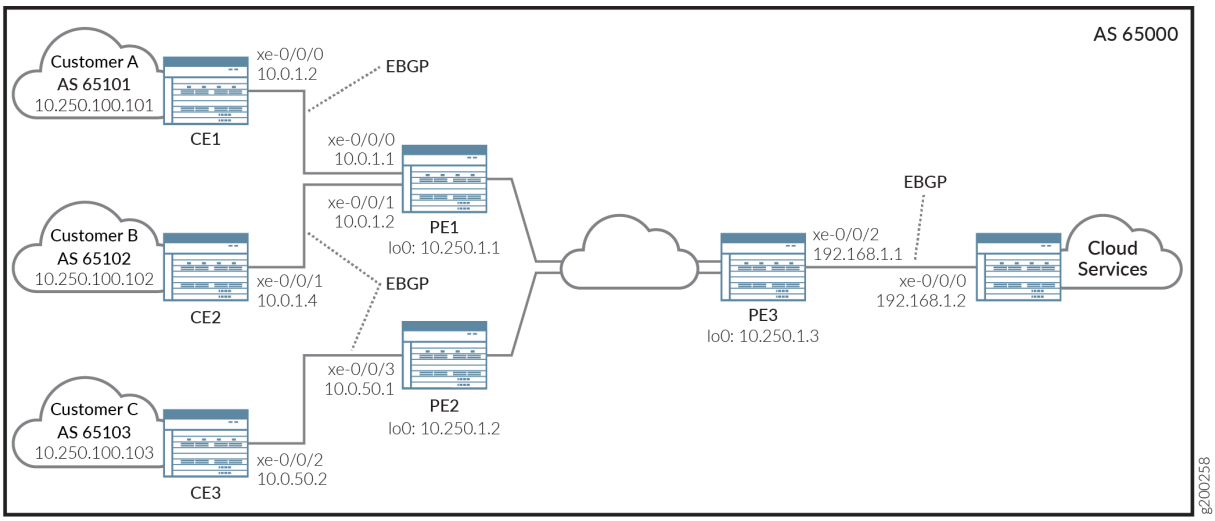
Configuring Connections from CE Routers and Cloud Services to PE Routers

IN THIS SECTION

Connections from CE Routers and Cloud Services to PE Routers Overview

We are using EBGP for routing between the CE routers and PE1 and PE2 and between cloud services and PE3. The CE routers use a routing policy that matches the address of the customer LAN. You apply this policy as an export policy in the EBGP peer, which causes EBGP to send these addresses to PE routers. The same configuration on the cloud services router causes its routes to be sent to PE3.

Figure 5: Connections to CE Routers and Cloud Services



Configuring the Connection Between CE1 and PE1

IN THIS SECTION

In this example we are using the loopback interfaces on the router to represent the customer LANs. That is why the loopback interfaces use the IP addresses of the customer LAN.

Configuring CE1

1. Configure the physical and loopback interfaces for CE1.

```
interfaces {
    xe-0/0/0 {
        description Cust-A_to_PE1;
        unit 0 {
            family inet {
                address 10.0.1.2/24;
            }
```

```
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.101/32;
            }
        }
    }
}
```

2. Configure a routing policy that matches the address of the Customer A LAN.

```
policy-options {
    policy-statement CustA-to-PE1 {
        term 1 {
            from {
                route-filter 10.250.100.101/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

3. Configure an EBGP group for peering between CE1 and PE1. Apply the routing policy that matches the Customer A LAN as an export policy. BGP advertises the address in the policy to PE1, which redistributes the customer LAN routes into the VPN.

```
protocols {
    bgp {
        group to-PE1 {
            type external;
            export CustA-to-PE1; ## BGP advertises routes to the L3VPN
            peer-as 65000;
            neighbor 10.0.1.1;  ## PE1 interface address
        }
    }
}
```

4. Configure the autonomous system for the router.

```
routing-options {
    autonomous-system 65101;
}
```

Configuring PE1

Add an EBGP group to the Cust-A-VRF routing instances for peering between PE1 and CE1.

```
routing-instances  {
    Cust-A-VRF {
        protocols {
            bgp {
                group to-Cust-A {
                    type external;
                    peer-as 65101;
                    neighbor 10.0.1.2;  ## CE1 interface address
                }
            }
        }
    }
}
```

Configuring the Connection Between CE2 and PE1

IN THIS SECTION

In this example we are using the loopback interfaces on the router to represent the customer LANs. That is why the loopback interfaces use the IP addresses of the customer LAN.

Configuring CE2

1. Configure the physical and loopback interfaces for CE2.

```
interfaces {
    xe-0/0/1 {
        description Cust-B_to_PE1;
        unit 0 {
            family inet {
                address 10.0.1.4/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.102/32;
            }
        }
    }
}
```

2. Configure a routing policy that matches the address of the Customer B LAN.

```
policy-options {
    policy-statement CustB-to-PE1 {
        term 1 {
            from {
                route-filter 10.250.100.102/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

3. Configure an EBGP group for peering between CE2 and PE1. Apply the routing policy that matches the Customer B LAN as an export policy. BGP advertises this address to the VPN network, which means that the customer LAN routes are distributed into the VPN.

```
protocols {
    bgp {
        group to-PE1 {
            type external;
            export CustB-to-PE1;
            peer-as 65000;
            neighbor 10.0.1.2;  ## PE1 interface address
        }
    }
}
```

4. Configure the autonomous system.

```
routing-options {
    autonomous-system 65102;
}
```

Configuring PE1

Add an EBGP group to the Cust-B-VRF routing instances for peering between PE1 and CE2.

```
routing-instances  {
    Cust-B-VRF {
        protocols {
            bgp {
                group to-Cust-B {
                    type external;
                    peer-as 65102
                    neighbor 10.0.1.4;  ## CE2 interface address
                }
            }
        }
    }
}
```

Configuring the Connection Between CE3 and PE2

IN THIS SECTION

In this example we are using the loopback interfaces on the router to represent the customer LANs. That is why the loopback interfaces use the IP addresses of the customer LAN.

Configuring CE3

1. Configure the physical and loopback interfaces for CE3.

```
interfaces {
    xe-0/0/2 {
        description Cust-C_to_PE2;
        unit 0 {
            family inet {
                address 10.0.50.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.103/32;
            }
        }
    }
}
```

2. Configure a routing policy that matches the address of the Customer C LAN.

```
policy-options {
    policy-statement CustC-to-PE2 {
        term 1 {
            from {
```

```
                route-filter 10.250.100.103/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
```

3. Configure an EBGP group for peering between CE3 and PE2. Apply the routing policy that matches the Customer C LAN as an export policy. BGP advertises this address to the VPN network, which means that the customer LAN routes are distributed into the VPN.

```
protocols {
    bgp {
        group to-PE2 {
            type external;
            export CustC-to-PE2;
            peer-as 65000;
            neighbor 10.0.50.1;  ## PE2 interface address
        }
    }
}
```

4. Configure the autonomous system.

```
routing-options {
    autonomous-system 65103;
}
```

Configuring PE2

Add an EBGP group to the Cust-C routing instance for peering between PE2 and CE3.

```
routing-instances {
    Cust-C {
        protocols {
            bgp {
                group to-Cust-C {
```

```
                    type external;
                    peer-as 65103
                    neighbor 10.0.50.2;  ## CE3 interface address
                }
            }
        }
    }
}
```

Verifying Connections from CE Routers and Cloud Services

1. Verify that the CE routers' physical interfaces are up. For example:

```
host@CE1> show interfaces xe-0/0/0 terse
Interface               Admin Link Proto    Local                 Remote
xe-0/0/0                up    up
xe-0/0/0.0              up    up   inet     10.0.1.2/24
                                  multiservice
```

