JUNIPER NETWORKS BUSINESS EDGE SOLUTION 1.0 DESIGN GUIDE

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks Business Edge 1.0 Design Guide

Junos OS 12.3R3 or later

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 http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to terms and conditions of that EULA.

Table of Contents

List of Figures

List of Tables

CHAPTER 1 Business Edge Overview

- Introduction
- Target Audience
- Problem Statement
- Benefits
- Services

Introduction

The Juniper Networks business edge solution provides design guidance and configurations that enable the provisioning of services to business customers and to support other carrier services to residential and mobile operating units. The solution provides a proven path to service convergence using common IP infrastructure so that the provider can quickly, safely, and conveniently realize the benefits of a fully verified Juniper Networks-based reference architecture. The solution is a complete and deployable network architecture designed to intelligently leverage the variety of advanced and often overlooked technologies inherent in Juniper Networks software and hardware.

Juniper Networks has developed this solution with the goal of enabling a streamlined business edge that can create new areas for monetization and help prevent erosion of service margins. By accelerating time to revenue at the business edge, the solution is the first step in streamlining the provider edge and creates new areas for expansion and consolidation. The next step in this transformation is the introduction of the Juniper universal edge. The universal edge is a consolidation of business services, residential services, and wireless edge onto a common IP infrastructure that reduces the need for redundant networks and network elements, enabling expense reduction and optimizing traffic flow to support a complete footprint of service provider service offerings.

Service providers constantly look to extract additional value from the network by positioning themselves to profitably leverage converging services and network functionality. A performance focused, highly reliable business edge is needed to cost-effectively meet the extraordinary growth in subscribers, services, and traffic driven by an increasingly connected workforce and business requirements that leverage the network as a commodity rather than a luxury. The first step in the transformation to a universal edge solution is the adoption of a complete business edge architecture that enables a smooth transition to an edge that supports not only business, but residential subscribers and mobility networks.

Target Audience

This guide is intended to assist services providers in designing and implementing business services for customers, partners, and other business units within the provider's footprint (for example, services to mobile or residential access networks). The business targets for these services are most often:

- Carriers that provide connectivity (with optional value-added services) for business customers. Value-added services are targeted to the customers that are looking to outsource IT services to carriers. These IT services include security and Network Address Translation (NAT) services.
- Carriers and content providers that have an extensive portfolio of higher level services that need connectivity in order to operate. The higher level services might include mobile communication, video broadcasting, or access to various content like video, webmail, or online gaming. For example, in the case of a cellular service provider, Layer 3 VPN service is often used to provide connectivity between base stations and network controllers.

This guide is designed for experienced network professionals, and assumes a broad understanding of network configuration and general networking principles. The target audience for this guide includes:

- Business decision makers—Network planners, decision makers, and business focused stakeholders who must evaluate the impact a new network might have on expenses, new business opportunities, and profitability
- Network architects—Network designers who must consider the implementation, operation, and maintenance of the end-to-end network
- Network engineers—Implementation and operations personnel who are responsible for turning up new service and maintaining or changing existing business customer services

Although readers should have working knowledge of Juniper Networks technology and the Juniper Networks® Junos® operating system software, additional information about any of the concepts and configuration details discussed in this guide is available on the Juniper Networks website at http://www.juniper.net/techpubs/.

Problem Statement

Service providers are being forced to modernize their traditional business service delivery business model using network designs that can provide multiservice networks capable of delivering high-quality voice, television, and data access over a unified network infrastructure. The key drivers of the traditional service provider business model are to optimize cost, increase revenue, and increase profit, all while delivering the highest quality user experience to customers. However, the current trend toward modernization is in direct conflict with these key drivers.

Figure 1 illustrates the challenges that many service providers face.

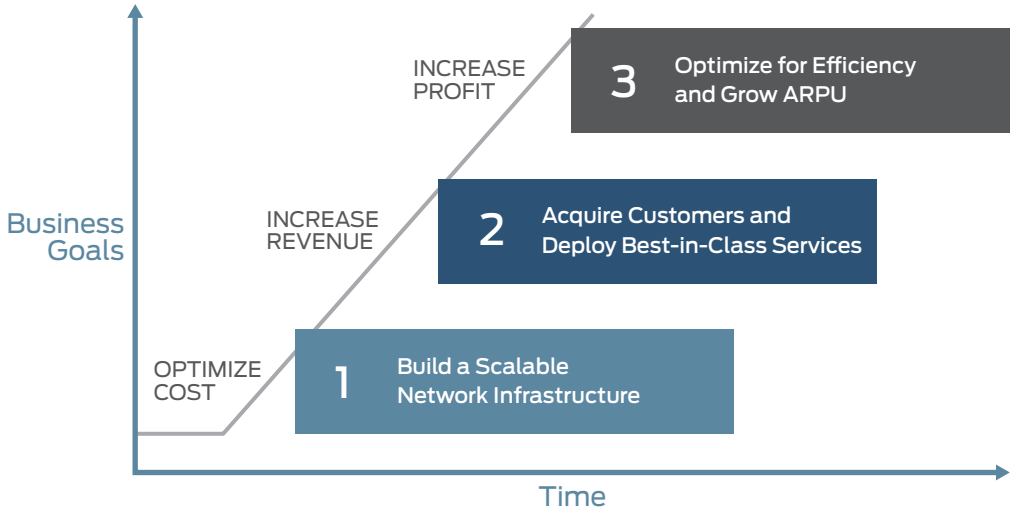

Figure 1: Service provider business edge challenges

In an increasingly competitive market, and in the face of market drivers that demand innovation and expansion rather than maintaining the status quo, meeting stringent business goals becomes increasingly challenging. It is important to understand the drivers of change when trying to address the core business needs of the service provider. The drive to modernize is driven by the following key challenges:

- Market forces have created an environment of unpredictability.
- The network is experiencing unprecedented traffic growth.
- The evolution of the nature of traffic has created merging inflection points that are driven by mobile traffic, cloud services, and a growing mountain of content.

Each of these factors exerts pressure on the service provider core requirement of optimizing cost, increasing revenue, and increasing profit. Market forces have created an environment of unpredictability. Mobile devices, new applications, and the flood of new content place a direct strain on the business network, forcing the business and the service provider to address the network in creative ways. Maintaining and evolving the end-user experience is critical to the success of the network and key to enabling the service provider to meet its core goals. Unprecedented traffic growth, brought about by these market forces, is driving the service provider to examine and optimize the network for flexibility and performance. A service provider network that lacks the agility to meet growing demands is easily overburdened, often resulting in subscriber churn and lost revenue.

The main challenge in the business edge is the future. The growth and purpose of the network of the future is largely driven by mobility, cloud, and video. The world is becoming increasingly mobile and this trend is apparent in the enterprise. By the year 2020, it is projected that 50 billion devices will be connected to the Internet—and many of these devices are in the hands of the business user. The service provider must consider solutions today that enable future scaling to meet the demands of mobile devices on the business network.

Business is increasingly dependent on cloud content, with a projected trillion dollars in public cloud IT services by the year 2020. This growth in centralized, essential business content will place a constantly growing strain on the business network. While video and streaming content is popular today, that growth is only expected to continue, with a projected 80 percent of broadband traffic being streamed via video and audio. These forces exert pressure on the business network and challenge the service provider to build a network that can handle the modern user in addition to the evolving business requirements of the network.

The deluge of new devices, intelligent and always connected, has created a demand for bandwidth that the modern business network is poorly equipped to handle. The flood of new applications and content, driven largely by mobile devices that connect to the business Wi-Fi, is straining even the most robustly designed business networks. In addition to handling non-business-critical and mobile traffic, the business network must also handle business-critical traffic to what is becoming a commonality—the cloud-connected data center. Businesses are centralizing critical applications and data in order to better control the security and availability of the data and to mitigate the expenses associated with hosting and serving data locally to the entire business user base. This compounds the challenge on the service

provider, because the business measures the success of the business network by its ability to handle critical data. With more and more of that data centralized, the importance of the business network, and its role in the success of the business, become even more critical. The business user base, however, places additional expectations on the network. Users expect their business e-mail and voice over IP calls to work flawlessly, while also expecting mobile video and applications to perform seamlessly. While this is not a primary concern of the business, its impact is real and must be factored into the design of future business networks. This challenges the business and the business edge provider to optimize cost and increase revenue and profit in the face of the increasingly complex and growing demand for a high-quality end-user experience.

Addressing all of these challenges requires service providers to focus on building scalable, efficient networks in order to optimize cost. The network must be built with the future at its core; powering growth and expansion quickly and efficiently to enable the service provider to address the evolving needs of the business customer. Optimizing costs alone, however, will not enable the business to maintain profitability. The service provider must also increase revenue by acquiring new customers and deploying best-in-class services. The goal of a successful service provider is to provide an exceptional user experience to new and existing customers, reducing churn and increasing revenue through the addition of new customers and customer services. Finally, the service provider strives to increase profit. This can be accomplished through the creation of a network infrastructure that is optimized for efficiency and future growth. A network that supports the addition of new, revenue-generating services, with a focus on minimizing the time to revenue for those new services, is a network that can enable increased profitability.

Benefits

The Juniper Networks business edge solution enables the provisioning of business services to customers, partners, and other businesses serviced by the provider. The business edge solution design and architecture is derived from best practices and lessons learned from partnerships and the ongoing operation of more than 2,500 service provider and large enterprise networks that rely on Juniper Networks platforms and technology. This solution can be leveraged in its entirety for new business edge deployments, or individual elements can be adopted for selective integration into existing environments.

Customers who choose to implement this business edge solution should use this guide to understand the overall design and learn how each element is configured. The solution consists of foundational customer connection services that cover the most demanded connectivity options for business customers. Each service provides configuration of the typical elements of a business service, including interface configuration, routing configuration, class of service, and security configuration.

The benefits of the Juniper business edge solution are threefold. The solution enables the service provider to optimize cost through the creation of a scalable network infrastructure. In addition, the solution architecture is focused on giving the service provider the ability to increase revenue through the acquisition of new customers and the agile deployment of best-in-class services to those customers. Finally, the solution helps to increase profitability through cost and operational optimization, enabling the service provider to reduce the overall expense of the network and increase the average margin per user.

The following sections provide more detailed information about the benefits of the Juniper Networks business edge solution:

- Solution Cost Benefit
- Solution Deployment Benefit
- Solution Profitability Benefit
- Design Guidance Benefit

Solution Cost Benefit

The Juniper business edge solution optimizes cost through the introduction of an architecture that scales to meet the needs of service providers and their customers both now and into the future. The edge infrastructure in the business edge solution is built to meet the increasing demand for bandwidth-hungry, media-rich applications, as well as the demands placed on the network by the continued movement toward hosted cloud content. The solution is designed to scale in three dimensions: bandwidth, subscriber, and services. The foundation of the solution, the Juniper Networks MX Series 3D Universal Edge Router, is designed to support as much as 80 Tbps of throughput per line card, enabling backhaul and customer-facing performance that scales to meet even the most extreme bandwidth demands. The solution is designed to be highly oversubscribed, allowing for a large number of business subscribers to connect at the edge with support for a truly universal edge, where both mobile and residential subscribers are supported in parallel with business subscribers. Finally, the MX Series routing platform provides services at scale, enabling the provider to introduce new service offerings quickly, and at line rate, to the business networks.

Solution Deployment Benefit

The Juniper Networks business edge solution supports the ability of the service provider to acquire and provision new customers, and it provides for agile deployment of best-in-class services. Defining class of service (CoS) as a required component is the first step in a network being made ready to adopt new services. Other technologies, such as RSVP-TE configuration, can offer bandwidth guarantees to critical traffic flows, further enhancing the network's ability to support new, high priority services. These bandwidth guarantees enable service providers to position their L4-L7 network services nodes (for example, stateful firewall or NAT systems) in the network where they can be used most economically (either closer to the customer premises or closer to the cloud), based on what makes economic sense for the individual provider (depending on the service adoption rates).

The business edge solution also supports the trend toward disintermediation of network and content typically provided by the operator. An operator that offers a one-stop shopping experience, which includes high-speed business bandwidth as well as hosted private cloud services, can capitalize on this trend. Hosted private clouds are a natural business growth opportunity, and service providers are well positioned to offer virtual cloud services using business VPNs. The hosted private cloud is a point in the network where you can build new revenue, reduce total cost of ownership, and increase customer retention. Public cloud services, however, have historically been unable to provide adequate service-level agreements (SLAs) or quality-of-service (QoS) guarantees, and building a cloud can be prohibitively complicated and expensive. The service provider is uniquely positioned to address these challenges by providing multiple services on a single bill with a focus on providing an exceptional experience to the business users.

Solution Profitability Benefit

The Juniper Networks business edge solution is designed to increase profitability through the control of capital and operational expenses (CapEx and OpEx). Design goals, such as reducing the number of nodes in the network and increasing the role of virtualization, can have a direct and immediate impact on these expense centers. At the same time, the base reference architecture creates a network that is well positioned to evolve as new industry trends develop in the market.

Design Guidance Benefit

The final benefit of the solution is the Business Edge Solution Design Guide. This Design Guide provides a one-stop shopping experience for configurations and design considerations related to the implementation and design of a carrier business edge. The configurations contained in this guide provide a baseline for service provider configuration of business edge network elements and customer attachment circuits. These configurations enable a service provider that is evaluating Juniper Networks as a business edge partner to start with predefined configurations, accelerating testing and acceptance of a Juniper Networks-based business edge. Guidance is provided for configuration of all edge elements across several different connection types, enabling service providers a wide array of baseline configurations from which to derive their testing and production configurations.

Services

The Juniper Networks business edge solution consists of several customer services that are based on a common foundation of MPLS technologies. These services cater to divergent customer business network requirements. In many customer networks, these profiles are each implemented as separate networks. One of the primary benefits of the solution architecture is that it enables customers to use a single MX Series chassis to fulfill multiple edge roles at the same time. These varying roles enable business edge solution deployments to use a phased approach with a focus on the solution being extensible and modular in nature. The assumption is that one or both of the foundational profiles introduced in the business edge solution will be deployed in the network and then updated with one or more of the optional profiles included in the services portion of the architecture or as design components of subsequent deployment phases.

The Juniper Networks business edge solution supports the following two services:

- Layer 3 VPNs: The Layer 3 VPN service enables private IP MPLS connectivity between geographically disbursed enterprise locations. A common IP/MPLS network using multiprotocol BGP (MP-BGP) signaling enables the creation of virtual private network services to customers and between customer locations, as well as value-added services that include stateful firewalling, Network Address Translation (NAT), and multitenant cloud connectivity.
- Direct Internet Access (DIA): The DIA service enables public Internet connectivity that service providers typically offer their enterprise customers. This Internet connectivity can be used as plain Internet access or as a peering point over which enterprise services can be provided (for example, IPsec VPN to non-MPLS connected sites, or enterprise Web hosting).

The solution is designed with these services functioning as the foundation elements. Some configuration settings and design choices have been made to accommodate these elements as the main components. That said, either service can be deployed separately or together as part of a business edge architecture. These base services are designed to mimic current standard service provider deployment scenarios and are meant to establish a baseline from which to establish a path toward a unified edge architecture.

On top of the foundational Layer 3 VPN and DIA connectivity, the solution also includes design considerations and configurations for the following additional services:

- Next-Generation Multicast VPN (NG MVPN)
- Stateful firewall service
- Network Address Translation (NAT) service

The following sections provide more detailed information about Juniper Networks business edge service covered in this guide:

- Business Edge Basic Connectivity Services
- Business Edge Advanced Connectivity Services
- Business Edge Value-Added Services
- Interface Stack
- Service Attributes
- Supported Class-of-Service Models

Business Edge Basic Connectivity Services

The Juniper business edge solution basic connectivity services are the foundation for all other services at the business edge. These services provide basic connectivity between enterprise customer sites and from enterprise customer sites and the Internet. This release of the Business Edge Solution Design Guide covers the following basic connectivity services:

- Layer 3 VPN Service
- Direct Internet Access Service

Layer 3 VPN Service

The Layer 3 VPN service is geared towards Juniper Networks service provider customers that offer private IP connectivity between different enterprise locations. Given wide geographical distribution and the varying capacity requirements of enterprise branch offices and their main office, the Layer 3 VPN services are typically offered via a wide array of physical connectivity types and offer various bandwidth options. These services often come with stringent SLAs and availability targets that must be met by the service provider. While the enterprise is offered a “private network” where its traffic stream never mixes with other carrier customer traffic, the service providers implement a shared IP/MPLS network where a common backbone network offers virtual private network services to their enterprise customers. In addition to the IP connectivity services, carriers often provide additional value-added services like stateful firewalling, NAT, and cloud data center connectivity to the enterprise.

Figure 2 shows the Layer 3 VPN base service topology.

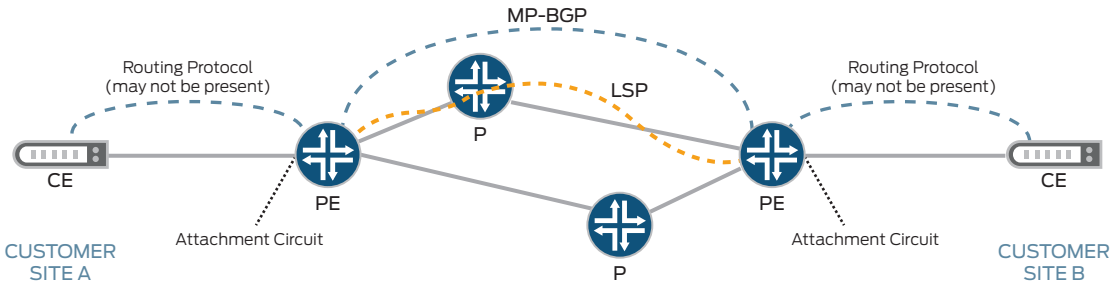

Figure 2: Layer 3 VPN service

A virtual private network (VPN) provides secure access to the enterprise for users and enterprise remote sites over a public network infrastructure. VPN services can support user bases that range in size from a large enterprise office down to individual users working from a home office. A Layer 3 VPN service is a managed virtual connection that connects business sites to their HQ and data centers as well as to other remote offices. In the business edge solution, Layer 3 VPN service is implemented using BGP/MPLS IP VPNs (RFC 4364).

Though Layer 3 VPN service is provided directly to the business customers, it is also used as a base network configuration to support other services. For example, the multicast VPN (MVPN) advanced connectivity service functions on top of the Layer 3 VPN service, as do many of the other value-added services described in this guide.

The deployment scenarios outlined in this solution include Layer 3 any-to-any VPN as well as hub-and-spoke VPN topologies. The solution is designed to support both IPv4 and IPv6. While the solution focuses primarily on customer attachment circuits utilizing Layer 3 VPN, the underlying backbone is built upon the existing MPLS core architecture found in many service provider core networks. MPLS is extensible and flexible, allowing the service provider to transport various services (business, residential, or mobile) over the same backbone in a manner that makes the multitenant nature of the network transparent to the traffic and applications transported by the backbone.

Direct Internet Access Service

Direct Internet Access (DIA) service is used to provide connectivity between enterprise customer sites and the Internet. The service architecture is similar to the architecture of the Layer 3 VPN service and utilizes the same building blocks with an addition of the AS boundary router.

Figure 3 shows the DIA base service topology.

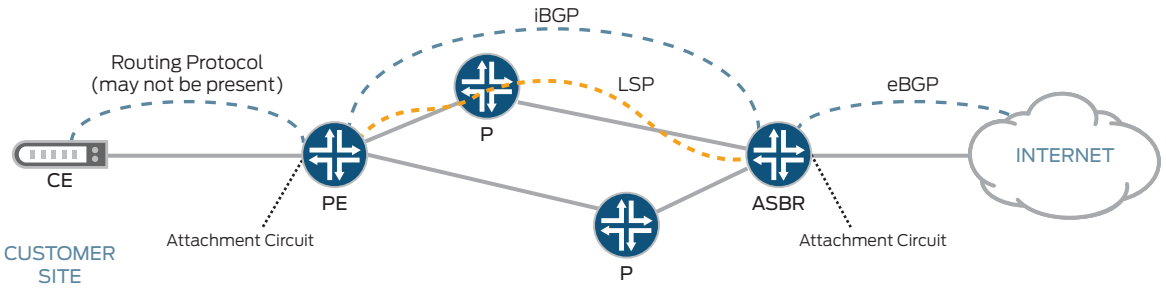

Figure 3: Direct Internet Access service

Business Edge Advanced Connectivity Services

Advanced connectivity services include configuration and service elements that rely on the underlying Layer 3 VPN and DIA configuration. Services in this category are often sold as value-added services to enterprise customers and represent a key area where the service provider can increase average margin per customer. An enterprise customer that purchases the base services to provide business connectivity to all of its locations might seek to add services such as security, video replication, and hosting to the base package. These areas represent additional revenue to the service provider and must be provided in such a way that the existing SLAs can be met by the new services. Resiliency, performance, and availability are as important to value-added services as they are to the base connectivity services.

One of the major challenges facing service providers is being able to offer the diversity and breadth of different services in a competitive marketplace. To remain competitive, service providers use multicast VPNs to deploy multicast service in an existing VPN or as part of a transport infrastructure. Essentially, multicast data is transmitted between private networks over a VPN infrastructure by encapsulating the original multicast packets. As expected, in networks running a multicast VPN service, traffic sources send multicast traffic to the network, while multicast receivers consume that traffic.

This solution outlines configuration of the next-generation multicast VPN (NG MVPN) service where multicast traffic is delivered within a customer VPN, including traffic sourced from extranets. This service is valuable to customers that provide or consume any sort of media (video, music, or voice announcements) over the WAN. NG MVPN provides for efficient transmission and replication of multicast streams across service provider infrastructure, enabling higher quality content presentation as well as bandwidth savings due to efficient traffic replication within the network core.

Figure 4 shows the next-generation MVPN service topology.

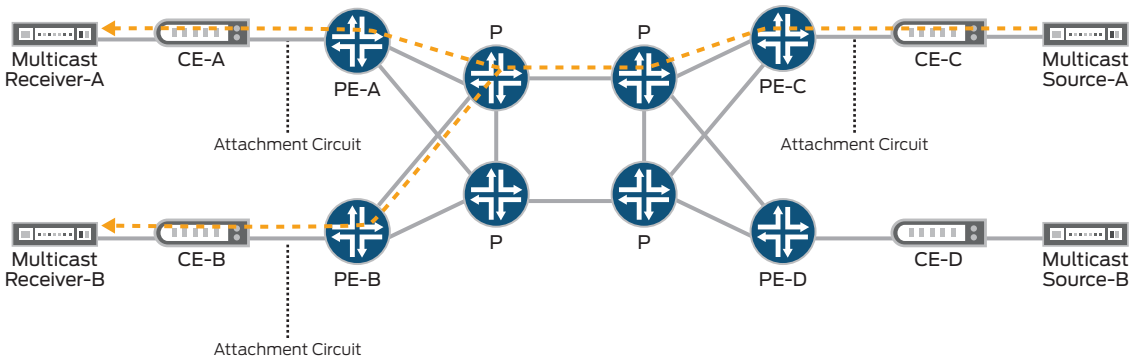

Figure 4: NG MVPN service

Multicast traffic sources and receivers reside behind CE devices, sending and consuming multicast traffic to and from the network. The separation is functional. Receivers and sources can be combined with customer edge (CE) devices (Figure 4). The functional merge usually results in a different set of protocols running at the attachment circuit (IGMP instead of PIM).

Figure 5 shows a next-generation MVPN service topology with both CE source and receiver support.