2. Verify connections from PE routers to CE routers. For example:

```
host@PE1> ping 10.0.1.2 routing-instance Cust-A-VRF source 10.0.1.1 count 4
PING 10.0.1.2 (10.0.1.2): 56 data bytes
64 bytes from 10.0.1.2: icmp_seq=0 ttl=64 time=1.087 ms
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=1.434 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=3.478 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=15.761 ms

--- 10.0.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.087/5.440/15.761/6.028 ms
```

3. Verify connection from PE3 to cloud services.

```
host@PE3> ping 192.168.1.2 routing-instance CloudSvcs source 192.168.1.1 count 4
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.936 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=6.257 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.868 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.849 ms
```

```
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.849/2.228/6.257/2.327 ms
```

4. On PE3, verify BGP peers. Cloud services (192.168.1.2) is now a BGP peer.

```
host@PE3> show bgp summary
Groups: 2 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       0          0          0          0          0          0
bgp.l3vpn.0
                       0          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.250.1.1            65000        946        941       0       0     7:03:46 Establ
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 0/0/0/0
10.250.1.2            65000        946        941       0       0     7:03:42 Establ
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 0/0/0/0
192.168.1.2           65200        159        155       0       0     1:09:47 Establ
  CloudSvcs.inet.0: 1/1/1/0
```

5. On PE1, verify that CE1 and CE2 are now BGP peers.

```
host@PE1> show bgp summary
Groups: 3 Peers: 3 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       0          0          0          0          0          0
bgp.l3vpn.0
                       1          1          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.0.1.2              65101          11          3       0       0    16:41:31 Establ
10.0.1.4              65102           7          2       0       0        1:24 Establ
  Cust-B-VRF.inet.0: 1/1/1/0
10.250.1.3            65000       3202       3210       0       1  1d 0:02:40 Establ
  inet.0: 0/0/0/0
  bgp.l3vpn.0: 1/1/1/0
```

```
  Cust-A-VRF.inet.0: 1/1/1/0
  Cust-B-VRF.inet.0: 1/1/1/0
```

6. On the CE routers, verify the EBGP group. For example:

```
host@CE1> show bgp group to-PE1
Group Type: External                              Local AS: 65101
  Name: to-PE1          Index: 0                  Flags: <Export Eval>
  Export: [ CustA-to-PE1 ]
  Holdtime: 0
  Total peers: 1        Established: 1
  10.0.1.1+60358
  inet.0: 1/1/1/0
```

Configuring Inline NAT

IN THIS SECTION

Inline NAT Design Considerations

IN THIS SECTION

Inline NAT provides stateless address translation on MX Series routers that have MPC line cards. The benefit of using an inline service is that you do not need a dedicated services card and there is almost no impact to forwarding capacity or latency. While inline services generally provide better performance than using a services card, their functionality tends to be more basic. For example, inline NAT supports only static NAT.

We are using source static NAT in this inline NAT example.

Types of Inline NAT

There are two types of inline NAT:

- Interface-style—an interface-based method, where packets arriving at an interface are sent through a service set. Use interface-style NAT to apply NAT to all traffic that traverses an interface.

  Interface-style NAT is simpler to configure than next-hop style NAT.

- Next-hop-style—a route-based method that is typically used when routing instances forward packets sourced from a specific network or destined to a specific destination through the inline service.

This example shows how to use both methods of inline NAT as follows:

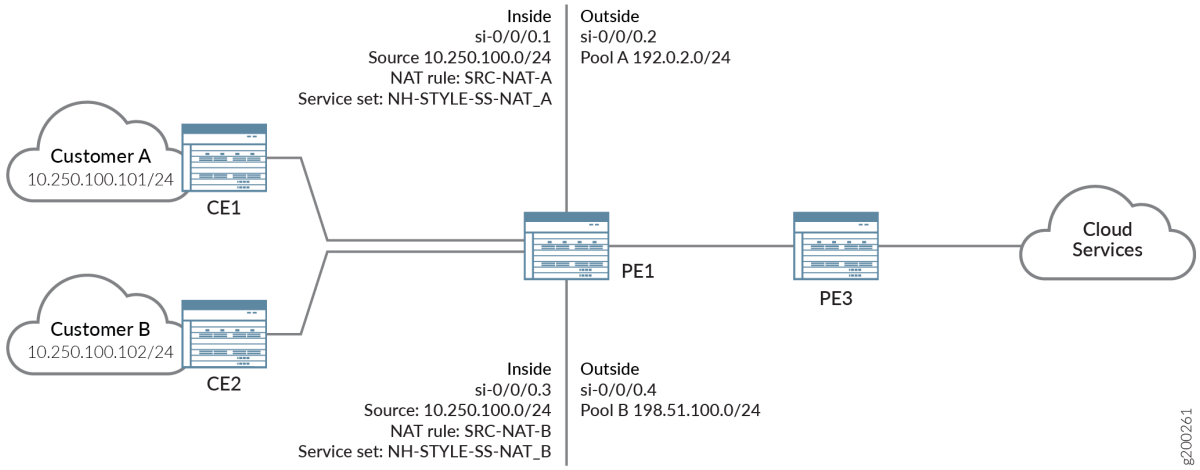
- PE1 uses next-hop based inline NAT for traffic from the Customer A and Customer B networks to cloud services.
- PE 2 uses interface-based inline NAT for traffic from the Customer C network to cloud services.

Configuring Next-Hop Style Inline Source NAT on PE1

This section shows how to configure route-based inline NAT using si- interfaces with next-hop style service-sets.

In this example, the Customer A LAN and Customer B LAN have overlapping subnets. The PE1 router differentiates the traffic according to which si- interface the traffic arrives on.

Figure 6: Next-Hop Style Inline NAT Configuration

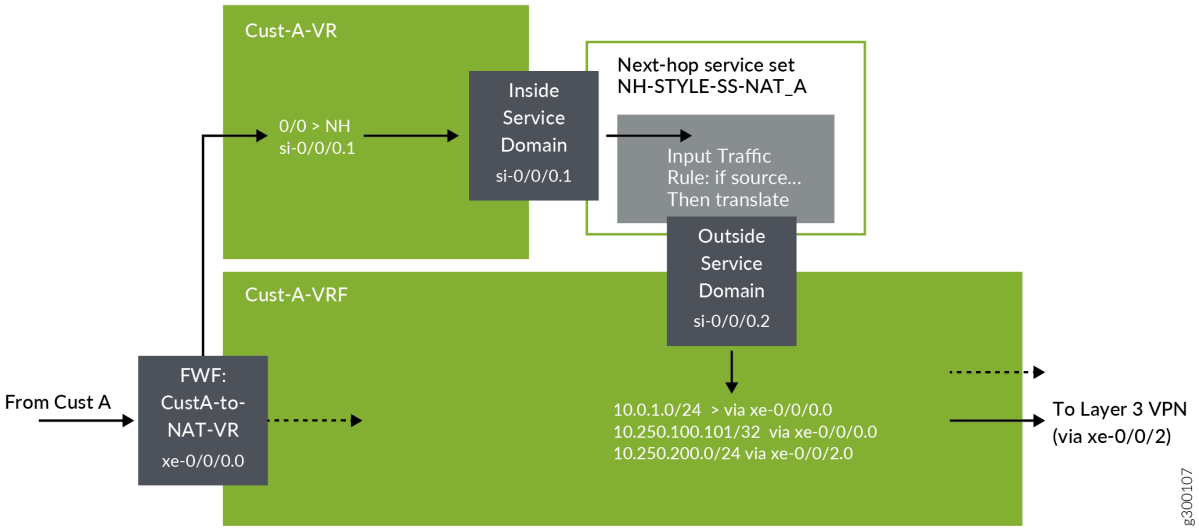


The following configuration items are used in this section:

- Inline service interface—a virtual interface that resides on the Packet Forwarding Engine of the MPC. To access services, traffic flows in and out of the si- (service-inline) interfaces.
- Service set—defines the service(s) performed, and identifies which inline interface will feed traffic into and out of the service set. This section implements next-hop style service sets, which uses a route-based method, where static routes are used to forward packets destined for a specific destination through the inline service.
- NAT rule—uses an if-then structure (like firewall filters) to define match conditions and then apply address translation to the matching traffic.
- NAT pool—a user-defined set of IP addresses that the NAT rule uses for translation.
- Virtual router (VR) routing instance—includes a default static route that sends customer traffic to the si- interface where NAT is applied.
- Firewall filters— redirects inbound traffic from Customer A and Customer B to the appropriate VR routing instances.
- VRF routing instance—the outside si- interfaces are added to the VRF routing instances for Customer A and Customer B so NAT-translated outbound traffic can continue on its intended path across the VPN.

Figure 7 on page 40 shows the traffic flow through PE1 for inline NAT traffic coming from the Customer A LAN and going to cloud services.

Figure 7: Traffic Flow on PE1 for Next-Hop Style Inline NAT Traffic from Customer A LAN to Cloud Services



To configure next-hop style inline NAT on PE1:

1. Enable inline services for the relevant FPC slot and PIC slot, and define the amount of bandwidth to dedicate for inline services.

   The FPC and PIC settings map to the si- interface configured.

```
chassis {
    fpc 0 {
        pic 0 {
            inline-services {
                bandwidth 1g;
            }
        }
    }
}
```

2. Configure the service interfaces used for NAT. The inside interfaces are on the CE side of the network. The outside interfaces are on the core side of the network.

   - For traffic from the customer network to the core:

     The inside interface handles ingress traffic from the customer network. NAT is applied to this traffic, and then the egress traffic is sent out the outside interface.

   - For traffic from the core to the customer network:

The outside interface handles ingress traffic from the core network. NAT is removed from this traffic, and then the egress traffic is sent out the inside interface.

```
interfaces {
    si-0/0/0 {
        unit 1 {  ## Customer A traffic
        family inet;  ## causes NAT to be applied to IPv4 traffic
        service-domain inside;  ## traffic from customer A LAN to core
    }
    unit 2 {  ## Customer A traffic
    family inet;
    service-domain outside;  ## customer A traffic from core to customer LAN
}
unit 3 {  ## Customer B traffic
    family inet;
    service-domain inside;  ## traffic from customer B LAN to core
}
unit 4 {  ## Customer B traffic
    family inet;
    service-domain outside;  ## customer B traffic from core to customer LAN
}
```

3. Configure NAT pools.

```
services {
    nat {
        pool A {  ## applied to customer A traffic
        address 192.0.2.0/24;
    }
    pool B {  ## applied to customer B traffic
    address 198.51.100.0/24;
}
```

4. Configure NAT rules that:
   - Match traffic from the Customer A and Customer B networks.
   - Apply the pool from which to obtain an address.

- Apply basic NAT 44, a type of static NAT that applies to IPv4 traffic.

```
services {
    nat {
        rule SRC-NAT-A {  ## applied to traffic from customer A
        match-direction input;  ## direction from which traffic is received
        term r1 {
            from {
                source-address {
                    10.250.100.0/24; ## from customer A
                }
            }
            then {
                translated {
                    source-pool A;
                    translation-type {
                        basic-nat44;  ## type of one-to-one static NAT
                    }
                }
            }
        }
    }
    rule SRC-NAT-B {  ## applied to traffic from customer B
    match-direction input;
    term r1 {
        from {
            source-address {
                10.250.100.0/24; ## from customer B
            }
        }
        then {
            translated {
                source-pool B;
                translation-type {
                    basic-nat44; ## type of one-to-one static NAT
                }
            }
        }
    }
}
```

5. Configure next-hop style service sets. The service sets associate service interfaces and define services. In this case, traffic sent through the defined inside and outside interfaces will have NAT processing applied.

```
services {
    service-set NH-STYLE-SS-NAT_A {  ## applied to Customer A traffic
    nat-rules SRC-NAT-A;
    next-hop-service {
        inside-service-interface si-0/0/0.1;
        outside-service-interface si-0/0/0.2;
    }
}
service-set NH-STYLE-SS-NAT_B {  ## applied to Customer B traffic
    nat-rules SRC-NAT-B;
    next-hop-service {
        inside-service-interface si-0/0/0.3;
        outside-service-interface si-0/0/0.4;
    }
}
```

6. Create VR routing instances for Customer A traffic and Customer B traffic. These routing instances separate Customer A and B traffic. They include default static routes that send traffic to the inside service interfaces and towards the service sets, where NAT can be applied.

```
routing-instances {
    Cust-A-VR {
        instance-type virtual-router;
        interface si-0/0/0.1; ## Cust A inside service interface
        routing-options {
            static {
                route 0.0.0.0/0 next-hop si-0/0/0.1; ## incoming Cust A traffic to si
            }
        }
    }
    Cust-B-VR {
        instance-type virtual-router;
        interface si-0/0/0.3; ## Cust B inside service interface
        routing-options {
            static {
                route 0.0.0.0/0 next-hop si-0/0/0.3; ## incoming Cust B traffic to si

            }
```

```
        }
    }
}
```

7. Configure firewall filters that redirect incoming traffic from Customer A and Customer B to the VR routing instances.

```
firewall {
    filter CustA-to-NAT-VR {
        term 1 {
            from {
                source-address {  ## traffic from Customer A network
                10.250.100.0/24;
            }
        }
        then {
            routing-instance Cust-A-VR; ## sends matching traffic to this RI
        }
    }
    term 2 {
        then accept;
    }
}
filter CustB-to-NAT-VR {
    term 1 {
        from {
            source-address {  ## traffic from Customer B network
            10.250.100.0/24;
        }
    }
    then {
        routing-instance Cust-B-VR; ## sends matching traffic to this RI
    }
}
term 2 {
    then accept;
}
```

8. Apply the firewall filters to the appropriate interfaces.

```
interfaces {
    xe-0/0/0 {  ## to Customer A
```

```
    unit 0 {
        family inet {
            filter {
                input CustA-to-NAT-VR;
            }
        }
    }
}
```

9. Add the outside si- interfaces to the previously configured VRF routing instances. This puts the now-NAT-translated outbound traffic back in its intended VRF routing instance, and it can now be sent across the VPN as usual.

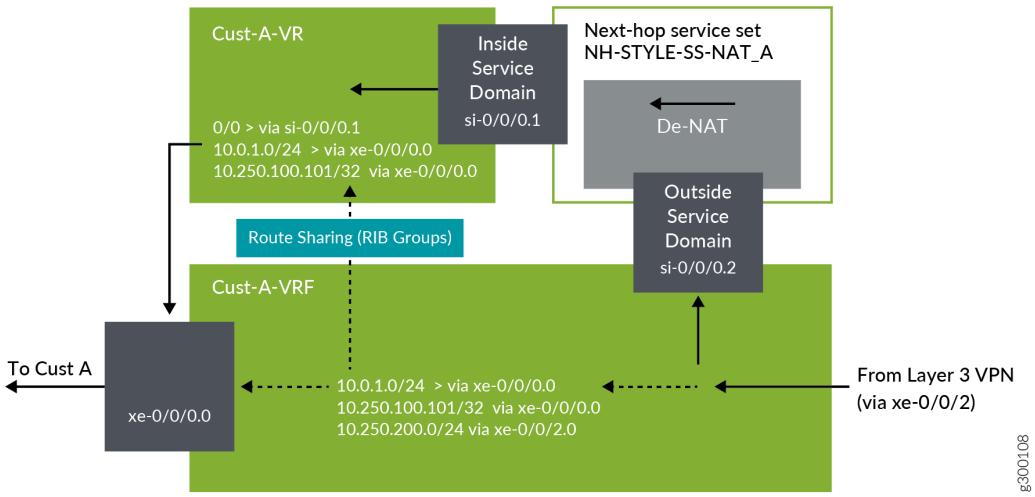
```
routing-instances {
    Cust-A-VRF {
        interface si-0/0/0.2;
    }
    Cust-B-VRF {
        interface si-0/0/0.4;
    }
}
```

Allowing Return Traffic from Cloud Services to Reach Customer LANs

When return traffic from Cloud Services goes through the services interfaces, NAT addressing is removed, and traffic is sent to the VR routing instances. However, there is no route to the customer LANs in the VR routing instances, so we need to add them. To do this, we will share routes from the VRF routing instances into the VR routing instances using RIB groups.