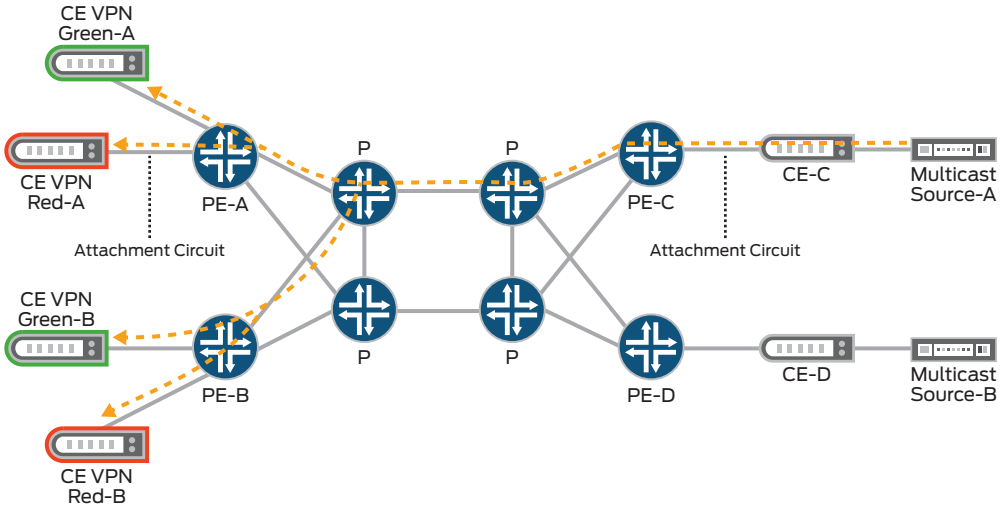

Figure 5: NG MVPN service (CE source and receiver support)

This document describes the use of next-generation multicast VPN as the advanced connectivity service for this solution. The MVPN service provides a multicast traffic distribution option that functions over a base Layer 3 VPN service. The MVPN service can be provided to business customers as a business edge application or used by the service provider as an infrastructure edge application.

Business Edge Value-Added Services

The service provider builds the basic connectivity services as a foundation for the creation and hosting of advanced services to the business customer. This service footprint can include services as basic as firewall and NAT service, and as advanced as content hosting and caching, application acceleration, and hosted virtualized applications and services. The Juniper Networks business edge solution provides security services to enhance the basic service offerings by providing the following:

- Internet access from the Layer 3 VPN network to the Internet using NAT
- Firewall service between Layer 3 VPN sites and between Layer 3 VPN sites and the Internet
- A combination of both firewall and NAT services

Service providers often provide NAT services to enable systems within a business network domain to have direct access to hosts on the Internet. Using a NAT service, the router translates the source addresses of private hosts within the business network into public IP addresses from a range of addresses reserved for the business. The router then maps the private host addresses to the public IP addresses in a table before forwarding the data on to the Internet. When the router receives replies from a remote host, the destination address is translated back to the private business host address using the mapping set up prior to sending the outbound traffic.

Figure 6 shows a service architecture where both firewall and NAT are enabled between Layer 3 VPN connected sites and the Internet. In this service, the firewall and NAT functions are designed to protect the enterprise customer from Internet-based threats, providing secure Internet access. Intersite traffic is not affected by this service combination.

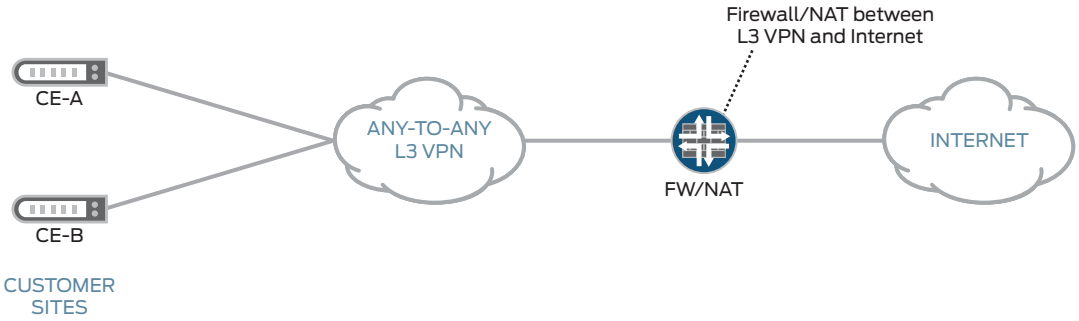

Figure 6: Basic firewall/NAT service topology

The firewall service also enables the enterprise to elevate the security and control of traffic between its own sites. Figure 7 shows how the Juniper Networks business edge firewall service enables the business customer to handle elevated security and regulatory requirements between various site types.

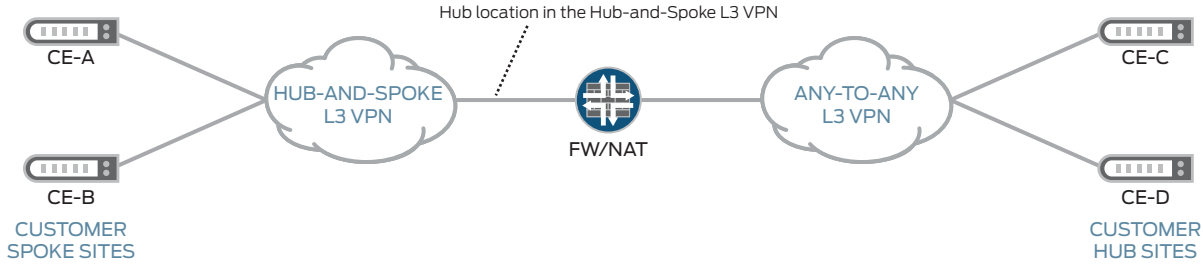

Figure 7: Firewall service topology providing varying security to hub-and-spoke sites

The above topology includes the following components:

- Spoke sites—These are branch locations that are typically not highly secured. Because of lower security requirements, firewall services are often activated between individual spoke sites and between spoke sites and hub sites.
- Hub sites—These are highly secured, centrally located sites. Traffic forwarded between hub sites typically does not require additional firewall processing. Medium-sized and large-sized enterprise customers often have two or more hub sites (for example, HQ and data center).

Finally, the firewall or VPN and NAT services can be combined. In this deployment scenario, the Layer 3 VPN-to-Layer 3 VPN firewall and the Layer 3 VPN-to-Internet firewall services are both enabled for the business customer, securing intersite traffic, Internet traffic, and providing NAT services to all Internet traffic. Figure 8 shows both the firewall and NAT services implemented simultaneously.

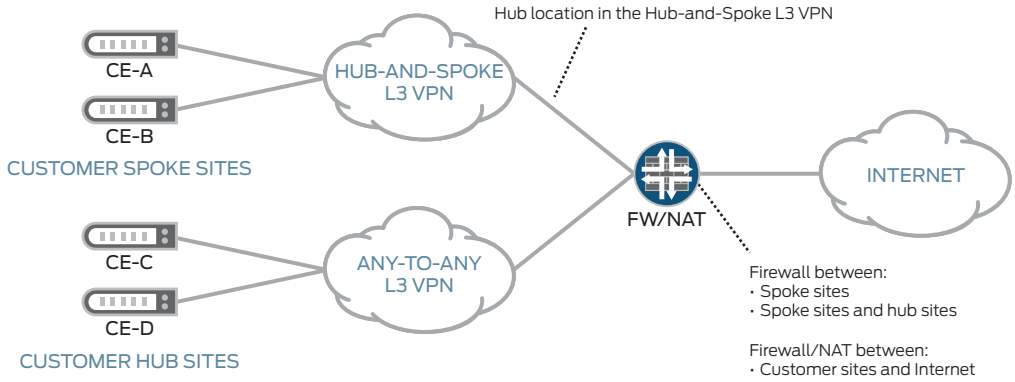

Figure 8: Firewall service topology (simultaneous firewall and NAT)

Interface Stack

Figure 9 shows the interface stack supported in this business edge network solution. The IPv4 and IPv6 customer interfaces are configured in one of two ways: either directly on a VLAN that is stacked on an aggregated Ethernet interface, or on a VLAN that is stacked on a pseudowire that is then encapsulated on another VLAN and stacked on a standard Ethernet interface.

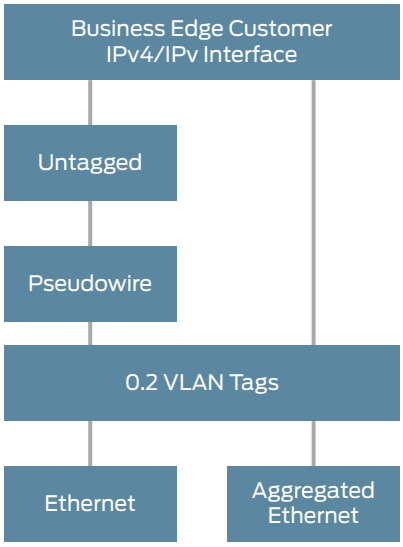

Figure 9: Supported interface stack

Service Attributes

The following sections provide different attribute values for the services described in this guide:

- Common Service Attributes
- Layer 3 VPN Service Attributes
- Direct Internet Access Service Attributes
- Multicast VPN Service Attributes
- Firewall Service Attributes

Common Service Attributes

Table 1 shows the firewall service attributes and their supported values.

Table 1 : Common Service Attributes

Service Attribute	Scope	Description	Supported Values
Attributes that allow for a mix of values:			
Number of ECMP RSVP-TE LSPs	Service with RSVP-TE core transport signaling	Number of ECMP LSPs in the event RSVP-TE is used in the core	16
Type	Attachment circuit	Type of attachment circuit	Physical interface, VLAN/Stacked VLAN, tunnel (pseudowire)
PE router redundancy model	Service	Supported topologies for PE router redundancy	CE dual-homing, MPLS access with PE device redundancy, Virtual Chassis with CE device dual-homing
Network core interface types	Service	Mix of interfaces supported	GbE, 10GbE, 100GbE, Aggregated Ethernet
N access links	Aggregated Ethernet attachment circuit	Number of links in an aggregated bundle attachment	1 through 16
N core links	Aggregated Ethernet core interface	Number of links in an aggregated bundle attachment	1 through 16
Layer 2 OAM	Attachment circuit	Layer 2 OAM mechanisms between PE and CE devices	802.1ag CFM
MPLS access service signaling	Pseudowire attachment circuit	Signaling type for the pseudowires in the MPLS access segment	Targeted LDP
Class-of-service profiles	Attachment circuit	Class-of-service profiles supported by the solution	See Supported Class-of-Service Models

Layer 3 VPN Service Attributes

Table 2 shows the firewall service attributes and their supported values.

NOTE: All attributes allow for a mix of values.

Table 2: Layer 3 VPN Service Attributes

Service Attribute	Scope	Description	Supported Values
Topology	Service	Topology of the customer route distribution between PE routers.	Any-to-any, hub-and-spoke
CE device type	Attachment circuit	Host or Layer 3 router	Router, host
Routing protocol	Attachment circuit	Protocol operating between the CE and PE devices	Static (no protocol), RIPv2, RIPng, OSPF, OSPFv3, BGP
Liveness detection	Attachment circuit	Liveness detection mechanism between the PE and CE devices	BFD

Direct Internet Access Service Attributes

Table 3 shows the firewall service attributes and their supported values.

NOTE: All attributes allow for a mix of values.

Table 3: DIA Service Attributes

Service Attribute	Scope	Description	Supported Values
Routed protocol	Service	Layer 3 routed protocol	IPv4 or IPv6
CE device type	Attachment circuit	Host or L3 router	Router, host
Routing protocol	Attachment circuit	Protocol running between the CE and PE devices	Static (no protocol), BGP
Liveness detection	Attachment circuit	Liveness detection mechanism between the PE and CE devices	BFD

Multicast VPN Service Attributes

Table 4 shows the firewall service attributes and their supported values.

NOTE: All attributes allow for a mix of values.

Table 4: Multicast VPN Service Attributes

Service Attribute	Scope	Description	Supported Values
Routed protocol	Service	IPv4 or IPv6	IPv4, IPv6
Multicast distribution mode	Service	Method by which multicast traffic flows are identified	• Source-specific multicast—both multicast source address and group address • Any source multicast—specifying group addresses only
Functional role	Attachment circuit	Device role	• None (no multicast) • Receiver • Source • Both—a receiver for some groups/ sources and a source for other groups
CE device type	Attachment circuit	Host or L3 router	Router, host
Router membership signaling	Router attachment circuit	Protocols that routers use to signal membership	PIM-SM, PIM-SSM
Rendezvous point location	Any source multicast service with PIM-SM attachment circuits	Location of the rendezvous point (either the CE device, PE device, or both devices)	• CE device • PE device • Both
Host membership signaling	Host attachment circuit	Protocols receiver hosts use to signal membership	IGMPv2, IGMPv3, MLDv2
Multicast source redundancy	Any source multicast service	Support for multicast source redundancy in any source multicast mode	Yes (2 sources only)
Multicast source redundancy	Source-specific multicast service	Support for multicast source redundancy in source-specific multicast mode	Yes (limited to the ingress multicast PE router failure or ingress PE-CE link failure)
Multicast source location	Service	Location of the multicast sources	• Same VPN • Extranet VPN See Figure 5

Firewall Service Attributes

Table 5 shows the firewall service attributes and their supported values.

NOTE: All attributes allow for a mix of values.

Table 5: Firewall Service Attributes

Service Attribute	Scope	Description	Supported Values
Service types	Service	Value security services/subservices	Internet NAT, Internet firewall, VPN firewall (or any combination)
Allow hub locations	VPN firewall service	Whether specific security policies are supported for certain hub locations in the Layer 3 VPN	Yes
Service combination	Service	Service combinations supported	NAT only, firewall only, NAT plus firewall
Translation type	NAT service	NAT translation types supported by the service	NAT44
Application-layer gateway (ALG) support	Service	ALGs supported	Enterprise class ALGs
Public address allocation policy	NAT service	Method by which public IP addresses are assigned to a business customer	Statically assigned to a business customer
Number of L3 VPN exit points	Service	The number of exit points from a VPN are requested by the customer (to support geo-redundancy)	Any

Service Attribute	Scope	Description	Supported Values
Exit points sharing same address	NAT service	Whether or not exit points can share the same external IP address	No
Security policy customization	Service	Method by which security policies are customized for the business customer needs (predefined templates with parameters configured at the activation phase, flexible self-configuration through a self-management portal, manual customization)	Predefined templates, custom (manual activation)

Supported Class-of-Service Models

The following sections describe the class-of-service (CoS) profiles supported by this solution:

- Single Traffic Class
- Multiple Traffic Classes Over a Single Attachment Circuit
- Multiple Traffic Classes Over Multiple Attachment Circuits

Single Traffic Class

In the simplest service contracts, the carrier treats all customer traffic the same, as best effort, and does not differentiate between different traffic classes. At the same time, the service provider enforces traffic rates in both ingress and egress directions.

The first and the simplest CoS profile is described as follows:

- Traffic from the customer (defined per attachment circuit) is policed (rate-limited) up to the committed information rate (CIR).
- Traffic towards the customer (defined per attachment circuit) is policed (rate-limited) up to the CIR.
- For traffic from the customer, DiffServ code point (DSCP) or traffic class (TC) fields can be reset to 0 or honored.

The second profile, enabling traffic shaping towards the customer, is described as follows:

- Traffic from the customer (defined per attachment circuit) is policed (rate-limited) up to the CIR.
- Traffic towards the customer (defined per attachment circuit) is shaped up to the CIR.
- For traffic from the customer, DSCP or TC fields can be reset to 0 or honored.

All rates in both models are enforced for all traffic, regardless of address family. Because both models are applicable to Layer 3 services, L3 packet overheads are used for CIR enforcement purposes.

Multiple Traffic Classes Over a Single Attachment Circuit

The CoS profiles for multiple traffic classes over a single attachment circuit support differentiated treatment for up to eight traffic classes per attachment circuit. The total transmission rate per circuit is limited in both directions.

In this solution, the following classes are defined:

- Expedited Forwarding—a traffic class with bandwidth and jitter guarantees
- Best Effort—a traffic class without bandwidth and jitter guaranties
- Assured Forwarding—a traffic class with a minimum bandwidth guarantee

This solution supports up to six Assured Forwarding traffic classes. Four of these sub-classes comply with RFC 2597 and correspond to AF1X - AF4x traffic classes. One class is nonstandard and is mapped to class sector 5 (CS5). The remaining traffic class is primarily used for network control traffic and is mapped to class sector 6 (CS6) and class sector 7 (CS7).

This solution supports two traffic profiles. The first profile enables traffic shaping in the customer direction as well as traffic policing from the customer. Normally, the customer equipment is configured to shape traffic towards the service provider; the ingress rate limiting on the service provider side is merely a precaution against misconfiguration on the customer side. In the second profile, traffic shaping of network bound customer traffic is implemented at the provider router (ingress shaping), assuming that no traffic shaping occurs on the customer side. For the ingress direction of the first traffic profile, traffic processing functions as follows:

- Expedited forwarding (EF) traffic from the customer (defined for this class per attachment circuit) is policed (rate-limited) up to the CIR.
- Assured forwarding (AF) traffic that exceeds the committed information rate is marked as discard eligible. Existing marking for AF1X to AF4X classes is honored (discard eligible packets remain discard eligible). In other words, the metering is color aware. The committed rate is set up per class per attachment circuit.
- All best-effort (BE) traffic is treated as discard eligible.

- Traffic exceeding the total allowed ingress rate is discarded. Only eligible traffic is discarded. The assumption is that the sum of all CIRs per AF classes, along with the EF CIR, is equal to less than the total allowed rate. If this is not the case, non-discard eligible traffic might get dropped.
- DSCP or TC fields can be reset to 0 or honored.
- When encapsulating into MPLS, discard eligible traffic is marked.

For the egress direction of the first traffic profile, traffic processing functions as follows:

- A dedicated queue for each traffic class is provided.
- The total egress rate per attachment circuit is configured to not exceed the configured maximum.
- The EF traffic class queue is always serviced first, to maintain jitter guarantees.
- In this queue, the EF traffic is rate-limited up to the allowed CIR (per attachment circuit setting).
- The AF traffic is serviced up to the allowed CIR (per attachment circuit, per class setting).
- Spare bandwidth is shared between AF classes first. If no traffic exists in the AF classes, it is assigned to the BE class.

For the ingress direction of the second traffic profile, traffic processing differs only in the processing of traffic from the customer to the network core. Traffic processing functions as follows:

- A dedicated queue for each traffic class is provided.
- The total ingress rate per attachment circuit is configured to not exceed the configured maximum.
- The EF traffic class queue is always serviced first, to maintain jitter guarantees.
- In this queue, the EF traffic is rate-limited up to the allowed CIR (per attachment circuit setting).
- The AF traffic is serviced up to the allowed CIR (per attachment circuit, per class setting).
- Spare bandwidth is shared between AF classes first. If no traffic exists in the AF classes, it is assigned to the BE class.

Multiple Traffic Classes Over Multiple Attachment Circuits

The CoS profiles for multiple traffic classes over multiple attachment circuits support differentiated treatment for up to eight traffic classes for a group of attachment circuits (interface sets). The semantics for the eight traffic classes are identical to the previous profiles discussed. The primary difference between these profiles and earlier profiles is that instead of specifying individual total allowed ingress or egress traffic rates for individual attachment circuits, the total allowed ingress or egress traffic rates are specified for groups of attachment circuits. In addition, each individual attachment circuit uses certain guaranteed traffic rates in both directions. Similar to the previous definitions, two profiles are described—the first using ingress rate limiting and the second using ingress shaping and queuing.

In the first (ingress rate limiting) profile, ingress traffic from the customer is processed as follows:

- EF traffic from the customer is policed (rate-limited) up to the CIR as defined for this class for each attachment circuit.

  

  NOTE: Each attachment circuit can have its own limit.
- AF traffic that exceeds the CIR is marked as discard eligible. Existing marking for AF1X to AF4X classes (with discard eligible packets remaining discard eligible) is marked. In other words, the metering is color aware. The committed rate is set up for each class on a per-attachment circuit basis.
- All BE traffic is treated as discard eligible.
- All discard eligible traffic that exceeds the total allowed ingress rate (per group of connections) is discarded. The assumption is that the sum of all CIRs per AF classes and the EF CIR is equal to less than the total allowed rate. If this is not the case, non-discard eligible traffic might be dropped.
- DSCP or traffic class fields can be reset to 0 or honored.
- Discard eligible traffic is marked when encapsulating into MPLS.

In the first (ingress rate limiting) profile, egress traffic is processed as follows:

- A dedicated queue is provided for each traffic class on a per-attachment circuit basis.
- The total egress rate per group of attachment circuits is configured to not exceed the configured maximum.
- The EF traffic class queue is always serviced first, to maintain jitter guarantees.
- In this queue, the EF traffic is rate-limited up to the allowed CIR (per attachment circuit setting).
- The AF traffic is serviced up to the allowed CIR (per attachment circuit, per class setting).
- Each attachment circuit is configured to receive the minimum guaranteed share of the bandwidth.
- Spare bandwidth is shared between AF classes first. If no traffic exists in the AF classes, it is assigned to the BE traffic class.

For the ingress direction of the second traffic shaping profile, traffic processing differs only in the processing of traffic from the customer to the network core. Traffic processing functions as follows:

- A dedicated queue is provided for each traffic class on a per-attachment circuit basis.
- The total ingress rate per group of attachment circuits is configured to not exceed the configured maximum.

- The EF traffic class queue is always serviced first, to maintain jitter guarantees.
- In this queue, the EF traffic is rate-limited up to allowed CIR (per attachment circuit setting).
- The AF traffic is serviced up to the allowed CIR (per attachment circuit, per class setting).
- Each attachment circuit is configured to receive the minimum guaranteed share of the bandwidth.
- Spare bandwidth is shared between AF classes first. If no traffic exists in the AF classes, it is assigned to the BE traffic class.

CHAPTER 2 Basic Connectivity Design Considerations

- Basic Connectivity Design Overview
- Service Relationships
- Core Design
- BGP Design
- OAM Considerations
- Load-Balancing Options
- MTU Considerations
- Business Edge High Availability Design Considerations
- Business Edge Class-of-Service Design
- Business Edge Security Design
- Layer 3 VPN-Specific Design Considerations
- Direct Internet Access-Specific Design Considerations
- Components

Basic Connectivity Design Overview

Business Edge Basic Connectivity includes Layer 3 VPN and Direct Internet Access (DIA) services, providing basic connectivity between end-user sites. Both services are usually deployed over the same network infrastructure. Figure 10 shows the overall solution topology for the basic connectivity service.

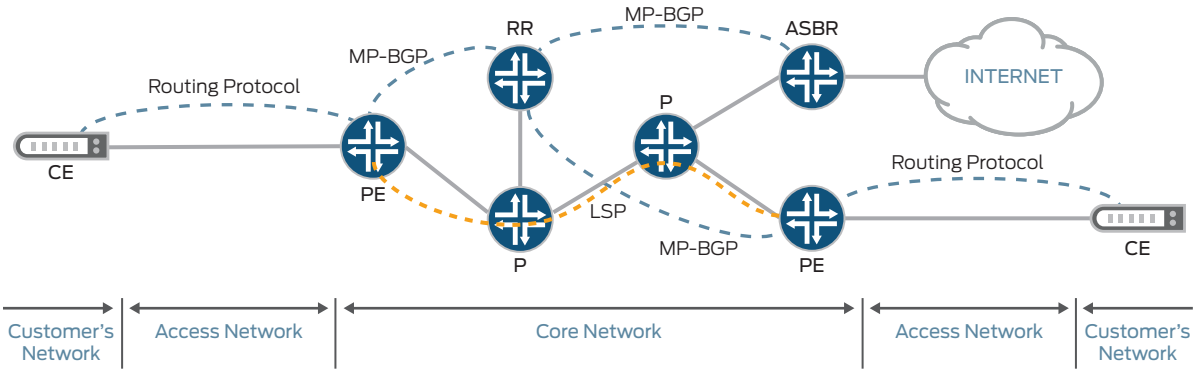

Figure 10: Basic connectivity services solution topology

This connectivity design serves as a base configuration for other services described for this solution (that is, advanced services and value-added services). In other words, when configuring the advanced services or value-added services described in this document, you must first design and implement the basic connectivity network over which these subsequent services function.

You can further subdivide the design of basic connectivity services into IGP, MPLS transport signaling, and BGP design principles. Specific design aspects are covered in separate topics. Layer 3 VPN specific considerations are listed separately. The design also covers CoS, redundancy, and security considerations when building the solution.

NOTE: Access and aggregation network design considerations (that is, configuration of routing and signaling protocols) are beyond the scope of this design, but certain assumptions are made about routing and signaling.

Service Relationships

You can create some services for use directly by customers. However, some services are required for the creation of other services. Figure 11 shows possible relationships between the various services described in this document.

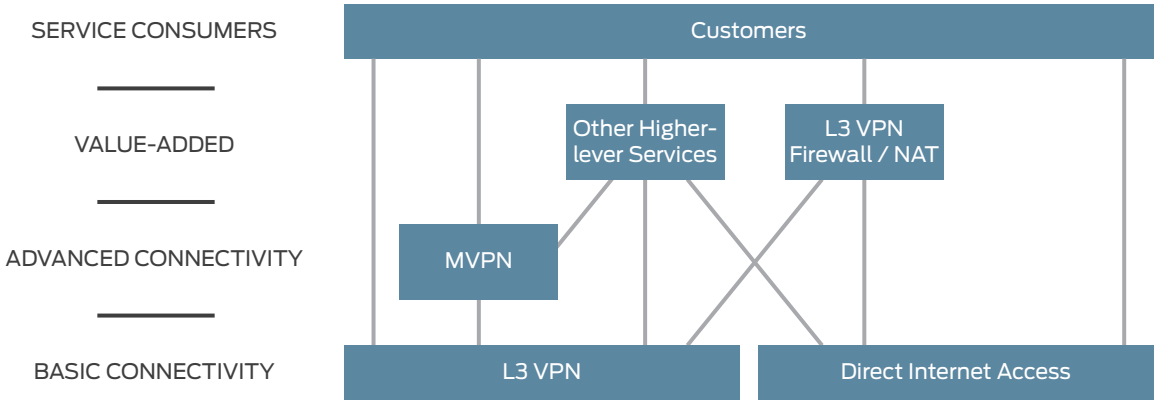

Figure 11: Business Edge 1.0 service relationships

Although the services can be deployed in different ways, the MVPN service described in this document relies on a L3 VPN base configuration. The firewall/NAT services described rely on either an L3 VPN or direct Internet access (DIA) base configuration (if Internet connectivity must be provided from a Layer 3 VPN). The Layer 3 VPN implementation is provided first, followed by a separately described DIA service and then enhancements to base configurations to implement the MVPN and firewall or NAT services.

NOTE: Not all possible configurations have been tested and, in some cases, not all tested configurations have been included in the documentation. However, various subsets have been implemented in tested deployment scenarios.

Core Design

An interior gateway protocol (IGP) exchanges routing information within an autonomous system (AS). This solution supports IS-IS as an IGP protocol. The solution is designed to accommodate nodes in the core network in the order of thousands.

The following sections provide key design considerations when configuring IGP protocols for the network core:

- General Core IGP Design Considerations
- Core Transport Signaling

General Core IGP Design Considerations

IGP protocols for this solution implement the following design considerations:

- In this solution, the IGP design is flat (that is, hierarchical addressing is not used).
- Addressing within the core is purely IPv4. All signaling and routing protocols (for example, LDP and BGP) use IPv4 transport.
- For the IS-IS configuration, all core nodes belong to IS-IS level 2.
- Only extended (wide) metrics are enabled for IS-IS. This configuration is common practice and used to increase the metric range to 16,777,215 ($2^{24} - 1$).

Core Transport Signaling

PE devices must use label-switched paths (LSPs) between them to transmit signaled traffic over the MPLS core. In this solution, we considered the two most common signaling types—LDP and RSVP-TE.

The advantage that RSVP-TE has over LDP is the ability to engineer traffic paths to fully utilize network capacity in environments where traffic patterns change frequently (for example, in large multinational networks that span numerous time zones). However, RSVP-TE requires more state information in the network. In LDP, label-switched paths share state. In RSVP-TE, however, every PE-to-PE label-switched path is signaled separately, resulting in a higher number of RSVP-TE tunnels in the network.

In a full mesh topology between PE devices, the number of parallel paths is calculated using the equation N·(N-1)·M, where N is the number of PE devices and M is the number of parallel paths. The resulting number of paths might present a scaling challenge at the P router level, especially in partial mesh topologies where a few central sites aggregate traffic from multiple PE device locations. For example, using the N·(N-1)·M equation, 200 PE routers and 8 parallel paths results in 318,400 tunnels. If 40 percent of them cross a single P router (the case in a very centralized topology), the result is 127,360 RSVP-TE tunnels per device, which is a value that exceeds the supported capacity of modern routers.

RSVP-TE also utilizes more flexible local repair mechanisms than LDP (for example, facility backup link and node protection), enabling bypass tunnels to follow any route. These repair mechanisms are unlike IS-IS or OSPF loop-free alternates that support only certain topologies that are driven mostly by IGP metrics assignment and have limited support for shared risk link groups.

Some additional key factors resulting in the decision to use RSVP-TE signaling include the following:

- Stability of the traffic patterns—RSVP-TE enables more flexibility in the creation of engineered traffic paths, which is an advantage if traffic patterns change frequently.
- Layer 3 network topology—In certain topologies, LDP (and any underlying IGP loop-free alternate mechanism) might not provide local repair. In heavily meshed networks, scaling might not be an issue for RSVP-TE, giving it an advantage over LDP.

It is important to note that in large service provider networks, LDP and RSVP-TE are often used together. RSVP-TE is used to signal paths between P routers and offers traffic engineering, while LDP is used on the P-to-PE links and also on top of the RSVP-TE tunnels between P routers. This combined solution not only provides better RSVP-TE scale, because fewer P routers are used than PE routers, it also facilitates functional separation. Traffic engineering functions

are provided by P routers, while PE routers provide services to customers and do not attempt to optimize traffic flows through the network core. The only exception to this functional separation is when remote PE routers are connected to the network core using WAN links. If, like the P routers, these PE routers run RSVP-TE, they are considered functionally the same.

Because it is more specific to core network design, the combination of both RSVP-TE and LDP in the core network is currently not part of the scope for this solution. However, the option is mentioned because of the rare case where remote PE routers might run both RSVP-TE and LDP over tunnels.

RSVP-TE Design

RSVP-TE signaling enables the ability to engineer traffic within the network. You can implement traffic engineering using one of the following modes:

- Offline—An external system runs a global optimization algorithm, engineers paths through the network, and configures explicit paths on the network elements.
- Online—Individual network elements optimize paths in a distributed fashion using information from the dynamic traffic engineering database (distributed using IGP extensions). These network elements utilize the Constrained Shortest Path First (CSPF) algorithm for that purpose.

In this solution, traffic engineering is implemented using the online mode method, using PE routers to optimize the paths. RSVP-TE tunnels are configured in a full-mesh topology. This RSVP-TE full mesh topology is provisioned using static configuration. Figure 12 shows the RSVP-TE design topology.

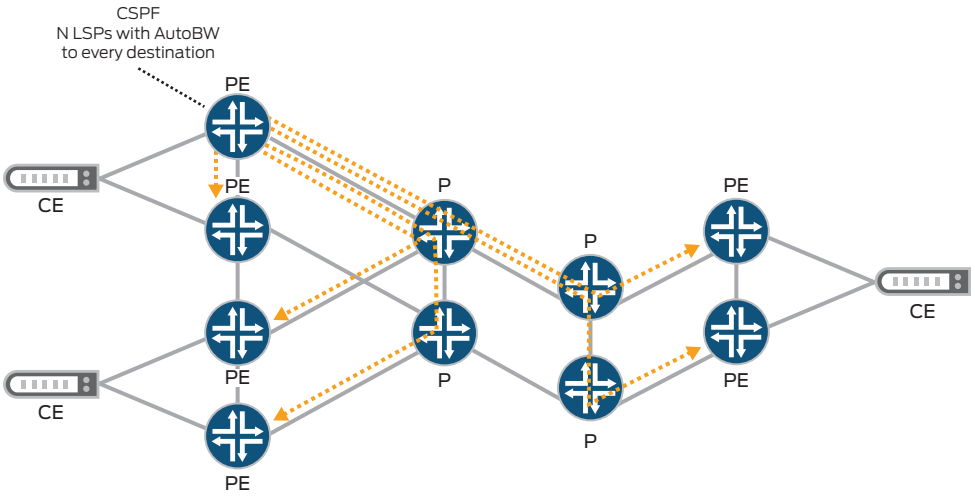

Figure 12: RSVP-TE design topology

Path computation relies on the available link bandwidth, which is verified in the traffic engineering database. Available link bandwidth is communicated through link state updates which, in turn, provide data that populates the traffic engineering database. This results in bandwidth reservation that reflects real bandwidth requirements (or matches them exactly). Therefore, RSVP-TE link bandwidth should be properly provisioned, and requested bandwidth must be configured at the LSP level. The LSP level configuration requirement is difficult to meet in real deployments, because of traffic rate fluctuations (even during daytime hours). This problem is solved by periodic, automatic bandwidth adjustment.

In this configuration, the **adaptive** statement is used to avoid double counting traffic during re-optimization and to implement a make-before-break behavior. Using an LSP bandwidth adjustment of three hours (10,800 seconds) is aggressive, but it is used to enable adaptations to daytime traffic fluctuations.

NOTE: For automatic configuration to function properly, you must enable LSP usage statistics collection.

The number of LSPs traversing a single core link of a large network can easily reach into the thousands. RSVP-TE refresh reduction helps to achieve higher RSVP-TE session scale.

BGP Design

This Business Edge solution does not employ BGP in the core. No provider (P) routers are running BGP (except in a case when they are used in a route reflector role). BGP sessions are established only between provider edge (PE) routers and route reflectors.

NOTE: This solution supports the merging of P router or AS boundary router functions with route reflector functions.

BGP route distribution design within a service provider network is often customer-specific and is typically based on the network topology, proximity, size, and the operational practices defined by each provider. Because these values range widely between provider networks, designing the BGP portion of the network often constitutes establishing BGP sessions between AS boundary router, PE, and route reflector (RR) devices.

Table 6 lists the address families that are enabled in this solution and their purpose.

Table 6: Address Families and Purposes

Address Family	Purpose
IPv4 unicast	Used to exchange IPv4 Internet routes
IPv6 unicast	Used to exchange IPv6 Internet routes
VPNv4 unicast	Used to exchange VPNv4 routes
VPNv6 unicast	Used to exchange VPNv6 routes
Route target	Facilitates better VPN PE router scale. A PE router distributes a list of configured import target communities to its peers. The list is updated automatically as the configuration changes. Peer routers filter VPN BGP advertisements to include only routes tagged with communities of interest.

This solution supports the following three modes of route reflector operation:

- Route reflectors are used only for route distribution; they are not used for forwarding.
- P routers also function as route reflectors.
- AS boundary routers also function as route reflectors.

In the first mode of operation, route reflectors are not running any transport signaling protocols (for example, RSVP-TE or LDP). To force a router to advertise paths to destinations that are not reachable over MPLS, default routes with a discard of next hop are installed into the inet.3 routing table. The same method is used in cases where P routers implement route reflector functionality.

In the last option, the inet.3 table is fully populated with RSVP-TE destinations, resulting in no need to install default discard routes to enable next-hop resolution.

Route reflectors also advertise a default target route in order to accept all VPN routes from peers. The remaining BGP session configuration is default.

OAM Considerations

Operation, Administration, and Maintenance (OAM) mechanisms are used to detect failures in the network and isolate them. Figure 13 shows OAM mechanisms used in this solution and how they are applied to the network and its segments, including the service provider core network, L2 Ethernet network, and MPLS access network.

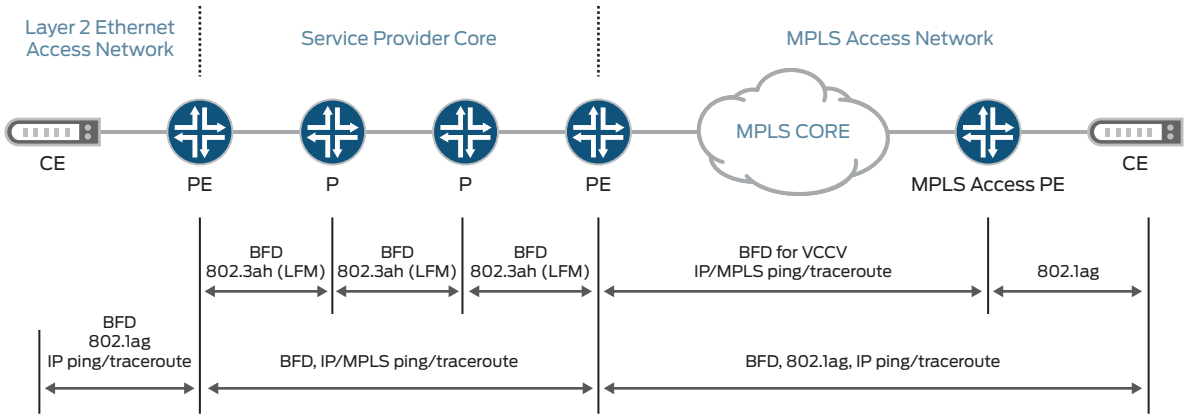

Figure 13: OAM mechanisms for L3 VPN or DIA service

BFD and 802.1ag are used in the service provider core at the link level, as a failure detection mechanism. To detect data plane failures end-to-end between the PE devices, BFD is configured on RSVP-TE LSPs.

Failures can be detected on PE–to-CE links in the following ways:

- Using BFD for routing protocols and static routes
- Configuring 802.1ag connectivity check

Whether or not to deploy BFD, 802.1ag, or both is dependent on the specific network scenario. This solution utilizes both, where 802.1ag is used for connectivity verification and BFD is used for protocol failure detection. When 802.1ag connectivity check fails, the logical interface is declared down and the failure is propagated to the rest of the network.

The MPLS access portion of the network differs from the rest of the network. In this portion, BFD for virtual circuit connectivity verification (VCCV) is used where the access network connects to the PE devices using pseudowires.

To further isolate failures and troubleshoot problems, the following mechanisms are used:

- IPv4 or IPv6 ping and traceroute in both the core and access MPLS network segments and on the PE-to-CE links
- 802.1ag link trace in the access networks (note two-level operation in the MPLS access network)
- Loopback (ping) in the access networks (note two-level operation in the MPLS access network)

Load-Balancing Options

Load balancing is the ability to balance traffic across multiple links, distributing the bandwidth used by each user and increasing the total amount of bandwidth available. Load balancing in this solution is accomplished at the following levels:

- Border Gateway Protocol
- Equal-Cost Multipath Routing
- Aggregated Ethernet Interfaces

Border Gateway Protocol

BGP enables you to load-balance by selecting multiple equal-cost external BGP (EBGP) or internal BGP (IBGP) paths as active paths. In this solution, BGP load balancing is implemented as follows:

- BGP multipath for inet.0
- EBGP and IBGP multipath for VRFs by using the **vpn-unequal-cost** statement at the **[edit routing-instances *routing-instance-name* routing-options multipath]** hierarchy level to apply protocol-independent load balancing to VPN routes that are equal until their interior gateway protocol (IGP) metrics with regard to route selection are taken into consideration; this configuration ignores the router ID when installing routes.

Equal-Cost Multipath Routing

Equal-cost multipath (ECMP) is an L3 mechanism for achieving load balancing for a path with multiple equal-cost next hops. An ECMP set is formed when the routing table contains multiple next-hop addresses for the same destination that have the same preference and metric values. When implementing this configuration, we enable per-flow balancing between equal-cost paths. In this solution, every RSVP-TE LSP represents a path. The creation of 16 LSPs from one router to another results in 16 equal-cost paths.

Aggregated Ethernet Interfaces

This solution supports Ethernet link aggregation to facilitate redundancy and increase link capacity, while still maintaining a single Layer 3 link between systems.

MTU Considerations

A maximum transmission unit (MTU) is the largest size packet or frame that can be sent over a network. Using an MTU size that is too large can result in packet retransmissions if a packet encounters a router that cannot handle a packet of that size. On the other hand, using an MTU size that is too small results in more overhead. In addition, low MTU configuration might not allow deployment of services with higher MTU requirements. For these reasons, the proper MTU size is important when designing your network.

The appropriate size of an MTU is typically based on the following criteria:

- Hardware restrictions
- Service requirements

Most modern hardware supports MTU configuration up to 9,192 bytes. For most IP services, the MTU size requested is 1,500 bytes. In some cases, however, end users might request larger MTUs to facilitate overlay network deployments (for example, IPsec VPNs). In addition, data center interconnect applications might require support for even larger MTUs (for example, 9,000 or more).

To accommodate the possibility of larger MTUs, core interfaces use the largest possible MTU supported by the network platforms (that is, 9,192).

To account for MPLS encapsulation overheads on either the core or edge side, edge interface MTUs are configured with smaller values than the core MTU. The edge MTU setting on the service level should also be consistent between the PE devices.

In this network design, the IP MTU size is 9,100. Table 7 shows the interface-level MTU configuration required for each interface (dependent on the interface overheads).

Table 7: Interface MTU Values

Interface Type	MTU Size
Untagged interfaces	9,118
Single-tagged interfaces	9,122
Dual-tagged interfaces	9,126

Business Edge High Availability Design Considerations

System reliability is often measured in terms of its availability. The latter depends on the sum of failure detection and recovery time and also on the failure rate.

Component redundancy helps to reduce the recovery time, while other techniques (for example, link fault signaling and Bidirectional Forwarding Detection) can help to minimize failure detection time. The following sections address both aspects at the network and device level:

- Network-Level Reliability
- Device-Level Reliability

Network-Level Reliability

Network-level reliability mechanisms target specific failure scenarios that lead to link outages or complete node failures. The following sections describe them in more detail:

- Link Aggregation
- Link-Level Failure Detection
- MPLS Link/Node Protection and Local Repair
- Provider Edge Router Redundancy
- Provider Edge Virtual Chassis Redundancy

Link Aggregation

Link aggregation is a method of combining (aggregating) multiple network connections (links) in parallel to increase throughput beyond that of a single connection and to provide redundancy if even one link fails. In this solution, P-to-PE, P-to-P, and P-to-CE links can form aggregated bundles and all of the links in a bundle remain active. To avoid wiring mistakes, and to assist in monitoring, the Link Aggregation Control Protocol (LACP) is employed. LACP provides the ability to control several physical ports bundled together to form a single logical channel. The protocol is configured in active mode and employs a 1-second periodic transmission of LACP packets. LACP periodic packet transmission can be used as a failure detection mechanism by itself. However, because LACP provides a detection time of 2 to 3 seconds, which is not within the typical 50 ms failure detection target (including recovery), added failure detection mechanisms are required.

Link-Level Failure Detection

IEEE 802.3 defines standard physical layer/link layer mechanisms to detect failures of 10GbE and 100GbE links. Link-layer Link Fault Signaling is often used for 10-Gigabit Ethernet or 100-Gigabit Ethernet interfaces. This form of detection failure is supported by all MX Series router interfaces and is enabled by default. Although it depends largely on the latency between systems, Link Fault Signaling provides very fast failure detection (often fewer than 4 to 6 milliseconds for very short links with low latency), even in unidirectional failure scenarios.

NOTE: The Link-layer Link Fault Signaling failure detection mechanism can be used for most dark fiber links.

Links traversing a transport xWDM/OTN network might require special failure detection mechanisms. This added requirement is sometimes necessary because transport equipment might not propagate local interface failures to the remote end appropriately. This error can result in unidirectional link scenarios that lead to black holes.

The failure detection timers for Bidirectional Forwarding Detection (BFD) have shorter time limits than default failure detection mechanisms, so they provide faster detection. This means that you can use BFD for IGP protocols to alleviate failure detection problems. When configured, the BFD protocol sends hello packets at a specified, regular interval. A neighbor failure is detected and BFD declares an interface to be down when the routing device stops receiving replies

after a specified interval or when a specified number of consecutively lost packets is detected.

In the solution example, BFD is configured with the following:

- A minimum detection interval of 10 milliseconds between packets
- Loss of three consecutive packets treated as a link failure

BFD for IGP protocols is implemented at the IP layer and is subject to load balancing between aggregate links. BFD control packets are transmitted over one link of a bundle. This means that it is not possible to detect failures for all bundle links.

If fast detection failure is required for individual links in a bundle, we recommend the configuration of Ethernet OAM 802.3ah connectivity verification mechanisms. In the event of an 802.3ah session failure, the link can be declared down by use of an action-profile that can be used to remove the next hop and provide a reroute event (if needed).

MPLS Link or Node Protection and Local Repair

Specific to the MPLS signaling protocol in use, link or node protection addresses failures of the P-to-P and P-to-PE links in the MPLS network and is applicable to both access and core segments. This protection mechanism is used to locally reroute traffic around a failed link or node.

When using RSVP-TE, a facility backup, link protection, and node protection are used.

When implementing this solution, we recommended using automatic backup tunnel provisioning with zero bandwidth reservation.

Provider Edge Router Redundancy

Techniques used for provider edge router redundancy depend on the redundancy models. The following sections describe the redundancy solution architecture for two deployment scenarios:

- Customer Edge Dual-Homing Redundancy
- MPLS Access Network Redundancy

Customer Edge Dual-Homing Redundancy

Dual-homing is a network topology in which a device connects to a network using two independent access points (attachment circuits). One access point acts as the primary connection and the other acts as a standby connection that is activated if the primary connection fails. Figure 14 shows a CE device with two connections to the service provider infrastructure. These connections are treated as separate attachment circuits.

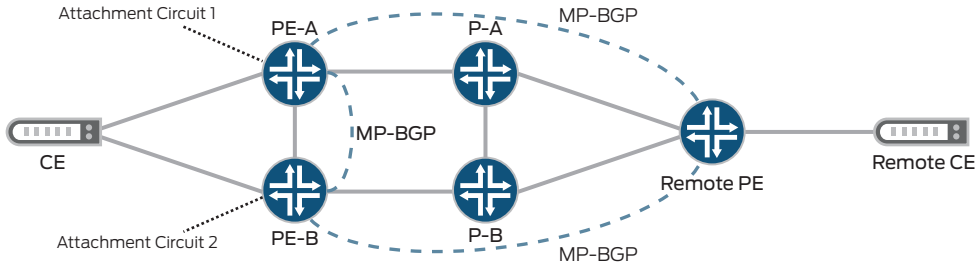

Figure 14: Customer edge dual-homing network configuration

There is no special routing configuration related to redundancy on the PE-CE interface. Routing information is propagated to PE(A) and PE(B), or configured statically on both circuits. When different route distinguishers are used on the PE devices, all CE device prefixes are announced to the rest of the PE devices. In addition, because of EBGP multipath, traffic flows in both directions through both devices. With this configuration, if the PE(A) interface connection fails, the router declares the interface down and begins using the route announced by its peer.

For traffic flowing from the CE device to the PE device, and in the event of a routing protocol failure between those devices, the CE device detects any failures and reroutes traffic to the redundant node. In this case, using the BFD protocol is highly recommended, because it can significantly improve failure detection time, even when only static configuration is used.

MPLS Access Network Redundancy

This configuration supports pseudowire redundancy. Figure 15 shows a single redundant pseudowire configured to transmit traffic from an MPLS access PE device to the other PE devices.