For example, for traffic to the Customer A LAN, we will share routes from the Cust-A-VRF.inet.0 routing table into the Cust-A-VR.inet.0 routing table. Figure 8 on page 46 shows the traffic flow through PE1 for inline NAT traffic coming from cloud services going to the Customer A LAN.

Figure 8: Traffic Flow on PE1 for Next-Hop Style Inline NAT Traffic from Cloud Services to the Customer A LAN



Before we set up route sharing on PE1, there is only a default static route in the Cust-A-VR.inet.0 routing table:

```
host@PE1> show route table Cust-A-VR.inet.0
Cust-A-VR.inet.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 02:10:53
                    > via si-0/0/0.1
```

To share routes from the Cust-A-VRF.inet.0 routing table with the Cust-A-VR.inet.0 routing table:

1. Create policies that match the routes to be shared.
   - Term 1 matches routes that provide reachability back to the customer LANs.
   - Term 2 matches interface routes that provide reachability to the CE devices.

```
policy-options {
    policy-statement leak-to-Cust-A-VR { ## share matching route with Cust-A-VR
    term 1 {
        from {
            route-filter 10.250.100.0/24 orlonger; ## match routes to Cust LAN
        }
        then accept;
```

```
    }
    term 2 {
        from protocol direct; ## match interface routes
        then accept;
    }
    term 3 {
        then reject;
    }
}
policy-statement leak-to-Cust-B-VR {  ## share matching route with Cust-B-VR
    term 1 {
        from {
            route-filter 10.250.100.0/24 orlonger; ## match routes to Cust LAN
        }
        then accept;
    }
    term 2 {
        from protocol direct; ## match interface routes
        then accept;
    }
    term 3 {
        then reject;
    }
}
```

2. Create RIB groups for route sharing between tables.

   - With the `import-rib` statement, list the source routing table to be shared followed by the destination routing table into which the routes will be imported.
   - With the `import-policy` statement, specify the policy used to define the specific routes that will be shared.

```
routing-options {
    rib-groups {
        Cust-A_leak-VRF-to-VR {  ## share routes from A-VRF to A-VR
        import-rib [ Cust-A-VRF.inet.0 Cust-A-VR.inet.0 ];
        import-policy leak-to-Cust-A-VR;
    }
    Cust-B_leak-VRF-to-VR {  ## share routes from B-VRF to B-VR
    import-rib [ Cust-B-VRF.inet.0 Cust-B-VR.inet.0 ];
```

```
    import-policy leak-to-Cust-B-VR;
}
```

3. In the VRF routing instances created previously, apply the RIB groups to the desired routes.
   - To import directly connected routes, apply the RIB group under the `routing-options` hierarchy.
   - To import the customer LAN routes (PE1 receives these routes through the CE-to-PE EBGP peerings), apply the RIB group under the `protocols` hierarchy.

```
routing-instances {
    Cust-A-VRF {
        routing-options {
            interface-routes {  ## sharing interface routes with A-VR
            rib-group inet Cust-A_leak-VRF-to-VR;
        }
    }
    protocols {
        bgp {
            group to-Cust-A {
                family inet {
                    unicast {  ## sharing Cust-A network with A-VR
                    rib-group Cust-A_leak-VRF-to-VR;
                }
            }
        }
    }
}
```

After we set up route sharing, a direct interface route and a BGP route to the customer A LAN have been added to the Cust-A-VR.inet.0 routing table:

```
host@PE1> show route table Cust-A-VR.inet.0
Cust-A-VR.inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 02:26:20
                    > via si-0/0/0.1
10.0.1.0/24        *[Direct/0] 00:00:50
                    > via xe-0/0/0.0
10.250.100.101/32  *[BGP/170] 00:00:50, localpref 100
                      AS path: 65101 I, validation-state: unverified
```

```
                    > to 10.0.1.2 via xe-0/0/0.0
```

Return traffic can now reach the Customer LANs.

Verifying Next-Hop Style Inline NAT

1. From CE1, verify connectivity between CE1 and cloud services.

```
host@CE1> ping 10.250.200.200 source 10.250.100.101 count 4
PING 10.250.200.200 (10.250.200.200): 56 data bytes
64 bytes from 10.250.200.200: icmp_seq=0 ttl=62 time=1.246 ms
64 bytes from 10.250.200.200: icmp_seq=1 ttl=62 time=1.042 ms
64 bytes from 10.250.200.200: icmp_seq=2 ttl=62 time=1.053 ms
64 bytes from 10.250.200.200: icmp_seq=3 ttl=62 time=1.083 ms

--- 10.250.200.200 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.042/1.106/1.246/0.082 ms
```

2. From PE1, show NAT statistics to verify that traffic is being NAT-translated.

```
host@PE1> show services inline nat statistics
Service PIC Name                                    si-0/0/0

 Control Plane Statistics
     Received IPv4 packets                             0
     ICMPv4 error packets pass through                 0
     ICMPv4 error packets locally generate             0
     Dropped IPv4 packets                              0
     Received IPv6 packets                             0
     ICMPv6 error packets pass through for NPTv6       0
     ICMPv6 error packets locally generated for NPTv6  0
     Dropped IPv6 packets                              0

Data Plane Statistics          Packets                 Bytes
     IPv4 NATed packets           73                    7154
     IPv4 deNATed packets         73                    7446
     IPv4 error packets           0                     0
     IPv4 skipped packets         0                     0
     IPv6 NATed packets           0                     0
```

```
    IPv6 deNATed packets          0                    0
    IPv6 error packets            0                    0
    IPv6 skipped packets          0                    0
```

3. From PE1, verify the NAT pools being used to translate addresses for Customers A and B.

```
host@PE1> show services inline nat pool
Interface: si-0/0/0, Service set: NH-STYLE-SS-NAT_A
  NAT pool: A, Translation type: BASIC NAT44
    Address range: 192.0.2.0-192.0.2.255
    NATed packets: 33, deNATed packets: 33, Errors: 0, Skipped packets: 0

Interface: si-0/0/0, Service set: NH-STYLE-SS-NAT_B
  NAT pool: B, Translation type: BASIC NAT44
    Address range: 198.51.100.0-198.51.100.255
    NATed packets: 23, deNATed packets: 23, Errors: 0, Skipped packets: 0
```

4. From cloud services, verify that the router is receiving the pings from PE1s Customer A pool of NAT translated source addresses (192.0.2.0/24).