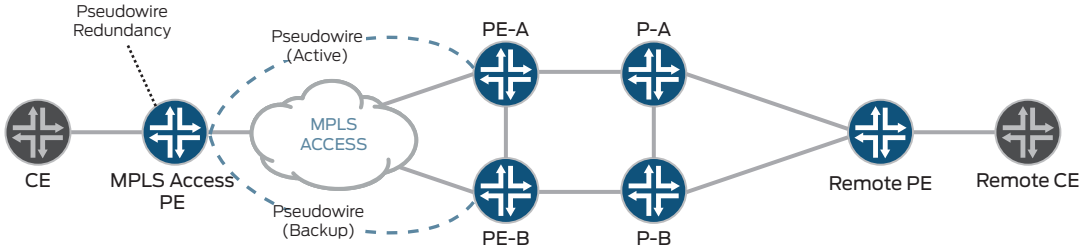

Figure 15: MPLS pseudowire redundancy

NOTE: Depending on the MPLS access PE router, the backup might be presignaled.

Both PE routers contain identical customer interface configurations. When pseudowire switchover occurs, traffic is forwarded through the backup PE device. To accept traffic immediately after the switchover, the backup PE device must either use the same media access control (MAC) address as the primary PE device, or it must ignore the destination MAC address in any datagrams it receives. The latter approach occurs in MX Series Modular Port Concentrator (MPC) implementations. Because of this implementation, MAC addresses might not be synchronized between systems.

The following techniques are configured in this solution to reduce failure detection time, failure recovery time, and to help reduce PE device failure issues:

- Pseudowire redundancy presignaled backup
- BFD protocol configuration for VCCV
- BFD configuration for LSPs between PE routers

Provider Edge Virtual Chassis Redundancy

MX Series 3D Universal Edge Routers support a Virtual Chassis configuration for stateful interchassis redundancy. A Virtual Chassis configuration enables a collection of member routers to function as a single virtual router and extends the features available on a single router to member routers in the Virtual Chassis. The interconnected member routers in a Virtual Chassis are managed as a single network element that appears as a single chassis with additional line-card slots. The Virtual Chassis also appears to the access network as a single system. Figure 16 shows a scenario with a CE router dual-homed to two different PE routers that are combined into a Virtual Chassis configuration.

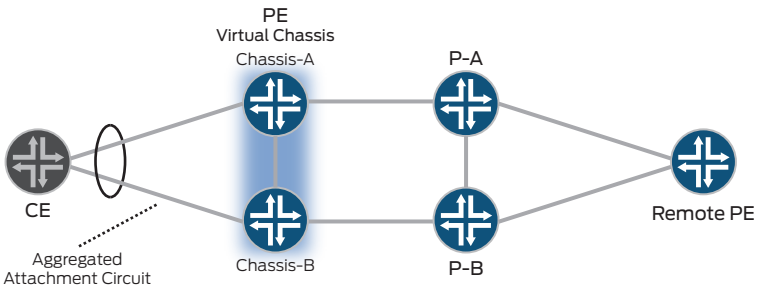

Figure 16: Virtual Chassis model

This model contains one Layer 2 link to each PE router. In this case, however, each link forms a single aggregated Ethernet attachment circuit. Adding another layer of redundancy protection, this model further combines the two PE routers using the Virtual Chassis configuration.

The topology contains direct links between the local PE devices (Chassis-A and Chassis-B). These links are used for Virtual Chassis control communication as well as for traffic forwarding. Traffic links exist between PE Chassis A and router P-A and between PE Chassis B and router P-B. The topology does not contain direct links between PE Chassis A and router P-B or between PE Chassis B and router P-A.

In this topology, if one PE chassis fails, it is treated as a multiple line-card failure. Once a chassis failure is detected, the other chassis continues to forward traffic.

Device-Level Reliability

At the device level, the following reliability mechanisms should be used:

- Chassis Hardware Component Redundancy
- Graceful Routing Engine Switchover
- Nonstop Routing

Chassis Hardware Component Redundancy

Whenever possible, you should implement a comprehensive strategy for chassis hardware component redundancy. This redundancy should include power supplies, switch fabrics, and Routing Engines where multiple, redundant hardware components are installed in the chassis to ensure consistent and reliable device uptime or faster network reconvergence.

Graceful Routing Engine Switchover

Graceful Routing Engine switchover (GRES) enables a routing platform with redundant Routing Engines to continue forwarding packets, even if one Routing Engine fails. GRES switchover preserves interface and kernel information. Traffic is not interrupted. However, GRES does not preserve the control plane state. Neighboring routers detect that the router has experienced a restart and react to the event in a manner prescribed by individual routing protocol specifications. To preserve routing during a switchover, GRES must be combined with either graceful restart protocol extensions or nonstop active routing.

Nonstop Active Routing

Nonstop active routing (NSR) enables a routing platform with redundant Routing Engines to switch from a primary Routing Engine to a backup Routing Engine without alerting peer nodes that a change has occurred. NSR uses the same infrastructure as GRES to preserve interface and kernel information. However, NSR also preserves routing information and protocol sessions by running the routing protocol process (RDP) on both Routing Engines. In addition, NSR preserves TCP connections maintained in the kernel.

Business Edge Class-of-Service Design

VPN services are usually the most complex in terms of class-of-service (CoS) design. For this reason, this section targets CoS design for VPN services specifically. Other services (for example, DIA) might reuse a subset of this design.

NOTE: For specific DIA design considerations, see “Direct Internet Access-Specific Design Considerations.”

The following sections provide detailed CoS design considerations for the Layer 3 VPN service configuration:

- Class-of-Service Design Overview
- Class-of-Service Design for P and PE Routers
- Forwarding Class Design Considerations
- Policing Considerations
- Aggregated Ethernet Policing Considerations
- Class-of-Service and the Direct Internet Access Service

Class-of-Service Design Overview

CoS design for VPN services follows models defined in RFC 2474 “Architecture for Differentiated Services” and RFC 3270 “Multi-Protocol Label Switching (MPLS) Support of Differentiated Services.” Figure 17 provides an example design topology for deploying CoS for VPN services.

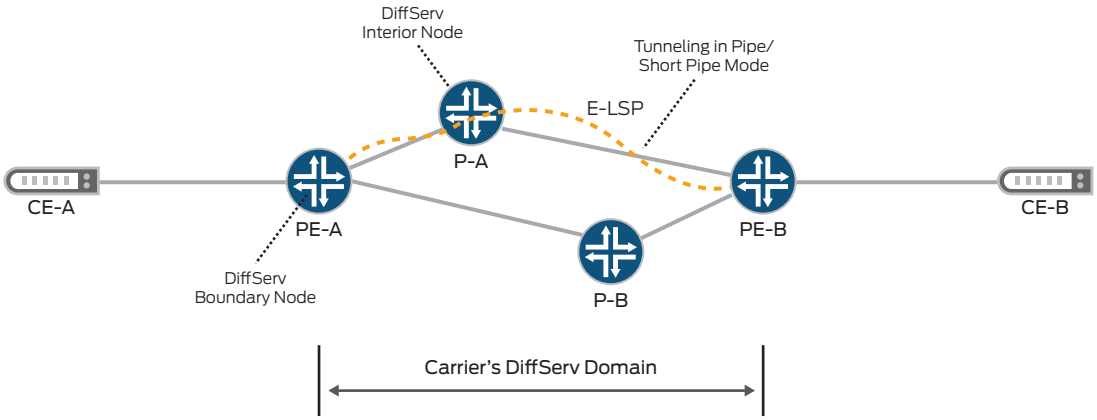

Figure 17: Business edge CoS design topology for VPN services

For the traffic flow from CE-A to CE-B, the devices have the following roles:

- CE-A—Marks traffic destined to the carrier network. This device performs egress shaping (in accordance with a traffic conditioning agreement [TCA]). If shaping is not available or not configured on the CE device, this function can be delegated to device PE-A.
- PE-A—Acts as a differential services boundary node for the carrier differential services domain. This device receives the traffic, marks it (if not done by the CE-A device), conditions the traffic in accordance with the TCA, and maps the traffic to the behavior aggregates of the carrier differential services domain. PE-A marks MPLS-encapsulated traffic with the appropriate EXP traffic class (EXP/TC) values. If the customer marking is trusted, it remains unchanged through the transmission over the carrier differential services domain. All carrier behavior aggregates are transmitted to device PE-B using an EXP-inferred packet scheduling class (E-LSP).
- P-A—Acts as a differential services interior node. The functions of this device are limited to traffic classification and respective per-hop behavior implementation (for example, queuing or shaping).
- PE-B—Receives traffic from the P routers, conditions the traffic, and performs any queuing or shaping of the traffic upon egress. Any applied traffic classification principles depend on the agreement between the customer and the carrier. Per-hop behavior can be inferred from the EXP/TC marking of the carrier differentiated services domain (called a ***pipe model***) or the customer marking (called a ***short pipe model***).

In this design, pipe or short pipe tunneling modes are used because they are better suited to support DSCP transparency.

Class-of-Service Design for P and PE Routers

This section outlines a design that supports certain customer CoS profiles as defined in Supported Class-of-Service Models.

Figure 18 shows the traffic processing flow for PE or P devices.

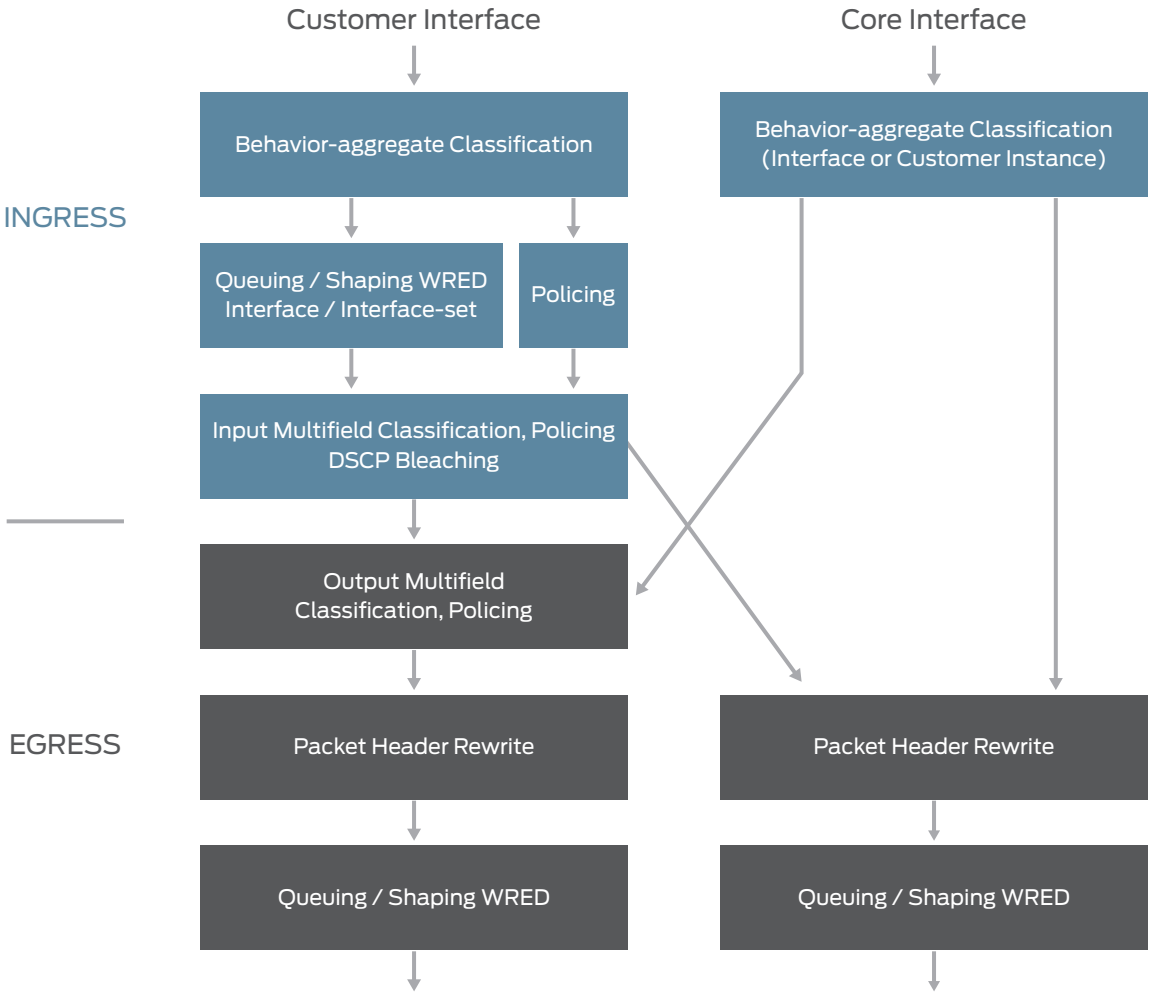

Figure 18: P and PE router traffic processing flow

The left column lists the various traffic processing steps at the customer (PE-CE) interface. The right column shows processing at the core (PE-P) interface. The arrows illustrate the traffic flows between the different processing steps as well as between the external systems. The processing is further divided into ingress and egress actions.

Ingress traffic processing at the customer interface is performed as follows:

1. Traffic is processed through a behavior aggregate (BA) classifier.

   NOTE: This form of classification is useful when subscribers provide the correct DSCP marking. If this marking is not provided, this classification is skipped.

2. Depending on the customer agreement, additional steps might include ingress queuing and shaping or policing.
3. Multifield classification, traffic conditioning, and DSCP bleaching (if required) are performed.

   Not every customer contract requires multifield classification. However, this form of classification is the most flexible method of implementing customer policies. Due to this flexibility, carriers might choose to implement multifield classification, even if the same goal could be achieved by other means (for example, using interface policing or behavior aggregate classification).

Following this process, an MPLS core forwarding class (DiffServ Packet Scheduling Class) and a loss priority are assigned to the customer traffic.

Egress traffic processing at the customer interface is performed as follows:

1. Output multifield classification and conditioning is performed (if needed).
2. In general, there is no direct one-to-one relationship between customer behavior aggregates and core behavior aggregates. Because of this disconnect, it is necessary to determine the customer behavior aggregate to ensure proper treatment at the PE-to-CE link. This determination is one of the features of the short pipe model.
3. A packet header rewrite is performed.
4. In most cases, only lower level packet headers are modified. For example, for a Layer 3 VPN service, the DSCP field remains unchanged (to maintain DSCP transparency). However, the EXP/TC bits (in MPLS access networks) are modified.
5. Interface-level queuing and shaping is performed along with any congestion avoidance processing (weighted early random detection).

Core interface traffic processing is performed in much the same way. At ingress, traffic runs through a BA classification. As a result, the core forwarding class is determined.

At the egress of the core interface, packet header rewrite occurs, followed by a queuing and shaping and congestion avoidance step.

Forwarding Class Design Considerations

This solution supports up to eight customer behavior aggregates (or *classes of service*). Service providers often limit the number of classes in the core to reduce configuration complexity. To decrease this management overhead, the number of classes typically drops to three or four in the network core. Because core networks are often overprovisioned, and BA bandwidth limits are only temporarily reached during rerouting or local repair events, the boundaries between them are unnecessary most of the time. It is logical, therefore, to group (aggregate) traffic classes with similar bandwidth guarantees.

In this configuration design, forwarding semantics are determined by the queuing and traffic control profile configuration. This configuration is implemented at the attachment circuit level and is specific to each circuit. This method of configuration enables greater flexibility in how forwarding semantics are used. Where one circuit might implement a set of semantics to transmit eight traffic classes, another circuit might implement a different set of semantics to transmit fewer traffic classes. In this design, a total of eight forwarding classes are used. Four aggregate forwarding classes are configured in the network core, whereas eight classes are configured at the customer interfaces. Because forwarding class treatment is local to the egress interface, it is possible that edge classes and core classes will overlap.

Ultimately, forwarding classes are mapped to specific interface queues that are identified by a specific queue number. Historically, some of these queues have received special treatment in Junos OS and this treatment should be taken into consideration during the CoS design process. Table 8 provides proposed class-to-queue mappings.

Table 8: Forwarding Class and Queue Number Mappings

Customer Class	Core Class	Queue Number
Best effort (BE) or default	Best effort (BE) or default	0
Expedited forwarding (EF)	Expedited forwarding (EF)	1
Assured forwarding 1 (AF1x)	Assured forwarding (AF)	2
Assured forwarding extra class 2 + management	Network control	3
Assured forwarding 2 (AF2x)		4
Assured forwarding 3 (AF3x)		5
Assured forwarding 4 (AF4x)		6
Assured forwarding extra class 1		7

Queue 0 is usually used for best-effort traffic and for host outbound traffic (for example, BGP, with BGP retransmissions going into queue 3). In order to maintain uniform control over the management traffic bandwidth allocation, direct all host outbound traffic to queue 3.

Policing Considerations

This solution targets deployments that use dual-stack interfaces (both IPv4 and IPv6). Normally, customer CoS profiles do not differentiate between protocol families in a particular behavior aggregate; therefore, logical interface policers have to be used instead of policers applied to a specific family.

This solution design incorporates the following policer types:

- Hierarchical ingress policers (to support customers with more than one traffic class)
- Color-aware, two-rate, three-color policers (to support AF traffic policing)
- Two-rate, two-color (regular) policers (for Layer 3 VPN customers with one traffic class and most DIA customers)

MX Series routers support the following two policer types:

- Layer 2 policers, which take into account L2 packet sizes, are applied to the logical interface.
- Layer 3 policers, which include L3 packet sizes, can be applied to logical interfaces and used in filters.

In this design, Layer 3 policers are used.

Aggregated Ethernet Policing Considerations

Most high-end routing platforms implement a distributed traffic processing model with multiple Packet Forwarding Engines (PFEs) processing traffic at the line card level, the interface card level, or both. The fact that aggregated links can be terminated at different PFEs poses a challenge to operations (for example, policing or queuing/shaping that must maintain their state). This state is not synchronized between PFEs. Instead, a policer applied to the aggregated Ethernet interface is programmed into all PFEs that terminate bundle links.

You can use one of the following two methods to program policer parameters:

- Program the same configured rate into all PFEs. As a result, more traffic might be accepted.
- Program the rate in proportion to the relative capacity of individual member links terminated at the PFE. For example, in a group of four 10GbE interfaces, where one is terminated at one PFE and three at another PFE, set the rates to 25 percent of the configured rate on the first PFE and 75 percent on the second PFE. This is known as the ***shared bandwidth method***.

In this design, all policers applied to aggregate interfaces use the shared bandwidth method.

Class-of-Service and the Direct Internet Access Service

In this solution, DIA service traffic is delivered in a global routing table context. This form of delivery does have certain implications to the CoS design, because of the default penultimate hop-popping behavior of MPLS LSPs. Traffic entering the egress PE router has no label and its packet scheduling class within the carrier differential service domain is lost, meaning that this operation is not compatible with the pipe model. There are two options used to address this problem:

- Use the short-pipe model instead of the pipe model
- Disable penultimate hop-popping by announcing explicit null labels from the egress PE device

The use of explicit null labels might result in certain interoperability issues when the same PE device is used for VPN services. The interoperability issues occur because the explicit null label push of semantics at the P router level might differ between certain implementations. Because of these issues, this design uses the short-pipe model for implementation. In this model, the egress PE device uses the DSCP or type of service (ToS) fields of the IP packet to apply the appropriate forwarding treatment, requiring consistent DSCP or ToS marking semantics between ingress sites.

Business Edge Security Design

Networks are subject to attacks from various malicious sources. These attacks can be passive, where a network intruder intercepts data traveling through the network (for example, wiretapping), or active, where an intruder initiates commands to disrupt the normal operation of the network (for example, denial-of-service attacks or address spoofing). Security for this network design involves preventing and monitoring unauthorized access, network misuse, unauthorized network modification, or attacks that result in the denial of network services or network accessible resources.

This section is not a complete discussion regarding network security threats and mitigation options. It does not list all potential threats and mitigation techniques. However, the following sections do focus on security techniques that are specific to securing the business edge solution:

- Protecting Infrastructure Elements
- Layer 3 VPN Service Security Considerations
- Direct Internet Access Service Security Considerations

Protecting Infrastructure Elements

The service provider infrastructure is the most obvious attack target for any services described in this solution. As such, it is imperative that you take measures to protect infrastructure elements. The design elements described in this section are not service-specific, but apply to all business edge services. The following threats are applicable to the infrastructure elements:

- Unauthorized access, where an attacker might attempt to gain access to network infrastructure elements with the purpose of reconfiguring them. This action might be a part a larger attack (for example, attacks involving traffic redirection or malicious traffic injection).
- Software and hardware security flaws (for example, vulnerabilities in the TCP stack implementation that enable an attacker to arbitrarily terminate BGP sessions).
- Hijacking of various management and network control communications, where infrastructure elements that are often running various control management routines and protocols are accessed by malicious users. By establishing new control and management sessions, or intercepting existing ones, these malicious users might impact forwarding behavior.
- Distributed denial of service (DDoS), where there is an attempt to deny valid users access to network or server resources by using up all of the resources of the network element or server. DDoS attacks involve an attack from multiple sources, enabling a much greater amount of traffic to attack the network.

Protecting Against Unauthorized Access

Unauthorized access and other software and hardware security flaws can be mitigated by implementing the following:

- Limit management access to various network elements. These limitations include physical access as well as network access (for example, loopback filter configuration, TACACS+ or RADIUS authorization, and allow or deny expressions for certain user groups).
- Disable all control protocols that are not in use.
- When it comes to mitigating hijacking threats, you can configure authentication for most control protocols used in the network (for example, LDP, IS-IS, OSPF, BGP, or RSVP-TE).
- Implement secure communication for management access. The secure communication can be in the form of the following:
  - SSH—Devices in this solution support the SSH protocol as a secure alternative to Telnet for system administration.
  - Secure Copy Protocol (SCP)—Based on the SSH protocol, SCP file transfer securely and reliably transfers files between a local host and a remote host or between two remote hosts.
- Hide infrastructure elements from the end user. For example, at the PE router level, you can hide P routers by disabling time-to-live (TTL) propagation from the IP packet header into the MPLS shim header.

Protecting Against Denial of Service Threats

The most complex security threats to manage are denial-of-service (DoS) attacks that target the control plane of a network element. For VPN or DIA services, the PE-to-CE interface connection is the most vulnerable location for these attacks. For example, an excessive rate of legitimate host-bound packets coming from a user can affect the processing of other user requests and eventually result in the tearing down of control or routing protocol sessions that ultimately lead to traffic loss.

MX Series routers have a hierarchy of built-in (and sometimes nonconfigurable) control plane protection mechanisms that operate at different levels. These protection mechanisms are contained in the Routing Engine, the line-card host processor, and the network processor. In general, it is relatively difficult to launch an attack from a single end user, served from one line card, which would affect a user served from a different line card. Because of this attack limitation, the easiest way to reduce potential security damage is to isolate failure domains. You can accomplish this isolation by configuring PE–to-P interfaces to dedicated line cards and by distributing users between several line cards.

NOTE: For cost reasons, this isolation and distribution approach might not be possible in many deployments.

Unfortunately, even when isolation and distribution are implemented, DoS attacks within the failure domain are not blocked. The only way to mitigate these attacks is to police control plane traffic from each customer individually as follows:

- Enable individual, per-interface Address Resolution Protocol (ARP) policers
- Enable individual, per-interface policers for other control plane traffic (such as OSPF, BGP, BFD, and Ethernet OAM)

This added, individual configuration results in added overhead and is reflected in the individual attachment circuit routing and OAM protocol configuration. Assuming that the risk of a large-scale DoS attack initiating from an enterprise customer network is very low, service providers often choose not to implement complex, individual policing of customer control plane traffic.

This class-of-service design follows the same assumption, choosing not to implement custom policers for individual attachment circuits.