   Run pings again from CE1 to cloud services, and enter the following command on cloud services.

```
host@CldSvcs> monitor traffic interface xe-0/0/0 detail no-resolve
Address resolution is OFF.
Listening on xe-0/0/0, capture size 1514 bytes
21:54:42.937849  In IP (tos 0xc0, ttl   1, id 38220, offset 0, flags [none], proto: TCP (6),
length: 52) 192.168.1.1.179 > 192.168.1.2.55253: . ack 2452754676 win 16384
<nop,nop,timestamp 117113408 1131425253>
21:54:43.651296  In IP (tos 0x0, ttl  62, id 36015, offset 0, flags [none], proto: ICMP (1),
length: 84) 192.0.2.101 > 10.250.200.200: ICMP echo request, id 7224, seq 24, length 64
21:54:43.651336 Out IP (tos 0x0, ttl  64, id 55445, offset 0, flags [none], proto: ICMP (1),
length: 84) 10.250.200.200 > 192.0.2.101: ICMP echo reply, id 7224, seq 24, length 64
```

5. From PE3, show the BGP Layer 3 routing table. This table verifies that NAT translation is happening. 192.0.2.0 /24 is address Pool A for Customer A traffic. 198.51.100.0 is address Pool B for Customer B traffic

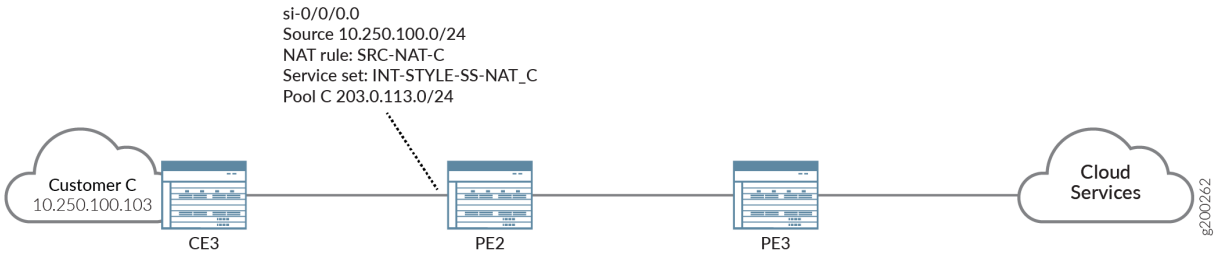
```
user@PE3> show route table bgp.l3vpn.0
bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.250.1.1:1:192.0.2.0/24
                   *[BGP/170] 00:20:09, localpref 100, from 10.250.1.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
10.250.1.1:2:198.51.100.0/24
                   *[BGP/170] 00:20:09, localpref 100, from 10.250.1.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.1 via xe-0/0/0.0, label-switched-path PE3toPE1
```

Configuring Interface-Style Inline NAT on PE2

For traffic from Customer C, we are using interface-style inline NAT. This section shows how to configure interface-based inline NAT, which uses a simpler configuration than next-hop style NAT.

Figure 9: Interface-Style NAT Configuration



The following configuration items are used in this section:

- Inline service interface—a virtual interface that resides on the Packet Forwarding Engine of the MPC. To access services, traffic flows in and out of the si- (service-inline) interface.
- Service set—defines the service(s) performed, and identifies which inline interface(s) will feed traffic into and out of the service set. This section implements interface-style service sets, where packets arriving at an interface are sent through the inline service interface.
- NAT rule—uses an if-then structure (similar to firewall filters) to define match conditions and then apply address translation to the matching traffic.
- NAT pool—a user-defined set of IP addresses that the NAT rule uses for translation.
- VRF routing instance—the si- interface is added to the VRF routing instance for Customer C.

Figure 10 on page 52 shows the traffic flow on PE2 for traffic sent from the Customer C LAN to cloud services.

Figure 10: Traffic Flow on PE2 for Interface-Style Inline NAT traffic from Customer C LAN to Cloud Services

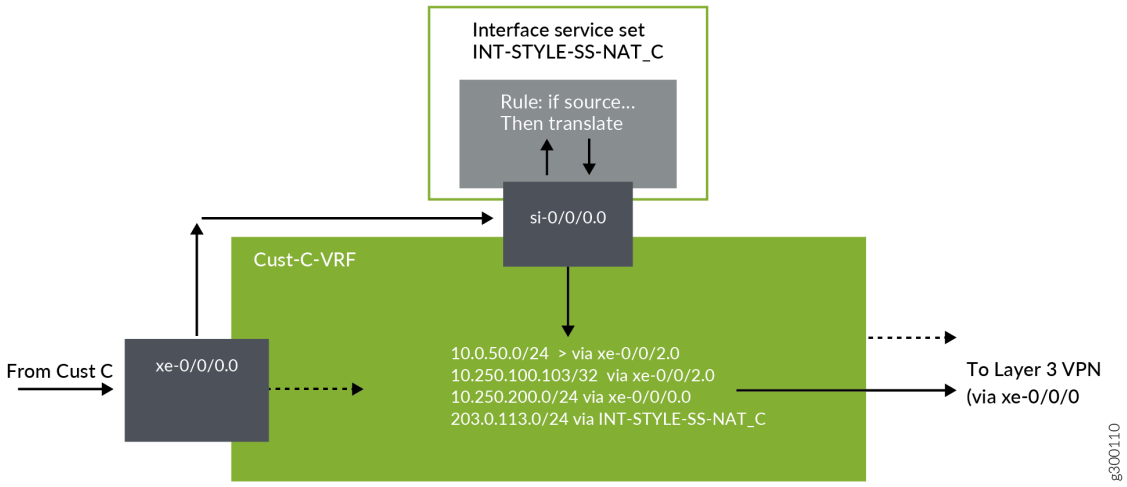
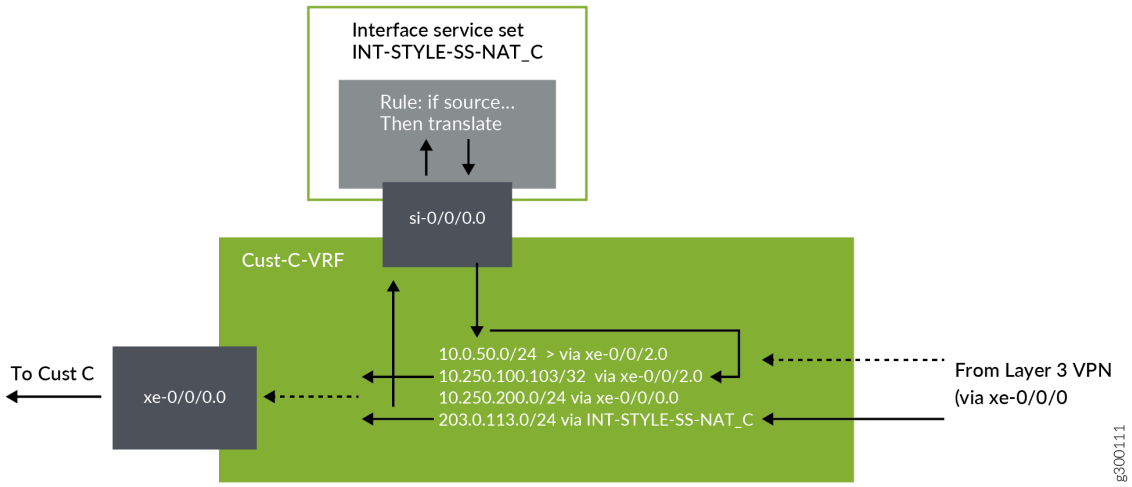


Figure 11 on page 52 shows the traffic flow on PE2 for traffic from cloud services to the Customer C LAN.

Figure 11: Traffic Flow on PE2 for Interface-Style NAT Traffic from Cloud Services to Customer C LAN



To configure interface style inline NAT on PE2:

1. Enable inline services for the relevant FPC slot and PIC slot, and define the amount of bandwidth to dedicate for inline services.

The FPC and PIC settings map to the si- interface configured previously.

```
chassis {
    fpc 0 {
        pic 0 {
            inline-services {
                bandwidth 1g;
            }
        }
    }
}
```

2. Configure the service interface used for NAT. Interface-style NAT requires only one interface.

```
interfaces {
    si-0/0/0 {
        unit 0 {
            family inet; ## protocol family that will need NAT services
        }
    }
}
```

3. Configure a NAT pool.

```
services {
    nat {
        pool C {
            address 203.0.113.0/24;
        }
    }
}
```

4. Configure a NAT rule that:

   - Matches traffic from the Customer C network.
   - Applies the pool from which to obtain an address.
   - Applies basic NAT 44, a type of static NAT that applies to IPv4 traffic.

```
services {
    nat {
```

```
        rule SRC-NAT-C { ## applied to traffic from Customer C
        match-direction input;  ## direction from which traffic is received
        term r1 {
            from {
                source-address {
                    10.250.100.0/24;  ## customer C
                }
            }
            then {
                translated {
                    source-pool C;
                    translation-type {
                        basic-nat44; ## type of one-to-one static NAT
                    }
                }
            }
        }
    }
}
```

5. Configure an interface-style service set that associates the service interface to the NAT service. In this case, the NAT service uses the SRC-NAT-C NAT rule.