Layer 3 VPN Service Security Considerations

In a Layer 3 VPN service, enterprise WAN routing functions are outsourced to the service provider. In addition, some of the security functions might also be outsourced, especially when a PE router serves as a Layer 3 gateway for CE host devices. In this configuration design, we configure unicast reverse path forwarding (RPF) check in strict mode on the PE router in order to prevent IP address spoofing in the LAN segment.

Direct Internet Access Service Security Considerations

Security attacks are more prevalent when DIA service is used, as it exposes the carrier and, in turn, the end user to the Internet. The purpose of the service is to enable different customers of the same carrier to communicate between each other and over the Internet. However, this access increases the inherent risks of external attacks. The following security techniques are described in this section:

- IPFIX export for monitoring of traffic flows and detection of DoS attacks
- Unicast RPF check strict configuration at customer interfaces
- Unicast RPF check loose configuration at peer or uplink interfaces
- Unicast RPF check loose discard next-hop verification mode to support source, remotely triggered black holes
- A scheme of distributing and installing customer routes with black hole communities

Protecting Against IP Address Spoofing

IP address spoofing is often a major component of these attacks. A simple spoofing mitigation technique is to check the source IP address of packets received from all users, the Internet, and blocking traffic that is not expected from these sources. Configuring unicast RPF check in strict mode at the PE-to-CE interface junction prevents spoofing by effectively blocking any traffic from prefixes that are not reachable through the configured interfaces.

This solution, however, does not support a topology with asymmetric routing (that is, when the CE device announces better paths through another PE device). This lack of support might result in locally received routes becoming inactive. To address this issue, we configure RPF checking over all feasible paths.

At peering locations, you cannot configure unicast RPF check in the same mode because of the asymmetry of typical traffic flows. Instead, you can configure limited protection from spoofing attacks at the peering location by configuring unicast RPF check in loose mode. This configuration results in traffic from unallocated address space (that is, traffic that has no routes in the routing table) being dropped.

Protecting DIA Infrastructure Elements

Network infrastructure elements continue to remain primary attack targets. Protecting the control plane of the AS boundary router and PE routers from Internet-sourced DoS attacks can be a challenge because not all service providers protect against IP address spoofing. This lack of protection increases the possibility of injecting illegitimate packets into the network.

Implementing the Generalized Time-to-Live (TTL) Security Mechanism (GTSM), as described in RFC 5082, can help to mitigate infrastructure attacks. GTSM provides a way to validate the expected TTL value of packets from outside of the network. Protection of AS boundary routers and PE routers using the GTSM requires the following:

- Assigning maximum possible TTL for packets sourced from the router
- Filtering packets not only by source address (most typically a BGP peer), but also by their TTL value (255, in most cases)

NOTE: GTSM is not a substitute for authentication mechanisms. It does not secure against insider, on-the-wire attacks (for example, packet spoofing or replay).

Techniques to Block Denial-of-Service Attacks

Should DIA service DoS attacks occur, many unprotected hosts can quickly exhaust the capacity of the carrier network or its peering capacity. Techniques described in RFC 3882, “Configuring BGP to Block Denial-of-Service Attacks,” describe how to use BGP communities to remotely trigger black-holing of a particular destination network to block DoS attacks. Using these techniques, a CE router can announce a route that points to an attack target so that the service provider can discard traffic intended for that target closer to the peering location. This information can be further propagated to other carrier peers to block the traffic closer to the source.

NOTE: The techniques described in RFC 3882 can also block traffic from legitimate hosts.

The techniques described in RFC 5635, “Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF),” provide a more granular approach to blocking problematic traffic. Using RFC 5635, a CE device announces prefixes from which it wants to block traffic. These prefixes are identified by using a community that the customer and the service provider have agreed upon. Unfortunately, because it can be initiated by one carrier customer and ultimately

affect all carrier customers, this technique possesses a security threat within itself. Because of this potential security threat, this option is not normally provided to regular customers, but is used to integrate with other DDoS mitigation systems internal to the carrier.

Layer 3 VPN-Specific Design Considerations

The Junos OS software provides several configuration choices for Layer 3 VPNs that can affect scaling and operation of other solution components. These choices include the following:

- Label Allocation Principle
- Implementing Routing Topology Restrictions
- Assigning Route Distinguishers

Label Allocation Principle

The term VRF *table label* refers to a particular way of implementing per-virtual routing and forwarding (VRF) table allocation for local routes in a Layer 3 VPN that are announced to other PE routers in the network. This allocation can occur using one of three different methods:

- Per prefix—Every Layer 3 VPN prefix receives its own label allocation.
- Per next hop—All Layer 3 VPN prefixes that point to a specific next hop (typically, a CE device) receive the same label allocation.
- Per table—All Layer 3 VPN prefixes receive the same label allocation.

Junos OS supports the last two allocation methods. Per-table label allocation is enabled through explicit configuration of a virtual tunnel interface in a routing instance or through using the **vrf-table-label** statement.

Implementing label allocation sometimes affects the configuration of certain features that are required in the forwarding path on PE routers. For example, per-table allocation requires destination address IP lookup on the egress PE node, in addition to the MPLS lookup, while per-prefix and per-next-hop allocation do not. In addition, Junos OS implementation of per-next-hop and per-table allocation requires that certain constraints be taken into account during the network design process.

It is generally recommended that you include the **vrf-table-label** statement in your configuration. In the following situations, the inclusion of the statement is required:

- Enabling address resolution for directly attached hosts where the traffic moves in the MPLS to IP direction. Without including the **vrf-table-label** statement (or a virtual tunnel interface configuration), the Junos OS router will not announce the subnetwork prefix to the rest of the network (unless a single static route exists that resolves to a next hop on a LAN segment).
- Using the host fast reroute feature.
- Using the next-generation multicast VPN feature set with MPLS transport.

Usage of the per-table label method might also improve convergence in CE device dual-homing scenarios following primary PE-CE link failure.

For the above reasons, per-table allocation is used in this solution.

Implementing Routing Topology Restrictions

One goal of this solution deployment is to support multiple route distribution topologies with the ability to implement various traffic forwarding restrictions. The default topology for this solution is any-to-any, where all VPN sites exchange routes. However, another topology option is hub-and-spoke, where route distribution is allowed between hub-and-spoke sites and between hub sites, but routes are not distributed between different spoke VPN sites.

NOTE: Forwarding between spoke sites is also prevented.

Figure 19 provides an example of a hub-and-spoke topology.

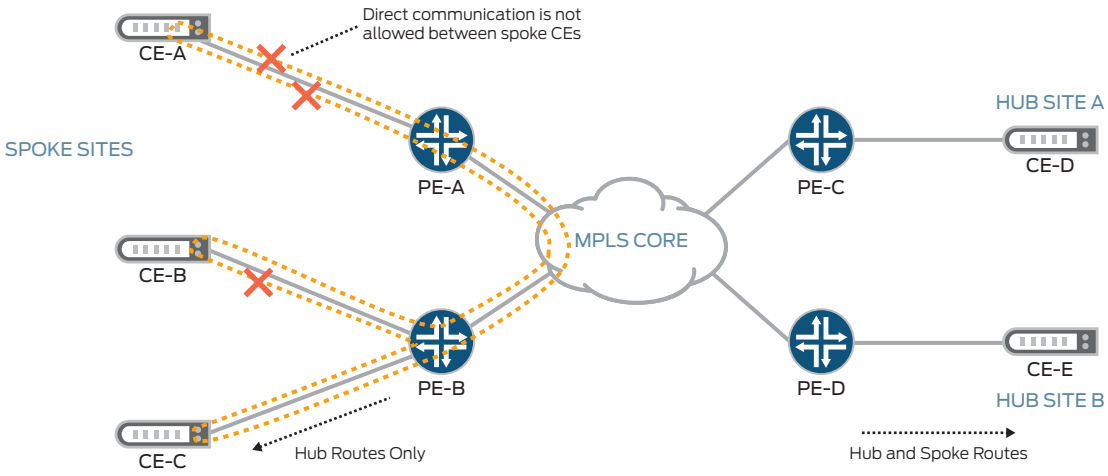

Figure 19: Hub-and-spoke route distribution example

The following method is used to limit route distribution:

- Hub PE devices export routes with a target community that identifies specific hubs.
- Spoke PE devices export routes with a target community that identifies certain spokes.
- Hub PE devices import routes tagged with both a spoke community and hub community.
- Spoke PE devices import routes tagged with hub communities only.

Limiting routing exchange between PE devices does not prevent spoke communication completely, because several spoke CE devices can connect to the same PE device. However, in configurations where multiple CE devices connect to the same PE device, you can implement one of the following design options to limit forwarding between CE devices:

1. Allocate a dedicated VRF for each attachment circuit on each spoke PE device.
2. On each spoke PE device:

- Create a hub VRF that imports all hub site routes and contains no attachment circuits
- Create a separate spoke VRF that has only spoke routes and contains all configured spoke attachment circuits
- Create a routing policy and filters to direct all traffic from spoke sites to the hub VRF, while allowing traffic from the hub site to reach spoke sites

The first option is easier to configure for one or two attachment circuits. However, the second option is more scalable, because it does not require a dedicated VRF per attachment circuit. For this reason, we use the second option in this design.

A similar problem can occur when traffic from one spoke to another spoke arrives at the same hub PE device. When implementing the **vrf-table-label** Layer 3 VPN configuration setting, the IP lookup occurs upon egress and the traffic is returned directly to the target spoke. This event should also be avoided.

To block this traffic path, you can fall back to the default per-next-hop label allocation mechanism. This configuration does not perform the IP lookup and results in all traffic being forwarded to the hub CE device. Implementing this configuration, however, results in the inability of the hub CE router to be a host device.

The BE solution implements this fallback design option because, in most deployments, it is expected that this limitation is negligible.

Assigning Route Distinguishers

We recommend that you use a unique route distinguisher for each PE router participating in a specific routing instance. Although you can use the same route distinguisher on all PE routers for the same VPN routing instance, if you use a unique route distinguisher, you can determine the CE router from which a route originated within the VPN. In this solution design, individual route distinguishers are assigned to each PE router, as this is the best common practice today.

There are two options for configuring route distinguishers:

- Static configuration in the routing instance
- Automatic configuration (where the **route-distinguisher-id** is specified in the global routing options configuration)

This design configuration employs the first option of static configuration in the routing instance.

Direct Internet Access-Specific Design Considerations

When designing the Business Edge, DIA design considerations must often be taken into account. In addition to any Layer 3 VPN considerations, you must also account for the distribution of public IP prefixes and additional security considerations which accompany exposing a VPN to the public Internet. The following sections outline general routing and BGP policy design decisions that are specific to the way DIA service is implemented for this solution:

- DIA and Routing
- DIA and BGP Policies

DIA and Routing

To match the best current practice, this solution implements Internet routing within an individual router global routing table (that is, inet.0) and not in a VPN.

DIA and BGP Policies

Service providers use routing policies to achieve many different network management goals. These goals can include defining specific business relationships with neighboring network domains, improving end-to-end network performance for end users, enhancing routing protocol scalability, and protecting the network from denial-of-service attacks. The following sections provide some policy design considerations to enhance performance in networks that implement a DIA service:

- Defining BGP Policy Business Rules
- Prefix Aggregation
- DIA and Route Import
- Multihoming Support
- Location Considerations
- Support for Remote Triggered Black Hole Filters
- Designing Policies

Defining BGP Policy Business Rules

Service providers each implement certain BGP policies that reflect their own specific business rules. Service providers typically define the following business rules when implementing a Direct Internet Access (DIA) service:

- Connectivity between upstream providers (global Internet) and DIA customers
- Connectivity between peers and DIA customers
- Connectivity between DIA customers
- No connectivity between peers and upstream providers

Implementing these business rules result in the following routing policies:

- Export of customer routes to peers, upstream devices, and all other customers
- No export of peer routes to upstream devices
- No export of upstream routes to peers
- No export between peers or between upstreams

Prefix Aggregation

Customer routes can belong to the address block allocated to the provider or they can be provider independent. When they belong to the address block allocated to the provider, the route addresses are not exported to upstream devices or peers; an aggregate route is exported instead. When they are provider independent, the route addresses are exported as is, without aggregation.

It is possible that the aggregation of routes can result in black holes. Say, for example, that two border routers send aggregates to a peer. The first border router contains all customer prefixes, but the second border router contains only a portion of the customer prefixes because of a temporary network failure. The peer expects that both border routers will provide the same service, sending traffic to its intended destination. However, because the second device has only a partial list of customer prefixes, some traffic sent to the second router runs the risk of eventually falling into a black hole.

To avoid black holes, aggregation must be implemented cautiously and implemented only on routers that have identical route sets covered by aggregates. This means that aggregate routers should have equal connectivity to customer routers and maintain the same level of redundancy. This level of redundancy is typically implemented for AS boundary router locations, but not implemented for peripheral locations where PE routers reside. In other words, aggregate prefixes are typically sent to upstream devices and peers (connected to AS boundary routers), whereas customers (connected to PE routers) typically receive more specific routes without aggregation. For these reasons, this configuration implements prefix aggregation on AS boundary routers only.

DIA and Route Import

When importing routes from customers and peers, the following actions (at minimum) are taken to eliminate potential security problems:

- Reject bogon (martian) prefixes

CAUTION: Bogon prefixes should never be accepted from any customer.

- Reject certain infrastructure routes pertaining to the service provider network

Bogon prefixes are commonly used as source addresses in DDoS attacks. Bogons, also called ***martians***, are private and reserved addresses that are defined by RFC 1918, RFC 5735, and RFC 6598. A bogon prefix is a route that should never appear in a routing table, and a packet routed over the public Internet (not including over VPNs or other tunnels) should never have a source address in a bogon range.

You can find bogon lists that are maintained by various groups and posted on the Internet. However, it is important to note that bogon lists are not static. IP ranges are regularly added to and removed from bogon lists. When filtering bogons, you must ensure that you have a plan for managing your filters.

It is also possible for customers to advertise valid routes that do not belong to them. This type of advertisement can occur due to human error, a software problem, or as an attempt to purposely initiate a “man in the middle” attack. To eliminate the chances of these routes causing networking problems, service providers often implement customer-specific route filtering when importing routes from customers and from peers. For some trusted peers, these import policies might not be required. However, both import policy types should be supported (with strict prefix checks and without them) to cover both scenarios.

Service providers often maintain their own route community plan, with each community signaling special route import or export semantics. These communities can be used between customers and service providers to indicate special requirements to the route redistribution mechanisms. It is important that these communities are analyzed and route attributes modified according to the semantics of each community. Following the analysis, all communities pertaining to the service provider should be discarded to avoid any overlap with internally used communities.

The number of accepted routes from a neighbor is often limited to protect from any peer misconfiguration. The configured route limit must be high enough to function for all customers and peers, but low enough to protect the routing control plane from resource exhaustion. As a general rule, a route limit of 1,000 prefixes from customers and 10,000 prefixes from peers should be enough to achieve this goal.

NOTE: The number of routes received from an upstream device is not typically limited.

Multihoming Support

When a customer site is connected to two different routers, the customer might have some preference as to which link to use in order to reach certain destinations in their own network.

Informational RFC 1998 defines community values that can be used to signal route preference between the CE device and the PE device. Even though the RFC is intended to be informational only, many service providers adhere to the community plan it defines. For this reason, the same plan is used in this design.

Table 9 shows the association between the community value and the local preference value.

NOTE: You must substitute $AS with the actual autonomous system number.

Table 9: Community and Local Preference Association

Community	Local Preference Value
$AS:70	70
$AS:80	80
$AS:90	90

Location Considerations

When designing a network, it is common practice to implement location-specific routing policies that employ business rules that are specific to that location. For example, depending on the device location, you might choose to advertise some prefixes at some peering points and not advertise them at other peering points, or you might choose to configure different multiple exit discriminator (MED) values or AS paths to be implemented upon export.

A typical method for enabling route selection upon export is to assign a community that identifies route location upon importation of routes. This solution supports a community to identify the source location. However, because export policies that take region communities into account are very specific to individual customer network environments, they are not covered in the solution.

Support for Remote Triggered Black Hole Filters

Remote triggered black hole filtering is a security mechanism used to reduce the impact of DoS attacks. You can implement the filtering to trigger on either destination or source addresses in the customer network. A customer under attack can choose to discard traffic to certain destinations or from certain sources, managing the denial-of-service attack closer to the attack source within the customer network (for example, at the service provider AS boundary router). This is called remote triggered black hole (RTBH) filtering and is defined in RFC 5635.

When implementing destination-based RTBH filtering, the customer announces black hole prefixes with a special community that signals the provider to install a route with a discard next hop at all network locations, including AS boundary nodes. Traffic entering the network is automatically discarded at the first service provider router. A security check that allows installation of only those prefixes that belong to the customer address space should be made at each import location.

Source-based RTBH filtering works in a similar way. The prefixes that are tagged with a special community are installed into the forwarding table of all routers (including AS boundary routers) with a discard next-hop action. Each router has RPF check verification enabled in loose mode for peer and upstream interfaces. Source address lookup is performed for packets received from peers and upstream devices. If the address matches any black hole routes (those pointing to discard interfaces), the packets are discarded.

Source address prefixes that are signaled for installation belong to Internet address space and are not limited to a specific customer. By its nature, source address RTBH does affect all service provider customers, because packet destination is not taken into account when a packet from a certain source is discarded. In contrast, destination-based RTBH filtering affects only customer-specific destinations. For this reason, source-based RTBH filter installation is typically not enabled for service provider customers, but it might be enabled for some internal systems (for example, DDoS mitigation systems).

Designing Policies

Table 10 provides high-level policy definitions for various peers and profiles; these definitions implement the logic defined in the previous sections.

Table 10: High-Level Policy Definitions

Peer Type	Profile	Import Policy	Export Policy
Customer	#1, no prefix check	1. Reject bogons 2. Reject my infrastructure networks. 3. Process local preference communities. 4. Remove my communities. 5. Add community “customer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer,” “peer,” and “upstream” communities.
	#2, prefix check	1. Reject bogons. 2. Reject my infrastructure networks. 3. Accept routes from a customer-specific prefix list, reject the rest. 4. Process local preference communities. 5. Remove my communities. 6. Add community “customer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer,” “peer,” and “upstream” communities.
	#3, prefix check, allow destination black hole installation	1. Reject bogons. 2. Reject my infrastructure networks. 3. Accept routes from a customer-specific prefix list, reject the rest. 4. Accept black hole routes (tagged with black hole community names), set discard next hop. 5. Accept routes from a customer-specific prefix list, reject the rest. 6. Process local preference communities. 7. Discard my communities. 8. Add community “customer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer,” “peer,” and “upstream” communities.
	#4, no prefix check, allow source black hole installation	1. Reject bogons. 2. Reject my infrastructure networks. 3. Accept black hole routes (tagged with black hole community names), set discard next hop. 4. Process local preference communities. 5. Discard my communities. 6. Add community “customer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer,” “peer,” and “upstream” communities.
Peer	#1, no prefix check	1. Reject bogons. 2. Reject my infrastructure networks. 3. Discard my communities. 4. Add community “peer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer” community; aggregate on export.
	#2, prefix check	1. Reject bogons. 2. Reject my infrastructure networks. 3. Accept routes from a customer-specific prefix list, reject the rest. 4. Discard my communities. 5. Add community “peer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer” community; aggregate on export.
	#3, prefix check, allow destination black hole installation	1. Reject bogons. 2. Reject my infrastructure networks. 3. Accept routes from a customer-specific prefix list, reject the rest. 4. Discard my communities. 5. Add community “peer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer” community; aggregate on export.
Upstream Device	#1, accept all	1. Reject bogons. 2. Reject my infrastructure networks. 3. Discard my communities. 4. Add community “peer” and “region.”	1. Export direct routes, except infrastructure range. 2. Export routes marked with “customer” community; aggregate on export.

The Junos OS provides many ways with which you can translate these high-level policy definitions into low-level configurations. For example, you can configure policies in a flat list format where each new peer policy statement includes all of the terms that are specific to the peer; or you can define all policies at the outset of the design process and then combine them with import or export lists; or you can create a master policy that calls for nested policies.

Figure 20 provides a visual representation of policy chaining and term consideration.

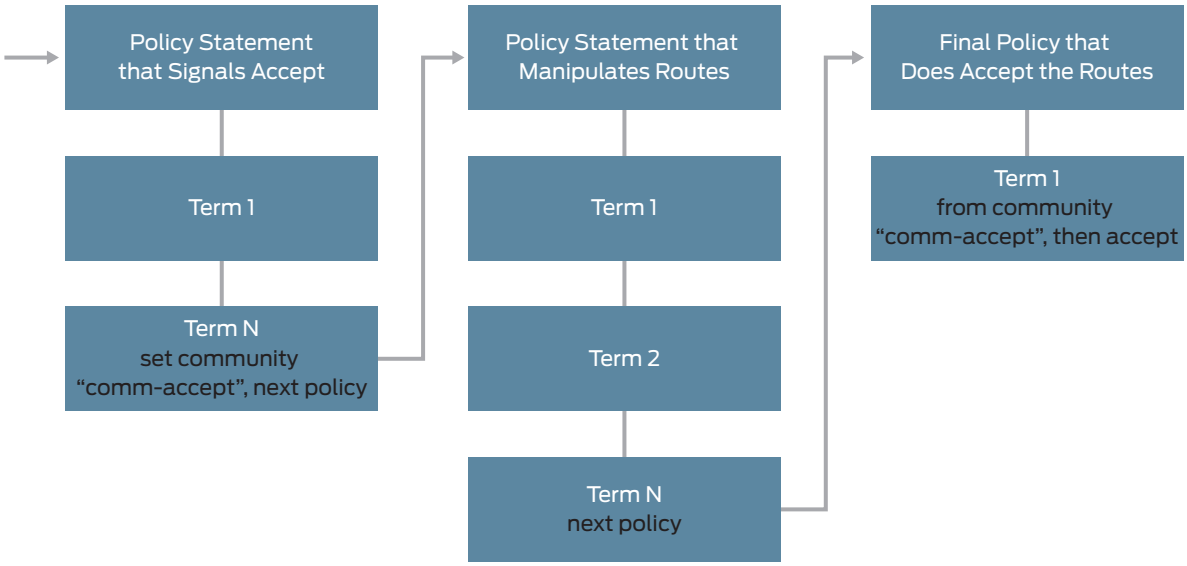

Figure 20: Solution policy chaining

In this configuration, we use elementary polices that are called in a specific order by an import or export policy list. When facilitating chained policy execution, it is important to use the **next** policy statement in each elementary policy to minimize usage of **accept** terminating actions, saving those actions for use as the very final policy action. This precaution ensures that if a route is accepted, the policy execution is not stopped; route characteristics continue to be manipulated by subsequent policies, and the route accept state is passed along within the special community.

Components

The business edge solution leverages Junos OS software, the common operating system that runs on all Juniper Networks routing, switching, and security platforms. Junos OS supports a wide variety of routing protocols, Layer 2 or Layer 3 VPN technologies, and seamless MPLS, providing true carrier-class network and service reliability and performance.

The following sections specify supported and required components:

- Topologies
- Services
- Platform Requirements
- Line Card Requirements

Topologies

This solution supports the following topologies:

- Any-to-any VPNs
- Hub-and-spoke VPNs

Services

This solution supports the following services:

- NSR for BGP, IS-IS, RSVP-TE in Layer 3 VPN context, LDP
- GRES for IS-IS and BGP
- Unified ISSU with a target of less than a 1 second dark window at scale
- Inline IP Flow Information Export (IPFIX) for IPv4 and IPv6 traffic
- Juniper Networks SRX Series Services Gateways stateful firewall for hub-and-spoke VPNs

Platform Requirements

The business edge solution utilizes platforms from the MX Series family of routers, including the Juniper Networks MX2020, MX2010, MX960, MX480, MX240, and MX80 3D Universal Edge Routers.

This version of the business edge solution requires the following platforms:

- MX2020, MX960, MX480, and MX240 routers
- SRX240H routers
- Enhanced Switch Control Board (eSCB) backplanes installed on all MX Series devices
- RE 1800-X4 Routing Engines installed in all MX Series devices

Line Card Requirements

The business edge solution utilizes service cards such as the MS-DPC which delivers advanced services such as network addressing, stateful firewall, Dynamic Application Awareness, and IPsec to the Juniper Networks universal edge solution. The advanced Junos Trio chipset, which provides 3D scaling of bandwidth, subscribers, and services, ensures that the service provider is able to meet the demands of even the fastest growing markets.

This version of the business edge solution requires MPC line cards.

CHAPTER 3 Advanced Connectivity Design Considerations

- Advanced Connectivity Design Overview
- Next-Generation Multicast VPN Design Options
- Multicast VPN Network-Level Reliability Design Options
- Next-Generation MVPN CoS Design Considerations

Advanced Connectivity Design Overview

Advanced connectivity services include various advanced options provided on top of the basic connectivity services (for example, multicast traffic distribution). Services in this category are requested by a fraction of the basic connectivity customers and are usually sold at premium prices and in small volume.

This solution describes the multicast VPN (MVPN) service where multicast traffic is delivered within a customer VPN, including traffic sourced from extranets.

This document describes the use of MVPNs as the advanced connectivity service for this solution. The MVPN service provides a multicast traffic distribution option that functions over a base Layer 3 VPN service. The MVPN service can be provided to business customers or used by the service provider.

Multicast VPNs are used to deploy multicast service in an existing VPN or as part of a transport infrastructure. Essentially, multicast data is transmitted between private networks over a VPN infrastructure by encapsulating the original multicast packets. As expected, in networks running a multicast VPN service, traffic sources send multicast traffic to the network, while multicast receivers consume that traffic. The difference between transmitting and receiving devices is functional. In other words, each customer edge (CE) device can be used as both a receiver and a source. However, this functional merge usually results in a different set of protocols functioning at the attachment circuit (for example, IGMP instead of PIM).

The following sections provide more information about different MVPN options:

- Understanding Draft-Rosen VPNs
- Understanding Next-Generation Multicast VPNs

Understanding Draft-Rosen VPNs

Originally defined by RFC2547, “BGP/MPLS VPNs” in October 2004, Layer 3 BGP/MPLS VPNs are widely deployed today. In February 2006, RFC 2547 was superseded by RFC 4364, “BGP/MPLS IP Virtual Private Networks (VPNs).” Because of their association with the original RFC number and the primary author of the RFC, these VPNs are commonly referred to as “2547” or “Draft-Rosen” VPNs.

NOTE: This document does not provide configurations for Draft-Rosen VPNs. However, it is important to understand this type of VPN to better understand its limitations and the subsequent advantages of using a next-generation multicast VPN configuration.

Draft-Rosen MVPN Deployment

RFC 2547 and RFC 4364 describe protocols and procedures for creating BGP/MPLS VPNs to forward VPN unicast traffic only. Multicast VPN traffic is handled using an incremental approach (RFC 6037), where the multicast traffic configuration is overlaid onto the BGP/MPLS unicast network.

Basic MVPN deployment requires the use of a mechanism for carrying MVPN-specific multicast routing information. This mechanism, known as the “control plane,” carries control messages from “receivers” in the network to different “sources” in the network to indicate that the receiver sites are prepared to receive traffic from source (sender) sites.

In addition to the control plane, a mechanism is required to carry the multicast traffic between routers. This mechanism, known as the “data plane,” carries traffic from sources to identified receivers.

NOTE: Different MVPNs might use the same address space (RFC 1918), including an IP multicast address space. To support the overlapping address space, a best practice would be to use the same route distinguisher as that used by the Draft-Rosen VPNs for unicast traffic.

Draft-Rosen MVPN Limitations

In a typical Draft-Rosen multicast VPN deployment model, VPN multicast traffic travels over an existing unicast Draft-Rosen network, resulting in the need for a virtual router architecture. In this model, Protocol Independent Multicast-Sparse Mode (PIM-SM) is used to exchange control plane information between virtual routing and forwarding (VRF) instances. Employing this type of network, where customer domain multicast protocol information (for example, PIM join or prune messages) is received from local CE routers and propagated to other PE devices, scale can become an issue. For any given MVPN, a PE device must maintain a PIM session with every other PE device that has a membership in that MVPN. As an example, a network containing 1,000 MVPNs per PE device and 100 sites per MVPN would have 100,000 PIM neighbors per PE device generating 3,300 PIM hellos per second.

Understanding Next-Generation Multicast VPNs

To address the control and data plane scaling issues brought about through the use of Draft-Rosen MVPNs, which requires providers to maintain two different routing and forwarding mechanisms for VPN unicast and multicast services, the IETF Layer 3 VPN working group defined next-generation MVPNs. The working group published an IETF draft (draft-ietf-l3vpn-2547bis-mcast) that is a superset of the previous specifications for MVPNs. In addition, the following two main drafts were created to define next-generation MVPN design and configuration:

- draft-ietf-l3vpn-2547bis-mcast-bgp—Outlines the procedures of the next-generation MVPN
- draft-ietf-l3vpn-2547bis-mcast—Describes the building blocks for the different next-generation MVPN configuration options

The use of scalable, reliable, and secure next-generation multicast VPN configurations can enable service providers to increase revenues by deploying more lucrative, media-rich applications.

The next-generation MVPN drafts introduced a BGP-based control plane that is modeled after its highly successful counterpart of the VPN unicast control plane. Leveraging the functionality of common unicast BGP-MPLS VPNs, next-generation MVPNs utilize the following properties:

- The BGP protocol distributes all necessary routing information to enable the VPN multicast service.
- The BGP protocol distributes C-multicast routes, resulting in the control traffic exchange being out-of-band from the data plane. This implementation enables the separation of the control and data plane protocols and simplifies the use of new transport technologies (for example, point-to-multipoint MPLS) for delivering MVPN services.

The use of a BGP-based NG MVPN control plane enables the support of both flexible topologies (for example, extranet or hub-and-spoke) and IPv6 addressing. Implementing IPv6-based NG MVPN enables the use of MPLS encapsulation for IPv6 multicast. As an added benefit, IPv6-based next-generation MVPN uses the same model as IPv6 VPN for unicast (as defined in RFC 4659), ensuring a more compatible integration of IPv6 multicast services with existing IPv4 next-generation MVPN or IPv6 unicast VPN models.

BGP MVPNs provide multihoming support for connecting a multicast source to two provider edge (PE) devices, enabling subsecond failover from a PE node failure. The autodiscovery of MVPN members available with the BGP approach provides a high degree of automation for establishing provider tunnels that are used for carrying MVPN data between PE devices.

The multicast VPN connectivity service designed in this solution adheres to RFC 6513, “Multicast in MPLS/BGP IP VPNs,” also known as next-generation MVPNs. Figure 21 shows the general NG MVPN design topology.

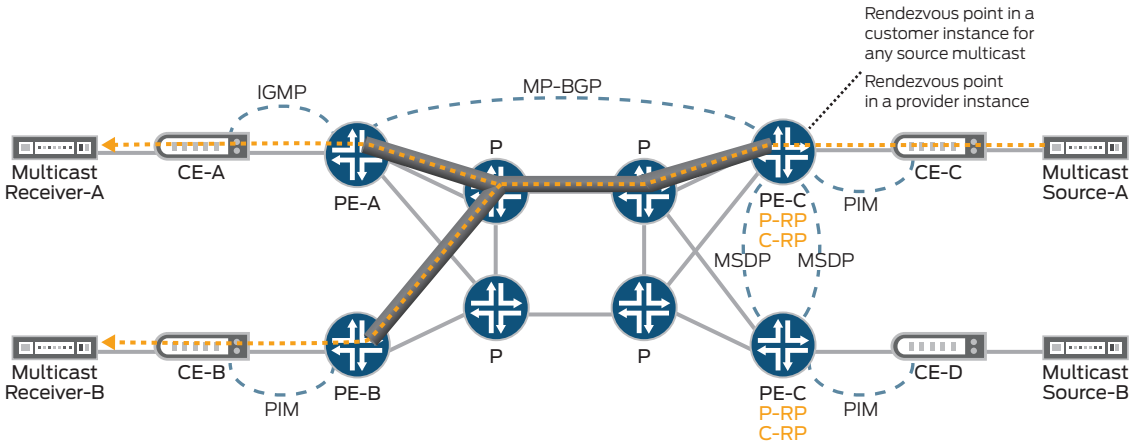

Figure 21: NG MVPN design topology

Because of the large number of transport and signaling options, multicast VPN is one of the most complex services to design. Table 11 lists the NG MVPN designs supported in this example.

Table 11: MVPN Transport and Signaling Options

Parameter	Next-Generation MVPN P2MP RSVP-TE	Next-Generation MVPN P2MP mLDP	Next-Generation MVPN P2P	Rosen V7	Rosen V6
PE device autodiscovery mechanism	BGP	BGP	BGP	BGP	PIM-SM
Routing of the customer multicast between PE devices	BGP	BGP	BGP	PIM-SM	PIM-SM
Replication	Core	Core	Ingress	Core	Core
Provider tunnel type	P2MP RSVP-TE	P2MP mLDP	P2P RSVP-TE/LDP	IP in GRE	

The same next-generation MVPN BGP signaling mechanism is shared by each option. The only difference is in the provider tunnel configuration.

This solution incorporates NG MVPN signaling.

Next-Generation Multicast VPN Design Options

The following sections provide design considerations for NG MVPN configuration:

- Next-Generation MVPN BGP Signaling
- Next-Generation MVPN Provider Tunnels

Next-Generation MVPN BGP Signaling

In this scenario, BGP is used to signal information about active multicast sources and receivers and to facilitate PE router autodiscovery. Figure 22 shows multicast traffic being delivered over an MPLS core using a provider tunnel, P2MP, or P2P (ingress replication) signal using RSVP-TE.

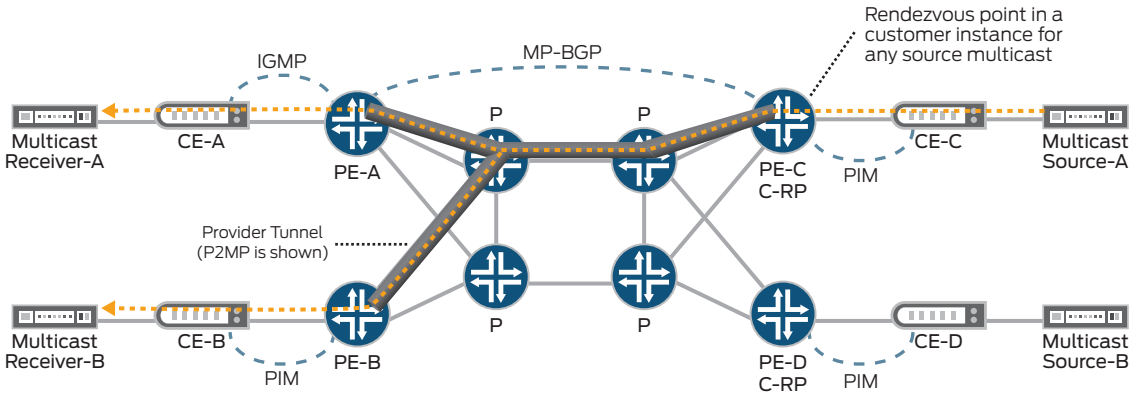

Figure 22: Next-generation MVPN design

Directly attached hosts signal group membership to PE routers using IGMP or Multicast Listener Discovery (MLDv2). Routers use PIM-SM/SSM instead.

In any source multicast scenario, PE routers should discover active sources and should distribute this information to the rest of the network. Automatic discovery is greatly simplified when a source is directly attached to the router.

When a CE device resides between the source and the router, discovery can occur in one of the following two ways:

- Configure a rendezvous point (RP) at the PE router in a customer instance (C-RP) and specify that CE routers use the defined RP
- Configure the Multicast Source Discovery Protocol (MSDP) protocol to operate between the PE router and a customer RP

This design uses the first RP configuration option, because the deployment of basic PIM configurations is prevalent in service provider networks.

For redundancy, you can configure several C-RPs, each on a different PE router. Rendezvous points can share the same IP address (anycast RP) and information about active sources that are registered on different C-RPs is exchanged automatically using BGP, eliminating the need to run MSDP between C-RPs.

As this is a multicast VPN deployment, it is expected that some multicast source sites might send multicast traffic in some groups and receive multicast traffic in other groups. Defining selective multicast trees (where the default behavior is inclusive and all groups share the same tree) results in optimal traffic distribution. Provider tunnel configuration remains the same, but there is more than one tunnel and there is an additional mapping configuration between the multicast groups and tunnels at the PE device.

RFC 6513 also defines aggregate multicast distribution trees, which are those that aggregate multicast traffic from multiple VPNs. Aggregate trees are not supported by this solution.

Next-Generation MVPN Provider Tunnels

This NG MVPN design supports the P2MP RSVP-TE provider tunnels type. When using this tunnel type, it is assumed that this multicast service is used for delivery of continuous multicast traffic flows with known bandwidth.

The design uses the **vrf-table-label** statement when configuring the VPN instance. This statement maps the inner label of a packet to a specific VPN routing and forwarding (VRF) table, enabling IP route lookup. All routes in the VRF configured with this option are advertised with the label allocated per VRF. The result is an inefficiency where multicast traffic is duplicated at the P-PE interface when the PE router also serves in a P router role. To alleviate this inefficiency, virtual tunnel interfaces are used for multicast traffic only.

Two virtual tunnel interfaces are needed for redundancy. One interface is defined as the primary. Traffic processing at the virtual tunnel interface does have performance implications. The PFE where the virtual tunnel interface resides is called the *anchor PFE*.

NOTE: Using either uplink or the access-facing PFEs as anchors can help to avoid extra hops through the switch fabric in distributed systems such as the Juniper Networks MX960 3D Universal Edge Router.

Choosing an access PFE as an anchor has the benefit of fate sharing with the access interfaces. In cases where only one PFE serves all customer interfaces, virtual tunnel interface redundancy might not be required at all. However, because customer access interfaces might reside on several different PFEs, virtual tunnel redundancy will most likely be required.

In some deployments, a customer might choose a specific anchor PFE to optimize multicast traffic replication through the device. In this solution, however, virtual tunnel interfaces are configured only at the uplink PFE.

One of the main benefits in using a next-generation MVPN deployment is its clean and straightforward extranet MVPN implementation. For multicast sources and receivers to reside in different VPNs, you must configure route import and export policies for the PE devices to enable communication between receivers and sources. When configuring these route policies:

- At the PE device with multicast ***sources*** attached, import into the source VPN routes from VPNs where receivers reside

  

  NOTE: You can limit the import to sites with receivers only.
- At the PE device with multicast ***receivers*** attached, import into respective receiver VPN routes from the source VPN

  

  NOTE: Limit the import to sites with sources only.

If the same PE device serves both source VPN (with local sources) and receiver VPNs, you must include an auto-export configuration.

Multicast VPN Network-Level Reliability Design Options

Multicast VPN high availability design inherits some of the redundancy and the reliability design considerations highlighted for the basic Layer 3 VPN service design.

Link-level redundancy is implemented by using the same aggregated Ethernet technology used for Layer 3 VPN unicast service. For local repair of the P2MP RSVP-TE LSPs in the network core, link protection is used.

Next-Generation MVPN CoS Design Considerations

The class-of-service (CoS) architecture used for the MVPN service is the same as that used for the basic connectivity (Layer 3 VPN/DIA) CoS architecture.

CHAPTER 4 Value-Added Connectivity Design Considerations

- Value-Added Services Design Overview
- Firewall Virtualiz ation
- Routing Design
- Class of Service
- Security
- Stateful Firewall

Value-Added Services Design Overview

Value-added services or additional services that generate additional revenue for service providers are advanced by offering added network features and benefits to subscribers. This solution supports the configuration of a firewall and NAT service.

Functioning over the Layer 3 VPN base network configuration, the firewall/NAT service can be apportioned as follows:

- Layer 3 VPN service between the customer locations and the data center locations where firewall/NAT devices reside
- DIA service between data center locations and the Internet
- Firewall/NAT processing for each customer VPN in a data center location

Figure 23 shows a high-level diagram of the firewall/NAT service design.

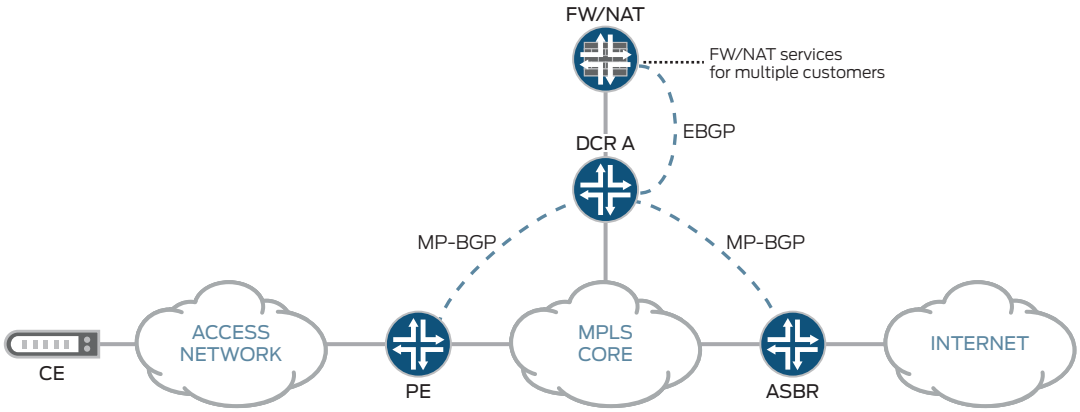

Figure 23: Layer 3 VPN firewall/NAT design topology

In this design, the firewall/NAT device (SRX240H) supports multiple customers. You can configure the device in the following two ways:

- Each customer is allocated two logical interfaces, one facing the customer VPN and the other facing the Internet.
- Each customer is allocated only one logical interface facing the customer VPN. Traffic from all customers is aggregated. The firewall/NAT device has only one logical interface facing the Internet.

While the second option has more optimal provisioning overhead, it does have the following adverse effects:

- Possible incompatibility with the billing requirements for the DIA service
- Difficult policing implementation
- Potentially significant changes to provisioning systems

Because of these potential problems, the firewall/NAT deployment uses only the first option where each customer is allocated two logical interfaces.

The Dynamic Context Router (DCR) functions as both a Layer 3 VPN PE and DIA PE device simultaneously. Throughout the firewall/NAT design, these names are used in different contexts to highlight the specific roles of the DCR.

Firewall Virtualization

This solution employs high-end SRX Series secure routers in the firewall/NAT device role. SRX Series devices support both virtual routers and logical systems for use in device virtualization. Without delving into the benefits of either virtual routers or logical systems, the design of this network uses only virtual routers when employing virtualization techniques.

In Junos OS software, a virtual router is a type of routing instance, which is a collection of routing tables, interfaces, and routing option settings. A security device like the SRX Series Services Gateways can divide its routing component into two or more virtual routers. Each virtual router supports static routing, dynamic routing protocols, and multicast protocols, all of which you can enable simultaneously in one virtual router.

Routing Design

Forwarding traffic between Layer 3 VPNs and the Internet requires the following:

- Propagating customer NAT pools to the Internet (for NAT services)
- Propagating customer prefixes to the Internet (for firewall services)
- Propagating Internet routes (or default routes) to customers

Because propagating the entire routing table would result in scaling limitations, Internet destination default routes are advertised into the Layer 3 VPN instead.

Default routes are injected using any of the following:

- Layer 3 VPN PE devices
- Firewall/NAT devices
- DIA PE devices

To eliminate possible black hole issues, we recommend injecting the routes as close as possible to their source. In this networking case, that means interjecting the routes at the DIA PE device level.

To propagate the default route, customer prefixes, and NAT pools to the Layer 3 VPN PE device requires the use of a routing protocol. Two possible choices are OSPF or BGP. For this design, we chose to use BGP because it provides better scaling and, therefore, is better suited for large deployments.

Figure 24 shows the Layer 3 VPN PE, firewall/NAT, and DIA PE devices and the relationships between them.

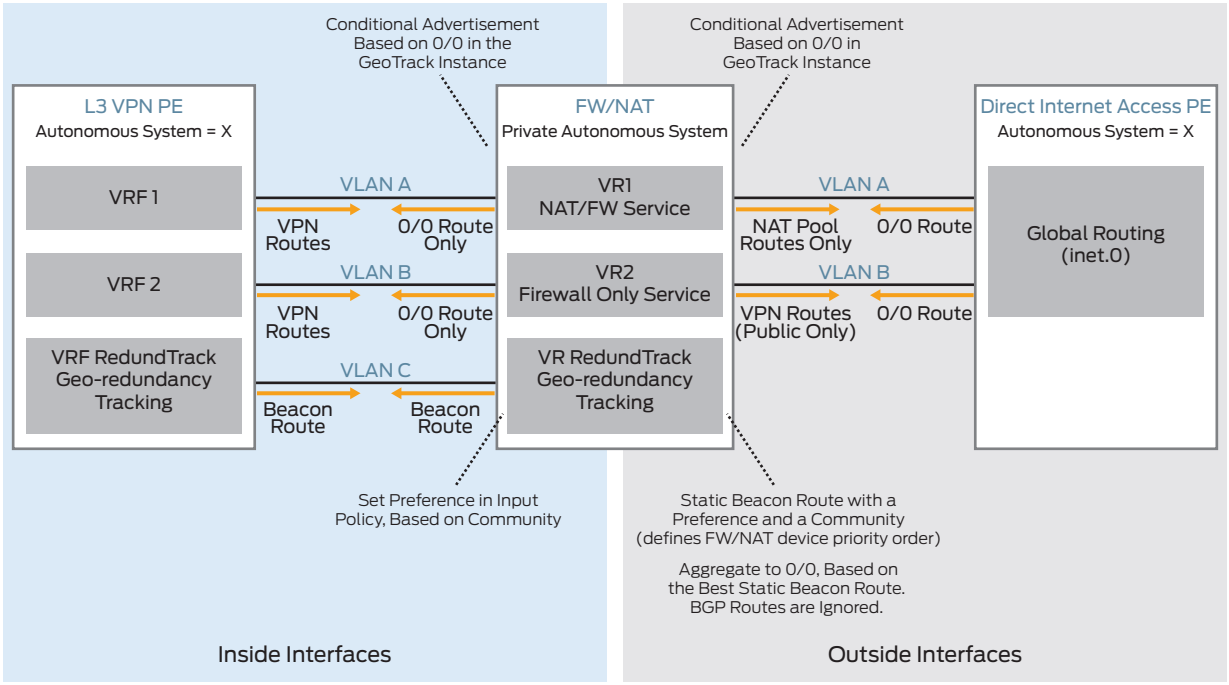

Figure 24: Firewall/NAT routing design

The DIA PE and Layer 3 VPN PE router functions are likely to be deployed on a single hardware platform (for example, over a data center router).

In this design, every customer is allocated one virtual router instance on the firewall/NAT device. This firewall/NAT instance has two logical VLAN interfaces, one facing the Layer 3 VPN PE router and the other facing the DIA PE router.

NOTE: In this design, it is assumed that both the Layer 3 VPN PE router and the DIA PE router reside in the same autonomous system and that the firewall/NAT device is located in a private AS.

Routes from the DIA PE device to customers are distributed as follows:

- The DIA PE device advertises 0/0 as the default route to the firewall/NAT device.
- The firewall/NAT device accepts the 0/0 route only and blocks any other routes (as a precaution). The firewall/NAT device overrides the AS number to any private autonomous system.
- The Layer 3 VPN PE device accepts routes from the firewall/NAT device and overrides the AS (to AS #X).