   Traffic will flow into and out of the si- interface to access the inline NAT service.

```
services {
    service-set INT-STYLE-SS-NAT_C {
        nat-rules SRC-NAT-C;
        interface-service {  ## defines that this is an interface-style service set
        service-interface si-0/0/0.0;
    }
}
```

6. Apply the input and output service set to the xe-0/0/2 interface, which is the interface to Customer C. This configuration specifies that all traffic to and from Customer C is redirected through the service set.

```
interfaces  {
    xe-0/0/3{
        unit 0 {
            family inet {
                service {
```

```
                    input {
                        service-set INT-STYLE-SS-NAT_C;
                    }
                    output {
                        service-set INT-STYLE-SS-NAT_C;
                    }
                }
            }
        }
    }
}
```

7. Add the si- interface to the Cust-C VRF routing instance.

```
routing-instances {
    Cust-C {
        interface si-0/0/0.0;
    }
}
```

With this configuration in place, traffic from Customer C enters the interface at PE2, is redirected through the service interface to the service set for NAT translation. The traffic is then returned to the VRF routing instance and can be sent across the VPN as usual.

Verifying Interface Style Inline NAT

1. From CE3, verify connectivity between CE3 and cloud services.

```
host@CE3> ping 10.250.200.200 source 10.250.100.103 count 4
PING 10.250.200.200 (10.250.200.200): 56 data bytes
64 bytes from 10.250.200.200: icmp_seq=0 ttl=62 time=1.109 ms
64 bytes from 10.250.200.200: icmp_seq=1 ttl=62 time=1.111 ms
64 bytes from 10.250.200.200: icmp_seq=2 ttl=62 time=1.087 ms
64 bytes from 10.250.200.200: icmp_seq=3 ttl=62 time=1.192 ms

--- 10.250.200.200 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.087/1.125/1.192/0.040 ms
```

2. From PE2, show the inline NAT statistics to verify that traffic is being NAT-translated.

```
host@PE2> show services inline nat statistics
 Service PIC Name                                    si-0/0/0

 Control Plane Statistics
     Received IPv4 packets                              0
     ICMPv4 error packets pass through                  0
     ICMPv4 error packets locally generate              0
     Dropped IPv4 packets                               0
     Received IPv6 packets                              0
     ICMPv6 error packets pass through for NPTv6        0
     ICMPv6 error packets locally generated for NPTv6   0
     Dropped IPv6 packets                               0

 Data Plane Statistics           Packets                 Bytes
     IPv4 NATed packets            18                    1248
     IPv4 deNATed packets          12                    1008
     IPv4 error packets            0                     0
     IPv4 skipped packets          0                     0
     IPv6 NATed packets            0                     0
     IPv6 deNATed packets          0                     0
     IPv6 error packets            0                     0
     IPv6 skipped packets          0                     0
```

3. From PE2, verify that inline NAT is being applied correctly.

```
host@PE2> show services inline nat pool
Interface: si-0/0/0, Service set: INT-STYLE-SS-NAT_C
  NAT pool: C, Translation type: BASIC NAT44
    Address range: 203.0.113.0-203.0.113.255
    NATed packets: 11, deNATed packets: 5, Errors: 0, Skipped packets: 0
```

4. From PE2, verify that the NAT translation pool shows in the routing table for the customer C network.

```
host@PE2> show route table Cust-C.inet.0
Cust-C.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.50.0/24       *[Direct/0] 1d 00:27:53
```

```
                    > via xe-0/0/2.0
10.0.50.1/32       *[Local/0] 1d 00:27:53
                      Local via xe-0/0/2.0
10.250.100.103/32  *[BGP/170] 19:01:38, localpref 100
                      AS path: 65103 I, validation-state: unverified
                    > to 10.0.50.2 via xe-0/0/2.0
10.250.200.0/24    *[BGP/170] 20:20:30, localpref 100, from 10.250.1.3
                      AS path: 65200 I, validation-state: unverified
                    > to 172.17.1.2 via xe-0/0/0.0, label-switched-path PE2toPE3
203.0.113.0/24     *[Static/1] 02:20:18
                      Service to INT-STYLE-SS-NAT_C
```

5. From cloud services, verify that the router is receiving the pings from PE2s pool of NAT translated source addresses (203.0.113.0/24).

   Run pings again from CE3 to cloud services, and enter the following command on cloud services.

```
host@CldSvcs> monitor traffic interface xe-0/0/0 detail no-resolve
Address resolution is OFF.
Listening on xe-0/0/0, capture size 1514 bytes
19:41:09.879603  In IP (tos 0x0, ttl  62, id 29437, offset 0, flags [none], proto: ICMP (1),
length: 84) 203.0.113.103 > 10.250.200.200: ICMP echo request, id 6667, seq 6, length 64
19:41:09.879644 Out IP (tos 0x0, ttl  64, id 51068, offset 0, flags [none], proto: ICMP (1),
length: 84) 10.250.200.200 > 203.0.113.103: ICMP echo reply, id 6667, seq 6, length 64
19:41:10.880605  In IP (tos 0x0, ttl  62, id 29456, offset 0, flags [none], proto: ICMP (1),
length: 84) 203.0.113.103 > 10.250.200.200: ICMP echo request, id 6667, seq 7, length 64
19:41:10.880644 Out IP (tos 0x0, ttl  64, id 51088, offset 0, flags [none], proto: ICMP (1),
length: 84) 10.250.200.200 > 203.0.113.103: ICMP echo reply, id 6667, seq 7, length 64
.
.
. ^C
18 packets received by filter
0 packets dropped by kernel
```

Complete Router Configurations

IN THIS SECTION

This section has the complete configuration of each router.

PE1 Configuration

```
## Configuring the Core
    interfaces {
    xe-0/0/2 {
        description "Outside to PE3";
        unit 0 {
            family inet {
                address 172.16.1.1/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.1.1/32;
            }
        }
    }
}
protocols {
    rsvp {
        interface xe-0/0/2.0;
```

```
    }
    mpls {
        no-cspf;
        label-switched-path PE1toPE3 {
            to 10.250.1.3; ## PE3 loopback address
        }
        interface xe-0/0/2.0; ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
            local-address 10.250.1.1;
            neighbor 10.250.1.3; ## PE3 loopback address
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/2.0; ## core-facing interface
            interface lo0.0;
        }
    }
}
routing-options {
    autonomous-system 65000;
}
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet; ## actually applied per flow
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
## Configuring the Layer 3 VPN
interfaces {
    xe-0/0/0 {
        description "Inside to CE1_Cust-A";
        unit 0 {
            family inet {
                address 10.0.1.1/24;
            }
```

```
        }
    }
    xe-0/0/1 {
        description "Inside to CE1_Cust-B";
        unit 0 {
            family inet {
                address 10.0.1.2/24;
            }
        }
    }
}
policy-options {
    policy-statement CustA-to-CloudSvcs {
        term 1 {
            from protocol static;
            then {
                community add SpokeA; ## Add SpokeA tag when BGP exports route
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement CustB-to-CloudSvcs {
        term 1 {
            from protocol static;
            then {
                community add SpokeB; ## Add SpokeB tag when BGP exports route
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement from-CloudSvcs { ## add routes received with Hub tag to routing table
    term 1 {
        from community Hub;
        then accept;
    }
    term 2 {
        then reject;
```

```
    }
}
routing-instances {
    Cust-A-VRF {
        instance-type vrf;
        interface xe-0/0/0.0;
        route-distinguisher 10.250.1.1:1;
        vrf-import from-CloudSvcs;
        vrf-export CustA-to-CloudSvcs;
        vrf-table-label;
    }
    Cust-B-VRF {
        instance-type vrf;
        interface xe-0/0/1.0;
        route-distinguisher 10.250.1.1:2;
        vrf-import from-CloudSvcs;
        vrf-export CustB-to-CloudSvcs;
        vrf-table-label;
    }
}
protocols {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast;
        }
    }
}
## Configuring the Connection to CE1
interfaces {
    xe-0/0/1 {
        description Cust-B_to_PE1;
        unit 0 {
            family inet {
                address 10.0.1.4/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.102/32;
            }
```

```
        }
    }
}
## Configuring the Connection to CE2
routing-instances {
    Cust-B-VRF {
        protocols {
            bgp {
                group to-Cust-B {
                    type external;
                    peer-as 65102
                    neighbor 10.0.1.4; ## CE2 interface address
                }
            }
        }
    }
}
## Configuring Next-hop Style Inline NAT
chassis {
    fpc 0 {
        pic 0 {
            inline-services {
                bandwidth 1g;
            }
        }
    }
}
interfaces {
    si-0/0/0 {
        unit 1 { ## Customer A traffic
        family inet; ## causes NAT to be applied to IPv4 traffic
        service-domain inside; ## traffic from customer A LAN to core
    }
    unit 2 { ## Customer A traffic
    family inet;
    service-domain outside; ## customer A traffic from core to customer LAN
}
unit 3 { ## Customer B traffic
    family inet;
    service-domain inside; ## traffic from customer B LAN to core
}
unit 4 { ## Customer B traffic
    family inet;
```

```
    service-domain outside; ## customer B traffic from core to customer LAN
}
services {
    nat {
        pool A { ## applied to customer A traffic
        address 192.0.2.0/24;
    }
    pool B { ## applied to customer B traffic
    address 198.51.100.0/24;
}
services {
    nat {
        rule SRC-NAT-A { ## applied to traffic from customer A
        match-direction input; ## direction from which traffic is received
        term r1 {
            from {
                source-address {
                    10.250.100.0/24; ## from customer A
                }
            }
            then {
                translated {
                    source-pool A;
                    translation-type {
                        basic-nat44; ## type of one-to-one static NAT
                    }
                }
            }
        }
    }
    rule SRC-NAT-B { ## applied to traffic from customer B
    match-direction input;
    term r1 {
        from {
            source-address {
                10.250.100.0/24; ## from customer B
            }
        }
        then {
            translated {
                source-pool B;
                translation-type {
                    basic-nat44; ## type of one-to-one static NAT
```

```
                }
            }
        }
    }
}
services {
    service-set NH-STYLE-SS-NAT_A { ## applied to Customer A traffic
    nat-rules SRC-NAT-A;
    next-hop-service {
        inside-service-interface si-0/0/0.1;
        outside-service-interface si-0/0/0.2;
    }
}
service-set NH-STYLE-SS-NAT_B { ## applied to Customer B traffic
    nat-rules SRC-NAT-B;
    next-hop-service {
        inside-service-interface si-0/0/0.3;
        outside-service-interface si-0/0/0.4;
    }
}
routing-instances {
    Cust-A-VR {
        instance-type virtual-router;
        interface si-0/0/0.1; ## Cust A inside service interface
        routing-options {
            static {
                route 0.0.0.0/0 next-hop si-0/0/0.1; ## incoming Cust A traffic to si
            }
        }
    }
    Cust-B-VR {
        instance-type virtual-router;
        interface si-0/0/0.3; ## Cust B inside service interface
        routing-options {
            static {
                route 0.0.0.0/0 next-hop si-0/0/0.3; ## incoming Cust B traffic to si
            }
        }
    }
}
firewall {
    filter CustA-to-NAT-VR {
        term 1 {
```

```
            from {
                source-address { ## traffic from Customer A network
                10.250.100.0/24;
            }
        }
        then {
            routing-instance Cust-A-VR; ## sends matching traffic to this RI
        }
    }
    term 2 {
        then accept;
    }
}
filter CustB-to-NAT-VR {
    term 1 {
        from {
            source-address { ## traffic from Customer B network
            10.250.100.0/24;
        }
    }
    then {
        routing-instance Cust-B-VR; ## sends matching traffic to this RI
    }
}
term 2 {
    then accept;
}
interfaces {
    xe-0/0/0 { ## to Customer A
    unit 0 {
        family inet {
            filter {
                input CustA-to-NAT-VR;
            }
        }
    }
}
routing-instances {
    Cust-A-VRF {
        interface si-0/0/0.2;
    }
    Cust-B-VRF {
        interface si-0/0/0.4;
```

```
    }
}
## Allow return traffic from Cloud Services
policy-options {
    policy-statement leak-to-Cust-A-VR { ## share matching route with Cust-A-VR
    term 1 {
        from {
            route-filter 10.250.100.0/24 orlonger; ## match routes to Cust LAN
        }
        then accept;
    }
    term 2 {
        from protocol direct; ## match interface routes
        then accept;
    }
    term 3 {
        then reject;
    }
}
policy-statement leak-to-Cust-B-VR { ## share matching route with Cust-B-VR
    term 1 {
        from {
            route-filter 10.250.100.0/24 orlonger; ## match routes to Cust LAN
        }
        then accept;
    }
    term 2 {
        from protocol direct; ## match interface routes
        then accept;
    }
    term 3 {
        then reject;
    }
}
routing-options {
    rib-groups {
        Cust-A_leak-VRF-to-VR { ## share routes from A-VRF to A-VR
        import-rib [ Cust-A-VRF.inet.0 Cust-A-VR.inet.0 ];
        import-policy leak-to-Cust-A-VR;
    }
    Cust-B_leak-VRF-to-VR { ## share routes from B-VRF to B-VR
    import-rib [ Cust-B-VRF.inet.0 Cust-B-VR.inet.0 ];
    import-policy leak-to-Cust-B-VR;
```

```
}
routing-instances {
    Cust-A-VRF {
        routing-options {
            interface-routes { ## sharing interface routes with A-VR
            rib-group inet Cust-A_leak-VRF-to-VR;
        }
    }
    protocols {
        bgp {
            group to-Cust-A {
                family inet {
                    unicast { ## sharing Cust-A network with A-VR
                    rib-group Cust-A_leak-VRF-to-VR;
                }
            }
        }
    }
}
```