Route distribution from the customer to the DIA PE devices is accomplished in a similar fashion:

- VPN routes are propagated from the Layer 3 VPN network to the firewall/NAT device. The firewall/NAT device overrides the AS number and discards any default routes.
- When running a firewall/NAT service, only NAT pool routes are distributed to the DIA PE device. Any VPN customer routes are blocked. When running a firewall-only service, public VPN routes are distributed to the DIA PE device.
- The DIA PE device overrides the AS number for all incoming routes. As a precaution, the DIA PE device also blocks the 0/0 route to avoid loops.

Class of Service

The firewall/NAT value-added service is a combination of the Layer 3 VPN and DIA services and inherits the supported class-of-service (CoS) mechanisms of both.

Even though the firewall/NAT device (the SRX240H) is a separate device that was added to support this value-added service, the SRX Series gateway does not require any specific CoS configuration except for management traffic classification and marking. The device is configured to transmit transit IP packets transparently.

Security

To implement the firewall/NAT value-added service, only an SRX Series device was added to the topology. This device is protected in a similar way as the AS boundary router for the DIA service. However, the firewall/NAT device is vulnerable to attack through session table overflow.

Because firewall/NAT is handled by a flow-based device with a finite session table capacity, session table capacity can be easily depleted by a virus, a trojan, or software errors occurring at the customer premises. To prevent session table overflow, we strongly recommend that you configure session limits by using an intrusion detection service (IDS) policy to limit flows from a single customer IP address.

NOTE: IDS policy configuration can result in a reduction in router performance.

Because customer security requirements can vary widely, their configuration is beyond the scope of this document. However, you can configure a typical firewall policy to allow sessions that are initiated from the customer network and block attempts to establish sessions from the Internet. As it is typical firewall behavior, traffic for established sessions is accepted both ways.

Stateful Firewall

The firewall/NAT value-added service also supports the implementation of stateful firewall between Layer 3 VPN locations. The firewalls are enabled between spoke sites and between spoke sites and hubs. Firewall configuration between hubs is not allowed. The service is implemented as shown in Figure 25.

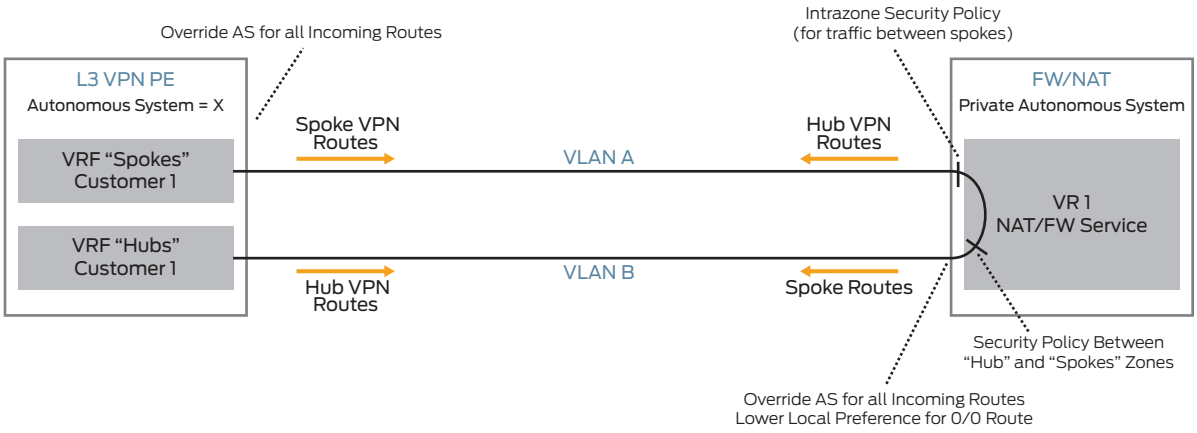

Figure 25: Layer 3 VPN stateful firewall design

Connectivity between spoke VPN sites is provided by a dedicated "spokes" VPN. Similarly, hub sites are placed into a "hubs" VPN.

"Spokes" VPNs are technically hub-and-spoke VPNs with data center locations (that is, a firewall/NAT device) defined as hubs. As a result, all traffic between the spoke VPNs must be processed through a stateful firewall. In this design, the firewall/NAT security policies are configured to process intra-zone traffic (that is, traffic between spokes).

A "hub" VPN is a VPN that uses an any-to-any topology. This VPN provides communication between customer hub sites and firewalls at the data center locations.

A security policy is configured to process traffic between a "hubs" VRF and a "spokes" VRF. The firewall/NAT device participates in BGP routing and transmits all routes between the hub-and-spoke sites, with the exception that the 0/0 route transmitted from hubs is configured with a local preference value lower than the default. This configuration ensures that the hub-to-spoke paths are less preferable if Internet connectivity is provided by the same firewall and the 0/0 route that is received from the DIA PE device.

CHAPTE R 5 Design Configuration Templates

- Using Device Configuration Templates
- Basic Device Configuration Templates
- Service-Specific Design Configuration Templates

Using Design Configuration Templates

This section provides design configuration templates that illustrate low-level configuration of solution components. The purpose of these templates is to help you reach the design goals outlined in the solution design section of this document.

- Understanding the Low-Level Template Design
- Applying Business Edge Low-Level Templates

Understanding the Low-Level Template Design

The following information is provided for every template:

- Template name—this name may be used later as a reference; names are unique.
- Device role to which this template applies—the same template can apply to many device roles.
- Service this template supports—the template may apply only to specific services such as Layer 3 VPN or DIA. It can also be used to support multiple services.
- Usage Guidelines—some templates should be used only if certain conditions are met or under special guidelines. For example, in the template **mpls-rr**, *inet.3* and *inet6.3* default routes should be configured on dedicated route reflectors only. Those guidelines are listed in this section.

Some of the template definition may also include:

- List of constants used in the template—some templates may include specific recommended configuration values pertaining to this solution. Those are listed in the constants section of the template definition.
- Parameters—templates in this solution are parameterized, and those parameters are listed. For each parameter, a name is provided along with the parameter description, including parameter type.
- Dependencies—the template may refer to Junos OS objects defined in some other templates. Those templates are listed in this section.

Finally, the template body contains a snippet of a Junos OS configuration.


NOTE: The template body might refer to parameters using names previously defined in earlier templates. Each reference is called out using a $ sign and braces (for example, *${vrf_target}*).

Table 12 provides an example template. In this example, the following four parameters are defined:


NOTE: Each parameter is referred to in the template body.

- **vrf_target**
- **vrf_id**
- **ipv4_loopback**
- **cust_name**

Table 12: Example Template (Layer 3 VPN Instance)

Template ID	l3vpn-instance-example	**Device Role**	PE device, DCR
Service	Layer 3 VPN	**Guidelines**	Use to provision any-to-any instance
Constants			
Each instance supports up to 10,000 IPv6 prefixes and 100,000 IPv4 prefixes. Once the 75 percent threshold limit is reached, a message is logged.			
Parameters			
vrf_target	VRF route target		
vrf_id	An integer number identifying this particular VRF, used to produce a route distinguisher identifier		
ipv4_loopback	IPv4 loopback address, used to create a route distinguisher identifier		
cust_name	Customer name, used to produce VRF name identifiers		

Configuration Apply Group

```
groups {
    l3vpn-instance-settings {
        routing-instances {
            <*> {
                routing-options {
                    rib <*.inet6.0> {
                        maximum-prefixes 10000 threshold 75;
                    }
                    maximum-prefixes 100000 threshold 75;
                    multipath {
                        vpn-unequal-cost;
                    }
                }
            }
        }
    }
}
routing-instances {
    /* l3vpn- prefix is used in the routing instance name to selectively apply l3vpn
       groups to l3vpn any-to-any service if needed later */
    l3vpn-${cust_name} {
        apply-groups [ l3vpn-instance-settings ];
        instance-type vrf;
        vrf-target ${vrf_target};
        route-distinguisher ${ipv4_loopback}:${vrf_id};
        routing-options {
            /* Need to explicitly define inet6 in order to the prefix limits from the
              l3vpn-max-routes to apply */
            rib l3vpn-${cust_name}.inet6.0;
        }
        vrf-table-label;
    }
}
```

In some cases, a parameter might refer to an array of values; this is indicated in the parameter description.

Table 13 provides a reference to an array element in the template body. The array element is denoted by an [i] suffix.

Table 13: Example Template (Array Element)

Template ID	address-space-example	Device Role	P and PE device, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	
Parameters			
mgmt.-prefix-v4	An array of IPv4 prefixes that refer to the management subnet		
Configuration Apply Group			

```
policy-options {
    /* This prefix list holds all management addresses
        (ssh, telnet, ftp may be initiated from those) */
    prefix-list pl-mgmt-v4 {
        ${mgmt-prefix-v4[i]};
    }
}
```

The semantics in this reference are loose. The actual application guidelines are typically provided in the comments within the template body, meaning that all array values are listed one by one.

Loose semantics are purposely used. Another approach would be to define some form of ***pseudo-language*** that supports loops, conditions, and other programming language constructs. We believe that this construct would complicate understanding of the templates by the unprepared reader. For that reason, this approach was not implemented.

To assist in usage, each template body makes use of the following syntax highlighting scheme:

- Parameters are green.
- User identifiers are blue.
- Comments are *orange (and in italic font)*.
- Interface names are **dark blue**.

Applying Business Edge Low-Level Templates

Applying Templates

The low-level design templates are illustrations of the Junos OS configuration; they do not provide a final configuration that you can apply to the router. To produce a usable configuration, you must perform the following actions:

- Assign a value to each template parameter
- Replace each parameter reference in the template body with the appropriate parameter value

You can accomplish these actions by copying and pasting the desired template hierarchy structure into a text file and making the appropriate modifications. Once completed, you can copy and paste the finished hierarchies into the router CLI.

Different templates might refer to the same Junos OS configuration hierarchy; the templates are prepared with the assumption that the Junos OS commands will eventually be merged with the final configuration. However, the removal of the configuration produced from a template is more difficult. This is because the final Junos OS configuration might include configuration elements shared by other configurations generated from other (or even the same) template.

As an example, assume that the above template was used to produce two configuration elements: VPN A and VPN B. Both configuration elements have the shared group definition **l3vpn-instance-settings**. If VPN B is removed, you must ensure that the group remains in the configuration file for use by the VPN A configuration.

CAUTION: When using templates to construct your configuration, additional checks are required to avoid removal of any shared elements.

Basic Device Configuration Templates

This section contains a collection of basic design configuration templates. These templates are used for the configuration of basic low level design components that are then applied to the routing devices used in the Juniper Networks business edge solution. The following subsections cover individual tiers of a complete solution configuration:

- Basic Router Configuration
- Core Physical Interface Configuration
- Edge Physical Interface Configuration
- IGP Routing Configuration
- Transport Signaling Configuration
- BGP Routing Configuration
- Class-of-Service Configuration
- Global Service Configuration

Basic Router Configuration

The following sections apply to all functional roles defined in this solution:

- Address Space Configuration
- Network Element Control Plane Protection
- Miscellaneous Settings
- Provider Edge Router Specific Configuration

Address Space Configuration

The template in Table 14 defines various prefix lists that are used later throughout this document.

Table 14: Address Space Configuration Template

Template ID	address-space	Device Role	P and PE device, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	
Parameters			
mgmt-prefix-v4	An array of IPv4 prefixes that refer to the management subnet		
infra-prefix-v4	An array of IPv6 prefixes that refer to the IPv4 infrastructure subnet		
infra-prefix-v6	An array of IPv6 prefixes that refer to the IPv6 infrastructure subnet		
my-as-prefix-v4	An array of IPv4 prefixes that refer to the subnets allocated to our own AS		
my-as-prefix-v6	An array of IPv6 prefixes that refer to the subnets allocated to our own AS		
Configuration Apply Group			

```
policy-options {
    /* All IPv4 address space */
    prefix-list pl-default-v4 {
        0.0.0.0/0;
    }
    /* All IPv6 address space */
    prefix-list pl-default-v6 {
        ::/0;
    }
    /* IPv6 link local addresses */
    prefix-list pl-link-local-v6 {
        fe80::/16;
    }
    /* IPv6 unspecified address */
    prefix-list pl-unspecified-v6 {
        0::/128;
    }
    /* This prefix list holds all management addresses
        (ssh, telnet, ftp may be initiated from those) */
    prefix-list pl-mgmt-v4 {
        ${mgmt-prefix-v4[i]};
    }
    /* This prefix list holds all addresses that cover carrier v4 infrastructure
       (loopbacks, P-P and P-PE links). Targeted LDP, BGP is enabled from those */
    prefix-list pl-infra-v4 {
        ${infra-prefix-v4[i]};
    }
    /* This prefix list holds all addresses that cover carrier v6 infrastructure
       (loopbacks, P-P and P-PE links). Targeted LDP, BGP is enabled from those */
    prefix-list pl-infra-v6 {
        ${infra-prefix-v6[i]};
    }
    /* Prefix list containing all prefixes allocated to this particular AS */
    prefix-list pl-my-as-routes-v4 {
        ${my-as-prefix-v4[i]};
    }
    prefix-list pl-my-as-routes-v6 {
        ${my-as-prefix-v6[i]};
    }
}
```

Network Element Control Plane Protection

As mentioned in the design portion of this document, protecting the infrastructure element can be a complex task, especially the edge network element and when granular, per-attachment circuit control is required. The proposed control plane protection solution provides a balance between configuration complexity and robustness.

This section defines the universal control plane protection filter. This filter is designed to support edge device roles (for example, PE device and AS boundary router roles). However, it can also be applied to P, DCR, RR, and access PE devices (in this case, some of the elements of the template might be redundant).

The constraints that lead to this particular configuration template are the following:

- The same template must be applied to all router loopbacks (to eliminate complexity).
- The template might process traffic that may come from trusted interfaces (management and core interfaces) as well as untrusted interfaces (attachment circuits).
- Trusted and untrusted interfaces might reside at the same line card and on the same Packet Forwarding Engine. Some differentiation is required between traffic coming from those interfaces.
- Edge interface traffic might come in VRF and non-VRF context. VRF addresses may overlap with valid carrier infrastructure addresses.

Typical loopback filter implementation discards control plane or management traffic from non-legitimate sources and allows traffic from legitimate sources. This can be used in conjunction with address spoofing prevention mechanisms such as RPF checks. However, that simple approach will not work for VPN traffic, because VPNs might use address space that overlaps with legitimate sources. Therefore another traffic separation mechanism is required.

In this solution, we employ a two-step approach:

1. For certain traffic types, identify legitimate traffic by its incoming interface and discard other traffic.
2. Verify that the source address of the traffic is valid and belongs to the legitimate source space.

Listing interfaces in a filter is not scalable. However, Junos OS allows users to assign interfaces to one of 255 groups and use the group number as a match condition in the filter. This feature is used in the solution.

Every core interface and management interface is assigned to group 1, with groups 2 and 3 reserved for future use. If an interface belongs to any other group (or default group), this is an edge interface.

The loopback filter discards the following protocol traffic from non-edge interfaces: RSVP, IGMP, DNS, NTP, FTP, SSH, HTTP, RADIUS, TELNET, LDP, and SNMP. It also has different policing parameters configured for control traffic coming from edge and from core interfaces.

The IPv4 filter processing algorithm is provided below (in simplified format):

1. Discard fragmented traffic.
2. Discard invalid IP options and process traffic with IP options (next steps).
3. Discard RSVP traffic from edge interfaces and from invalid sources.
4. Accept IGMP from core interfaces.
5. Accept and limit IGMP from edge interfaces.
6. Accept RSVP from core interfaces and valid sources.
7. Accept MPLS PING from core interfaces and valid sources.
8. Once all traffic with IP options has been processed, discard all other traffic with IP options.
9. Discard DNS, NTP, FTP, HTTP, RADIUS, TELNET, SSH, BGP, LDP, and SNMP from edge interfaces and from invalid sources.
10. Discard BGP nonconforming traffic from TTL-secured neighbors.

NOTE: The assumption here is that GTSM secured sessions are configured in the inet.0 context, and there are no neighbors with the same addresses configured in VPN context (this is a fair assumption, because global address space is typically not reused in VPN context).

11. Rate-limit traceroute packets, ICMP packets, management TCP traffic (SSH, TELNET, FTP), and management UDP traffic (SNMP, DNS, NTP, RADIUS).
12. Accept other TCP, BFD, LDP, and PIM control traffic from core interfaces (without rate limiting).
13. Limit TCP connection attempts from edge interfaces.
14. Limit TCP, BFD, OSPF, and PIM traffic from edge interfaces.
15. Discard other traffic.

The IPv6 filter is simpler, because fewer signaling protocols are involved, but the algorithm follows a similar pattern:

1. Discard IPv6 MTU discovery and BGP packets from non-neighbors. Note that the filter does not completely cover the following case: Two VPNs have overlapping address space, one VPN (VPN A) has a BGP neighbor configured and the other one (VPN B) does not; VPN B hosts might be able to initiate path MTU discovery or BGP packets if their address matches the VPN A BGP neighbor address. This problem is considered minor.
2. Discard BGP nonconforming traffic from TTL-secured neighbors. Note that the assumption here is that GTSM secured sessions are configured in the inet.0 context, and there are no neighbors with the same addresses configured in VPN context (this is a fair assumption, because global address space is typically not reused in VPN context).

3. Rate-limit UDP traceroute, ICMPv6.
4. Accept ICMP traffic from core interfaces.
5. Rate-limit ICMP traffic from edge interfaces.
6. Rate-limit TCP connection attempts from customers.
7. Rate-limit TCP control traffic from customers.
8. Rate-limit PIM, IGMP, hop-by-hop (MLD), and OSPFv3 traffic from customers.
9. Discard everything else.

For each protocol and action, the filter maintains separate counters named as "c-cp-[v4 or v6]-[accept or discard]-[protocol]-[location]"; for example, "c-cp-v4-accept-ospf-ctrl-cust" counts the number of OSPF packets and bytes accepted from customers.

Note that the filter relies on certain naming conventions for the BGP peers. The names for BGP groups should start with prefix v4 for IPv4 neighbors and with prefix v6 for IPv6 neighbors. Neighbors with TTL security enabled should be placed in groups starting with v4-gtsm or v6-gtsm prefixes.

The template in Table 15 defines the loopback filter.