PE2 Configuration

```
## Configuring the Corre
    interfaces {
    xe-0/0/0 {
        description "Outside to PE3";
        unit 0 {
            family inet {
                address 172.17.1.1/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.1.2/32;
            }
        }
    }
}
```

```
protocols {
    rsvp {
        interface xe-0/0/0.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE2toPE3 {
            to 10.250.1.3; ## PE3 loopback address
        }
        interface xe-0/0/0.0 ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
            local-address 10.250.1.2; ## local loopback address
            family inet {
                unicast;
            }
            neighbor 10.250.1.3; ## lo0 address of PE3
        }
    }
    ospf {
        area 0.0.0.0 {
            interface xe-0/0/0.0;
            interface lo0.0;
        }
    }
}
routing-options {
    autonomous-system 65000;
}
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet; ## actually applied per flow
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
policy-options {
    policy-statement CustA-to-CloudSvcs {
```

```
        term 1 {
            from protocol static;
            then {
                community add SpokeA; ## Add SpokeA tag when BGP exports route
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement CustB-to-CloudSvcs {
        term 1 {
            from protocol static;
            then {
                community add SpokeB; ## Add SpokeB tag when BGP exports route
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement from-CloudSvcs { ## add routes received with Hub tag to routing table
    term 1 {
        from community Hub;
        then accept;
    }
    term 2 {
        then reject;
    }
}
routing-instances {
    Cust-A-VRF {
        instance-type vrf;
        interface xe-0/0/0.0;
        route-distinguisher 10.250.1.1:1;
        vrf-import from-CloudSvcs;
        vrf-export CustA-to-CloudSvcs;
        vrf-table-label;
    }
    Cust-B-VRF {
        instance-type vrf;
```

```
        interface xe-0/0/1.0;
        route-distinguisher 10.250.1.1:2;
        vrf-import from-CloudSvcs;
        vrf-export CustB-to-CloudSvcs;
        vrf-table-label;
    }
}
protocols {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast;
        }
    }
}
## Configuring the Layer 3 VPN
interfaces {
    xe-0/0/3 {
        description "Inside to CE1_Cust-C";
        unit 0 {
            family inet {
                address 10.0.50.1/24;
            }
        }
    }
}
policy-options {
    policy-statement CustC-to-CloudSvcs { ## Add SpokeC tag when BGP exports route
    term 1 {
        from protocol static;
        then {
            community add SpokeC;
            accept;
        }
    }
    term 2 {
        then reject;
    }
}
policy-statement from-CloudSvcs { ## add routes received with Hub tag to routing table
    term 1 {
        from community Hub;
        then accept;
```

```
    }
    term 2 {
        then reject;
    }
}
routing-instances {
    Cust-C {
        instance-type vrf;
        interface xe-0/0/3.0;
        route-distinguisher 10.250.1.2:3;
        vrf-import from-CloudSvcs;
        vrf-export CustC-to-CloudSvcs;
        vrf-table-label;
    }
}
protocols {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast
        }
    }
}
## Configuring the Connection to CE3
routing-instances {
    Cust-C {
        protocols {
            bgp {
                group to-Cust-C {
                    type external;
                    peer-as 65103
                    neighbor 10.0.50.2; ## CE3 interface address
                }
            }
        }
    }
}
## Configuring Interface-Style Inline NAT
chassis {
    fpc 0 {
        pic 0 {
            inline-services {
                bandwidth 1g;
```

```
            }
        }
    }
}
interfaces {
    si-0/0/0 {
        unit 0 {
            family inet; ## protocol family that will need NAT services
        }
    }
}
services {
    nat {
        pool C {
            address 203.0.113.0/24;
        }
    }
}
services {
    nat {
        rule SRC-NAT-C { ## applied to traffic from Customer C
        match-direction input; ## direction from which traffic is received
        term r1 {
            from {
                source-address {
                    10.250.100.0/24; ## customer C
                }
            }
            then {
                translated {
                    source-pool C;
                    translation-type {
                        basic-nat44; ## type of one-to-one static NAT
                    }
                }
            }
        }
    }
}
services {
    service-set INT-STYLE-SS-NAT_C {
        nat-rules SRC-NAT-C;
        interface-service { ## defines that this is an interface-style service set
```

```
        service-interface si-0/0/0.0;
    }
}
interfaces {
    xe-0/0/3{
        unit 0 {
            family inet {
                service {
                    input {
                        service-set INT-STYLE-SS-NAT_C;
                    }
                    output {
                        service-set INT-STYLE-SS-NAT_C;
                    }
                }
            }
        }
    }
}
routing-instances {
    Cust-C {
        interface si-0/0/0.0;
    }
}
```

PE3 Configuration

```
## Configuring the Core
    interfaces {
    xe-0/0/0 {
        description "to PE1";
        unit 0 {
            family inet {
                address 172.16.1.2/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    xe-0/0/1 {
        description "to PE2";
        unit 0 {
```

```
            family inet {
                address 172.17.1.2/24;
            }
            family mpls; ## allows interface to support MPLS protocol traffic
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.1.3/32;
            }
        }
    }
}
protocols {
    rsvp {
        interface xe-0/0/0.0;
        interface xe-0/0/1.0;
    }
    mpls {
        no-cspf;
        label-switched-path PE3toPE1 {
            to 10.250.1.1; ## PE1 loopback address
        }
        label-switched-path PE3toPE2 {
            to 10.250.1.2; ## PE2 loopback address
        }
        interface xe-0/0/0.0; ## core-facing interface
        interface xe-0/0/1.0; ## core-facing interface
    }
    bgp {
        group IBGP {
            type internal;
            local-address 10.250.1.3; ## local loopback address
            family inet {
                unicast;
            }
            neighbor 10.250.1.1; ## lo0 address of spoke router PE1
            neighbor 10.250.1.2; ## lo0 address of spoke router PE2
        }
    }
    ospf {
        area 0.0.0.0 {
```

```
            interface xe-0/0/0.0;
            interface xe-0/0/1.0;
            interface lo0.0;
        }
    }
}
routing-options {
    autonomous-system 65000;
}
policy-options {
    policy-statement LB { ## load balancing policy
    then {
        load-balance per-packet; ## actually applied per session
    }
}
routing-options {
    forwarding-table { ## adds the LB policy to the forwarding table
    export LB;
}
## Configuring the Layer 3 VPN
interfaces {
    xe-0/0/2 {
        description "PE3 to CloudSvcs";
        unit 0 {
            family inet {
                address 192.168.1.1/24;
            }
        }
    }
}
policy-options {
    policy-statement from-Cust { ## add routes with hub tag to routing table
    term 1 {
        from community [ SpokeA SpokeB SpokeC ];
        then accept;
    }
}
policy-statement to-Cust { ## add hub tag when BGP exports route
    term 1 {
        from protocol bgp;
        then {
            community add Hub;
            accept;
```

```
        }
    }
}
routing-instances {
    CloudSvcs {
        instance-type vrf;
        interface xe-0/0/2.0;
        route-distinguisher 10.250.1.3:100; ## PE3
        vrf-import from-Cust;
        vrf-export to-Cust;
    }
}
protocols {
    bgp {
        group IBGP {
            family inet-vpn { ## enables MP-BGP to carry IPv4 Layer 3 VPN NLRI
            unicast
        }
    }
}
```

CE1 Configuration

```
interfaces {
    xe-0/0/0 {
        description Cust-A_to_PE1;
        unit 0 {
            family inet {
                address 10.0.1.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.101/32;
            }
        }
    }
}
policy-options {
```

```
    policy-statement CustA-to-PE1 {
        term 1 {
            from {
                route-filter 10.250.100.101/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
protocols {
    bgp {
        group to-PE1 {
            type external;
            export CustA-to-PE1; ## BGP advertises routes to the L3VPN
            peer-as 65000;
            neighbor 10.0.1.1; ## PE1 interface address
        }
    }
}
routing-options {
    autonomous-system 65101;
}
```

CE2 Configuration

```
interfaces {
    xe-0/0/1 {
        description Cust-B_to_PE1;
        unit 0 {
            family inet {
                address 10.0.1.4/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.102/32;
```

```
            }
        }
    }
}
policy-options {
    policy-statement CustB-to-PE1 {
        term 1 {
            from {
                route-filter 10.250.100.102/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
protocols {
    bgp {
        group to-PE1 {
            type external;
            export CustB-to-PE1;
            peer-as 65000;
            neighbor 10.0.1.2; ## PE1 interface address
        }
    }
}
routing-options {
    autonomous-system 65102;
}
```

CE3 Configuration

```
interfaces {
    xe-0/0/2 {
        description Cust-C_to_PE2;
        unit 0 {
            family inet {
                address 10.0.50.2/24;
            }
        }
```

```
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.250.100.103/32;
            }
        }
    }
}
policy-options {
    policy-statement CustC-to-PE2 {
        term 1 {
            from {
                route-filter 10.250.100.103/32 exact;
            }
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}
protocols {
    bgp {
        group to-PE2 {
            type external;
            export CustC-to-PE2;
            peer-as 65000;
            neighbor 10.0.50.1; ## PE2 interface address
        }
    }
}
routing-options {
    autonomous-system 65103;
}
```