Table 15: Network Element Control Plane Protection

Template ID	loopback-filter	**Device Role**	P and PE router, DCR, AS boundary router, access PE router, route reflector
Service	All	**Guidelines**	
Dependencies			
Constants			
4 Mb bandwidth and 48 Kb burst size limit for customer non-TCP control traffic			
20 Mb bandwidth and 48 Kb burst size limit for customer TCP control traffic			
4 Mb bandwidth and 48 Kb burst size limit for customer TCP connection attempts			
10 Mb bandwidth and 48 Kb burst size limit for management traffic			
5 Mb bandwidth and 48 Kb burst size limit for generic OAM traffic (MPLS pings, pings)			
Configuration Apply Group			

```
groups {
    re0 {
        interfaces {
            /* fxp0 interface belongs to group 1, non-customer facing */
            fxp0 {
                unit 0 {
                    family inet {
                        filter {
                            group 1;
                        }
                    }
                    family inet6 {
                        filter {
                            group 1;
                        }
                    }
                }
            }
        }
    }
  re1 {
        interfaces {
            /* fxp0 interface belongs to group 1, non-customer facing */
            fxp0 {
                unit 0 {
                    family inet {
                        filter {
                            group 1;
                        }
                    }
                    family inet6 {
                        filter {
```

```
                                group 1;                                }
                        }
                    }
                }
            }
        }
        /* Apply group below to the loopback interface */
        loopback-filter {
            interfaces {
                lo0 {
                    /* Apply to all loopbacks, including VPN loopbacks */
                    unit <*> {
                        family inet {
                            filter {
                                input sec-control-plane-v4;
                            }
                        }
                        family inet6 {
                            filter {
                                input sec-control-plane-v6;
                            }
                        }
                    }
                }
            }
        }
    }
    interfaces {
        lo0 {
            apply-groups loopback-filter;
        }
    }
    policy-options {
        /* This prefix list is automatically populated with configured DNS server addresses */
        prefix-list pl-dns {
            apply-path "system name-server <*>";
        }
        /* This prefix list is automatically populated with configured IPv4 inet.0 BGP neighbors,
           note that V4 neighbor is identified by the group name (it should start with v4) */
        prefix-list pl-bgp-inet0-v4 {
            apply-path "protocols bgp group <v4*> neighbor <*>";
        }
        /* This prefix list is automatically populated with configured IPV4 VPN BGP neighbors,
           note that V4 neighbor is identified by the group name (it should start with v4) */
        prefix-list pl-bgp-vpn-v4 {
            apply-path "routing-instances <*> protocols bgp group <v4*> neighbor <*>";
        }
       /* This prefix list is automatically populated with configured IPv6 inet.0 BGP neighbors,
           note that V6 neighbor is identified by the group name (it should start with v6) */
        prefix-list pl-bgp-inet0-v6 {
            apply-path "protocols bgp group <v6*> neighbor <*>";
        }
        /* This prefix list is automatically populated with configured IPv6 VPN BGP neighbors,
           note that V6 neighbor is identified by the group name (it should start with v6) */
        prefix-list pl-bgp-vpn-v6 {
            apply-path "routing-instances <*> protocols bgp group <v6*> neighbor <*>";
        }
        /* This prefix list is automatically populated with configured IPv4 inet.0 BGP neighbors
           that should be enabled for the TTL security check, note that ttl check enabled V4
           neighbor is identified by the group name (it should start with v4-gtsm) */
        prefix-list pl-bgp-inet0-ttl-secured-v4 {
            apply-path "protocols bgp group <v4-gtsm*> neighbor <*>";
        }
```

```
    /* This prefix list is automatically populated with configured IPv6 inet.0 BGP neighbors
       that should be enabled for the TTL security check, note that ttl check enabled V6
       neighbor is identified by the group name (it should start with v6-gtsm) */
    prefix-list pl-bgp-inet0-ttl-secured-v6 {
        apply-path "protocols bgp group <v6-gtsm*> neighbor <*>";
    }
    /* This prefix list is automatically populated with configured RADIUS server addresses */
    prefix-list pl-radius {
        apply-path "system radius-server <*>";
    }
    /* This prefix list is automatically populated with configured NTP peer addresses */
    prefix-list pl-ntp-peer {
        apply-path "system ntp peer <*>";
    }
    /* This prefix list is automatically populated with configured NTP server addresses */
    prefix-list pl-ntp-server {
        apply-path "system ntp server <*>";
    }
}

firewall {
    family inet {
        filter sec-control-plane-v4 {
            /* Discard fragments */
            term discard-first-fragment {
                from {
                    first-fragment;
                }
                then {
                    count c-cp-v4-discard-fragments;
                    log;
                    syslog;
                    discard;
                }
            }
            term discard-other-fragments {
                from {
                    is-fragment;
                }
                then {
                    count c-cp-v4-discard-fragments;
                    log;
                    syslog;
                    discard;
                }
            }
            /* Discard bad options */
            term discard-bad-options {
                from {
                    ip-options [ loose-source-route route-record security timestamp
                                 stream-id strict-source-route ];
                }
                then {
                    count c-cp-v4-discard-bad-options;
                    log;
                    syslog;
                    discard;
                }
            }

           /* Discard and log RSVP packets from non authorized interfaces,
               that includes all DIA/VPN attachment circuits */
            term discard-unauth-rsvp-intf {
                from {
```

```
            interface-group-except [ 1 - 3];
            protocol rsvp;
        }
        then {
            count c-cp-v4-discard-rsvp;
            log;
            syslog;
            discard;
        }
    }

    /* Discard and log RSVP packets from unauthorized sources */
    term discard-unauth-rsvp {
        from {
            source-prefix-list {
                pl-default-v4;
                pl-infra-v4 except;
            }
            protocol rsvp;
        }
        then {
            count c-cp-v4-discard-rsvp;
            log;
            syslog;
            discard;
        }
    }

    /* Next 5 terms might allow traffic with IP Options */
    /* Accept IGMP packets from non-customer facing interfaces, no rate limit enforced */
    term accept-igmp-infra {
        from {
            interface-group 1;
            protocol igmp;
        }
        then {
            count c-cp-v4-accept-igmp-infra;
            accept;
        }
    }
    /* Accept IGMP packets from customer facing interfaces, subject to the rate limit */
    term accept-igmp-customer {
        from {
            interface-group-except [ 1 - 3];
            protocol igmp;
        }
        then {
            policer pol-control-customer;
            count c-cp-v4-accept-igmp-cust;
            accept;
        }
    }
    /* Accept RSVP packets from non-customer facing interfaces */
    term accept-rsvp {
        from {
            /* Allow non-customer facing interfaces only */
            interface-group 1;
            source-prefix-list {
                pl-infra-v4;
            }
            protocol rsvp;
        }
    then {
            count c-cp-v4-accept-rsvp;
            accept;
```

```
        }
    }
    /* Accept MPLS ping from infrastructure addresses */
    term accept-mpls-ping {
        from {
            /* Allow non-customer facing interfaces only */
            interface-group 1;
            source-prefix-list {
                pl-infra-v4;
            }
            protocol udp;
            port 3503;
        }
        then {
            policer pol-mgmt-generic-oam;
            count c-cp-v4-accept-mpls-ping;
            accept;
        }
    }

    /* At this point all eligible traffic with IP Options is processed,
       discard the rest of the traffic with options */
    term discard-options {
        from {
            ip-options any;
        }
        then {
            count c-cp-v4-discard-options;
            log;
            syslog;
            discard;
        }
    }

    /* Discard and log DNS packets from non authorized interfaces,
       that includes all DIA/VPN attachment circuits */
    term discard-unauth-dns-intf {
        from {
            interface-group-except [ 1-3 ];
            protocol udp;
            port domain;
        }
        then {
            count c-cp-v4-discard-dns;
            log;
            syslog;
            discard;
        }
    }
    /* Discard and log DNS response packets from unauthorized sources */
    term discard-unauth-dns {
        from {
            source-prefix-list {
                pl-default-v4;
                pl-dns except;
            }
            protocol udp;
            port domain;
        }
        then {
            count c-cp-v4-discard-dns;
            log;
            syslog;
           discard;
```

```
        }
    }

    /* Discard and log NTP packets from non authorized interfaces,
       that includes all DIA/VPN attachment circuits */
    term discard-unauth-ntp-intf {
        from {
            interface-group-except [ 1-3 ];
            protocol udp;
            port ntp;
        }
        then {
            count c-cp-v4-discard-ntp;
            log;
            syslog;
            discard;
        }
    }
    /* Discard and log NTP packets from unauthorized sources */
    term discard-unauth-ntp {
        from {
            source-prefix-list {
                pl-default-v4;
                pl-ntp-peer except;
                pl-ntp-server except;
            }
            protocol udp;
            port ntp;
        }
        then {
            count c-cp-v4-discard-ntp;
            log;
            syslog;
            discard;
        }
    }

    /* Discard and log FTP packets from non authorized interfaces,
       that includes all DIA/VPN attachment circuits */
    term discard-unauth-ftp-intf {
        from {
            interface-group-except [ 1-3 ];
            protocol tcp;
            port [ ftp ftp-data ];
        }
        then {
            count c-cp-v4-discard-ftp;
            log;
            syslog;
            discard;
        }
    }
    /* Discard and log FTP packets from unauthorized sources */
    term discard-unauth-ftp {
        from {
            source-prefix-list {
                pl-default-v4;
                pl-mgmt-v4 except;
            }
            protocol tcp;
            port [ ftp ftp-data ];
        }
        then {
            count c-cp-v4-discard-ftp;
```

```
                log;
                syslog;
                discard;
            }
        }

        /* Discard and log HTTP packets from non authorized interfaces,
           that includes all DIA/VPN attachment circuits */
        term discard-unauth-web-intf {
            from {
                interface-group-except [ 1-3 ];
                protocol tcp;
                port [ http https ];
            }
            then {
                count c-cp-v4-discard-http;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log http packets from unauthorized sources */
        term discard-unauth-web {
            from {
                source-prefix-list {
                    pl-default-v4;
                    pl-mgmt-v4 except;
                }
                protocol tcp;
                port [ http https ];
            }
            then {
                count c-cp-v4-discard-http;
                log;
                syslog;
                discard;
            }
        }

        /* Discard and log RADIUS packets from non authorized interfaces,
           that includes all DIA/VPN attachment circuits */
        term discard-unauth-radius-intf {
            from {
                interface-group-except [ 1-3];
                protocol udp;
                port [ radacct radius ];
            }
            then {
                count c-cp-v4-discard-radius;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log RADIUS packets from unauthorized sources */
        term discard-unauth-radius {
            from {
                source-prefix-list {
                    pl-default-v4;
                    pl-radius except;
                }
                protocol udp;
                port [ radacct radius ];
            }
```

```
            then {
                count c-cp-v4-discard-radius;
                log;
                syslog;
                discard;
            }
        }

        /* Discard and log TELNET packets from non authorized interfaces,
           that includes all DIA/VPN attachment circuits */
        term discard-unauth-telnet-intf {
            from {
                interface-group-except [ 1-3];
                protocol tcp;
                port telnet;
            }
            then {
                count c-cp-v4-discard-telnet;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log unauthorized telnet login attempts */
        term discard-unauth-telnet {
            from {
                source-prefix-list {
                    pl-default-v4;
                    pl-mgmt-v4 except;
                }
                protocol tcp;
                port telnet;
            }
            then {
                count c-cp-v4-discard-telnet;
                log;
                syslog;
                discard;
            }
        }

         /* Discard SSH packets from non authorized interfaces, that includes all DIA/VPN
           attachment circuits */
        term discard-unauth-ssh-intf {
            from {
                interface-group-except [ 1-3];
                protocol tcp;
                port ssh;
            }
            then {
                count c-cp-v4-discard-ssh;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log unauthorized SSH login attempts */
        term discard-unauth-ssh {
            from {
                source-prefix-list {
                    pl-default-v4;
                   pl-mgmt-v4 except;
               }
               protocol tcp;
```

```
                port ssh;
            }
            then {
                count c-cp-v4-discard-ssh;
                log;
                syslog;
                discard;
            }
        }

        /* Discard and log unauthorized BGP connection attempts */
        term discard-unauth-bgp {
            from {
                source-prefix-list {
                    pl-default-v4;
                    pl-bgp-inet0-v4 except;
                    pl-bgp-vpn-v4 except;
                }
                protocol tcp;
                port bgp;
            }
            then {
                count c-cp-v4-discard-bgp;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log unauthorized BGP connection attempts from
           TTL-secured neighbors. These TTL checks are only implemented for DIA customers */
        term discard-invalid-ttl-secured-sessions {
            from {
                source-prefix-list {
                    pl-bgp-inet0-ttl-secured-v4;
                }
                protocol tcp;
                port bgp;
                ttl-except 255;
            }
            then {
                count c-cp-v4-discard-bgp-invalid-ttl;
                log;
                syslog;
                discard;
            }
        }
        /* Discard SSH packets from non authorized interfaces, that includes all DIA/VPN
           attachment circuits */
        term discard-unauth-ldp-intf {
            from {
                interface-group-except [ 1-3];
                protocol [ tcp udp ];
                port ldp;
            }
            then {
                count c-cp-v4-discard-ldp;
                log;
                syslog;
                discard;
            }
        }
       /* Discard and log LDP packets from unauthorized sources */
        term discard-unauth-ldp {
            from {
```

```
                source-prefix-list {
                    pl-default-v4;
                    pl-infra-v4 except;
                }
                protocol [ tcp udp ];
                port ldp;
            }
            then {
                count c-cp-v4-discard-ldp;
                log;
                syslog;
                discard;
            }
        }

        /* Discard and log SNMP packets from non authorized interfaces,
           that includes all DIA/VPN attachment circuits */
        term discard-unauth-snmp-intf {
            from {
                interface-group-except [ 1-3];
                protocol udp;
                port snmp;
            }
            then {
                count c-cp-v4-discard-snmp;
                log;
                syslog;
                discard;
            }
        }
        /* Discard and log SNMP packets from unauthorized sources */
        term discard-unauth-snmp {
            from {
                source-prefix-list {
                    pl-default-v4;
                    pl-mgmt-v4 except;
                }
                protocol udp;
                port snmp;
            }
            then {
                count c-cp-v4-discard-snmp;
                log;
                syslog;
                discard;
            }
        }
        /* Accept traceroutes */
        term accept-traceroute-udp {
            from {
                protocol udp;
                destination-port 33434-33523;
            }
            then {
                policer pol-mgmt-generic-oam;
                count c-cp-v4-accept-traceroute-udp;
                accept;
            }
        }
        /* Accept ICMP requests */
        term accept-icmp {
            from {
                protocol icmp;
                icmp-type [ echo-reply echo-request time-exceeded unreachable ];
            }
```

```
            then {
                policer pol-mgmt-generic-oam;
                count c-cp-v4-accept-icmp;
                accept;
            }
        }

        /* Accept eligible traffic in this section, and selectively rate limit it
           (set up different limits for customer facing & non-customer facing interfaces) */
        /* Limit TCP management traffic (from non-customer facing interfaces) */
        term limit-tcp-mgmt {
            from {
                interface-group 1;
                source-prefix-list {
                    pl-mgmt-v4;
                }
                protocol tcp;
                port [ ssh telnet ftp ftp-data 1024-65535 ];
            }
            then {
                policer pol-mgmt;
                count c-cp-v4-accept-tcp-mgmt;
                accept;
            }
        }
        /* Limit UDP management traffic (from non-customer facing interfaces) */
        term limit-udp-mgmt {
            from {
                interface-group 1;
                source-prefix-list {
                    pl-mgmt-v4;
                }
                protocol udp;
                port [ snmp domain ntp radius ];
            }
            then {
                policer pol-mgmt;
                count c-cp-v4-accept-udp-mgmt;
                accept;
            }
        }
        /* Accept TCP control traffic from non-customer facing (infrastructure) interfaces
           unconditionally */
        term accept-tcp-control-infra {
            from {
                interface-group 1;
                source-prefix-list {
                    pl-infra-v4;
                }
                protocol tcp;
                port [ ldp bgp ];
            }
            then {
                count c-cp-v4-accept-tcp-ctrl-infra;
                accept;
            }
        }
        /* Accept BFD control traffic from the non-customer facing (infrastructure)
           interfaces unconditionally, includes BFD only for now */
        term accept-bfd-control-infra {
            from {
                interface-group 1;
                source-prefix-list {
                    pl-infra-v4;
```

```
                }
                protocol udp;
                source-port 49152-65535;
                destination-port [ 3784-3785 4784 ];
            }
            then {
                count c-cp-v4-accept-udp-ctrl-infra;
                accept;
            }
        }
        /* Accept LDP discovery control traffic from non-customer facing (infrastructure)
           interfaces unconditionally */
        term accept-ldp-discovery-infra {
            from {
                interface-group 1;
                source-prefix-list {
                    pl-infra-v4;
                }
                protocol udp;
                port ldp;
            }
            then {
                count c-cp-v4-accept-udp-ctrl-infra;
                accept;
            }
        }
        /* Accept PIM from non-customer facing (infrastructure) interfaces
           unconditionally */
        term accept-pim-infra {
            from {
                interface-group 1;
                protocol pim;
            }
            then {
                count c-cp-v4-accept-pim-infra;
                accept;
            }
        }
        /* Limit the rate of TCP connection / reset attempts */
        term limit-tcp-conn-customer {
            from {
                interface-group-except [ 1-3];
                protocol tcp;
                port bgp;
                tcp-flags "(syn & !ack) | fin | rst";
            }
            then {
                policer pol-control-customer-tcp-conn;
                count c-cp-v4-limit-tcp-conn-cust;
                next term;
            }
        }
        /* Limit TCP control traffic from the customer facing interfaces */
        term limit-tcp-control-customer {
            from {
                interface-group-except [ 1-3];
                protocol tcp;
                port bgp;
            }
            then {
                policer pol-control-customer;
                count c-cp-v4-accept-tcp-ctrl-cust;
                accept;
            }
```

```
}
            term limit-bfd-control-customer {
                from {
                    interface-group-except [ 1-3];
                    protocol udp;
                    source-port 49152-65535;
                    destination-port [ 3784-3785 4784 ];
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v4-accept-udp-ctrl-cust;
                    accept;
                }
            }
            term limit-rip-control-customer {
                from {
                    interface-group-except [ 1-3];
                    protocol udp;
                    destination-port rip;
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v4-accept-udp-ctrl-cust;
                    accept;
                }
            }
            term limit-ospf-control-customer {
                from {
                    interface-group-except [ 1-3];
                    protocol ospf;
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v4-accept-ospf-ctrl-cust;
                    accept;
                }
            }
            term limit-pim-control-customer {
                from {
                    interface-group-except [ 1-3];
                    protocol pim;
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v4-accept-pim-ctrl-cust;
                    accept;
                }
            }
            /* Finally, discard and log everything else */
            term discard-unknown {
                then {
                    count c-cp-v4-discard-unknown;
                    log;
                    syslog;
                    discard;
                }
            }
        }
    }
    family inet6 {
        filter sec-control-plane-v6 {
            /* Discard path MTU discovery attempts, except from BGP neighbors */
            term discard-unauth-bgp-mtu-discovery {
                from {
                    source-prefix-list {
```

```
                pl-default-v6;
                pl-bgp-inet0-v6 except;
                pl-bgp-vpn-v6 except;
            }
            next-header icmpv6;
            icmp-type packet-too-big;
        }
        then {
            count c-cp-v6-discard-mtu-discovery;
            log;
            syslog;
            discard;
        }           }
    /* Discard BGP packets form every host except a configured BGP peer */
    term discard-unauth-bgp {
        from {
            source-prefix-list {
                pl-default-v6;
                pl-bgp-inet0-v6 except;
                pl-bgp-vpn-v6 except;
            }
            next-header tcp;
            port bgp;
        }
        then {
            count c-cp-v6-discard-bgp;
            log;
            syslog;
            discard;
        }
    }
    /* Discard and log unauthorized BGP connection attempts from
       TTL-secured neighbors.
       The assumption is that VPN space should not have any overlap
       with the TTL secured neighbor addresses. */
    term discard-invalid-ttl-secured-sessions {
        from {
            source-prefix-list {
                pl-bgp-inet0-ttl-secured-v6;
            }
            next-header tcp;
            port bgp;
            hop-limit-except 255;
        }
        then {
            count c-cp-v6-discard-bgp-invalid-ttl;
            log;
            syslog;
            discard;
        }
    }
    /* Accept traceroute */
    term accept-traceroute-udp {
        from {
            next-header udp;
            destination-port 33434-33523;
        }
        then {
            policer pol-mgmt-generic-oam;
            count c-cp-v6-accept-traceroute-udp;
            accept;
        }
    }
```

```
            /* Accept ICMP PINGs, etc, except address resolution */
            term accept-icmp {
                from {
                    next-header icmpv6;
                    icmp-type [ echo-request echo-reply destination-unreachable
                                time-exceeded packet-too-big ];
                }
                then {
                    policer pol-mgmt-generic-oam;
                    count c-cp-v6-accept-icmp;
                    accept;
                }
            }
            /* Accept ICMP neighbor solicitation & MLD */
            term accept-icmp-control-infra {
                from {
                    interface-group 1;
                    next-header icmpv6;
                    icmp-type [ neighbor-advertisement neighbor-solicit router-solicit
                                router-advertisement membership-query membership-report
                                membership-termination ];
                }
                then {
                    count c-cp-v6-accept-icmp-ctrl-infra;
                    accept;
                }
            }
            /* Accept hop by hop extension headers */
            term accept-icmp-control-infra {
                from {
                    /* Only accept them from core interfaces */
                    interface-group 1;
                    next-header hop-by-hop;
                }
                then {
                    count c-cp-v6-accept-hbh-ctrl-infra;
                    accept;
                }
            }
            /* Limit ICMP neighbor solicitation & MLD from the customer interfaces */
            term limit-icmp-control-customer {
                from {
                    interface-group-except [ 1-3];
                    next-header icmpv6;
                    icmp-type [ neighbor-advertisement neighbor-solicit router-solicit
                                router-advertisement membership-query membership-report
                                membership-termination ];
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v6-accept-icmp-ctrl-cust;
                    accept;
                }
            }
            /* Limit the rate of TCP connection / reset attempts */
            term limit-tcp-conn-customer {
                from {
                    interface-group-except [ 1-3];
                    next-header tcp;
                    port bgp;
                    tcp-flags "(syn & !ack) | fin | rst";
                }
                then {
```

```
                policer pol-control-customer-tcp-conn;
                count c-cp-v6-limit-tcp-conn-cust;
                next term;
            }
        }
        /* Limit TCP control packets from the customer facing interfaces */
        term limit-tcp-control-customer {
            from {
                interface-group-except [ 1-3];
                next-header tcp;
                port bgp;
            }
            then {
                policer pol-tcp-control-customer;
                count c-cp-v6-accept-tcp-ctrl-cust;
                accept;
            }
        }
        /* Limit PIM control packets from the customer facing interfaces */
        term limit-pim-control-customer {
            from {
                interface-group-except [ 1-3];
                next-header pim;
            }
            then {
                policer pol-control-customer;
                count c-cp-v6-accept-pim-ctrl-cust;
                accept;
            }
        }
        /* Limit IGMP control packets from the customer facing interfaces */
        term limit-igmp-control-customer {
            from {
                interface-group-except [ 1-3];
                next-header igmp;
            }
            then {
                policer pol-control-customer;
                count c-cp-v6-accept-igmp-ctrl-cust;
                accept;
            }
        }
        /* Limit hop-by-hop control packets from the customer facing interfaces
           they are used for MLD, which is in turn used by OSPF */
        term limit-hbh-control-customer {
            from {
                source-prefix-list {
                    pl-link-local-v6;
                    pl-unspecified-v6;
                }
                interface-group-except [ 1-3];
                next-header hop-by-hop;
            }
            then {
                policer pol-control-customer;
                count c-cp-v6-accept-hbh-ctrl-cust;
                accept;
            }
        }
        /* Limit OSPF control packets from the customer facing interfaces */
        term limit-ospf-control-customer {
            from {
                interface-group-except [ 1-3];
                next-header ospf;
```

```
                }
                then {
                    policer pol-control-customer;
                    count c-cp-v6-accept-ospf-ctrl-cust;
                    accept;
                }
            }
            /* Discard unknown traffic, log the packet */
            term discard-unknown {
                then {
                     count c-cp-v6-discard-unknown;
                    log;
                    syslog;
                    discard;                }
            }
        }
    }
    policer pol-control-customer {
        filter-specific;
        /* Allow 4Mbps of non-TCP packets per second,
           For example, this policer may accept up to ~7K 62 byte BFD packets per second,
           It is shared between all non-tcp protocols */
        if-exceeding {
            bandwidth-limit 4m;
            burst-size-limit 48k;
        }
        then discard;
    }
    policer pol-tcp-control-customer {
        filter-specific;
        /* Allow 20 Mbps of TCP packets per second */
        if-exceeding {
            bandwidth-limit 20m;
            burst-size-limit 48k;
        }
        then discard;
    }
    policer pol-control-customer-tcp-conn {
        filter-specific;
        /* Allow 4Mbps of TCP packets per second, this equals to 10416 TCP SYNs / second,
           1000 packets in a burst */
        if-exceeding {
            bandwidth-limit 4m;
            burst-size-limit 48k;
        }
        then discard;
    }
    policer pol-mgmt {
        filter-specific;
        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 48k;
        }
        then discard;
    }
    policer pol-mgmt-generic-oam {
        filter-specific;

        if-exceeding {
            bandwidth-limit 5m;
            burst-size-limit 48k;
        }
        then discard;
    }
}
```

Miscellaneous Settings

It is a good idea to enable as many aggregated devices as might be required through the router lifetime. This value is typically in order of dozens for P routers, access PE routers, and firewall/NAT devices, and in the hundreds for large-scale PE, DCR, and AS boundary routers.

The template in Table 16 is a collection of the configuration elements common to all device roles. This template performs the following:

- Configures aggregate Ethernet devices
- Enables per-flow load balancing

Table 16: Miscellaneous Settings Configuration (Aggregate Ethernet Devices and Per-Flow Load Balancing)

Template ID	router-global	**Device Role**	P and PE router, DCR, AS boundary router, access PE router, firewall/NAT, route reflector
Service	All	**Guidelines**	
Parameters			
ae_count	Aggregated Ethernet device count		

```
chassis {
    aggregated-devices {
        ethernet {
            device-count ${ae_count};
        }
    }
}

routing-options {
    forwarding-table {
        export pol-load-balance;
    }
}

policy-options {
    policy-statement pol-load-balance {
        then {
            /* Despite the per-packet name, the balancing is actually done per-flow on all
               modern Juniper platforms */
            load-balance per-packet;
        }
    }
}
```

On PE and DCR routers, a few additional global configuration statements must be enabled. The template in Table 17 defines the configuration of those global statements.

Table 17: Miscellaneous Settings Configuration for PE/DCR Routers

Template ID	pe-global	**Device Role**	PE and DCR routers
Service	All	**Guidelines**	

```
routing-options {
    forwarding-table {
        /* Use all feasible paths for RPF checks */
        unicast-reverse-path feasible-paths;
        /* Enable chained composite next-hop support for direct PE PE router case
           (single next-hop)*/
        chained-composite-next-hop {
            ingress {
                l3vpn pe-pe-connection extended-space;
            }
        }
    }
```

The “unicast-reverse-path feasible-paths” commands direct the router to perform a unicast RPF check by using all feasible paths (which are not being used for forwarding). This is required to support CE device dual-homing scenarios with unequal cost routes. For example, the CE device might signal to the PE router to set the local preference of that route to some lower value (see DIA policy). In this case, the route through another PE device will be used for forwarding traffic to the CE device. Still, the CE device might prefer to use both links to forward traffic towards the network. Typical RPF check behavior would result in discards on the link with a less preferred route towards the CE device. This command modifies unicast RPF check behavior to include feasible paths into consideration for RPF checks.

The second command enables a chained composite next-hop creation for a corner case where PE routers are connected directly. Chained composite next hops significantly improve convergence and reduce memory footprint. The **pe-pe-connection** command is supported only on Trio line cards and must be configured explicitly in order to indicate to Junos OS that such an operation is requested.

Provider Edge Router Specific Configuration

The template in Table 18 defines the PIC queuing configuration. This template should be applied to all PIC locations that might facilitate queuing.

Table 18: Edge Router—PIC Queuing Configuration

Template ID	edge-chassis-pic-queuing	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	
Parameters			
fpc	FPC location for the PIC with ingress or ingress and egress queuing enabled		
pic	PIC location with ingress or ingress and egress queuing enabled		
mode	Set to ingress-and-egress if ingress queuing or shaping is required; set to egress-only if egress queuing is required		

```
chassis {
    fpc ${fpc} {
        pic ${pic} {
            traffic-manager {
                mode ${mode};
            }
        }
    }
}
```

Core Physical Interface Configuration

The template in Table 19 defines the core physical interface configuration. It is applicable to all routers that might have connectivity to the network core (in this solution, that means everything except firewall/NAT devices).

This template specifically configures Ethernet interfaces. The template sets the MTU; enables IPv4, IPv6, MPLS, and ISO address families; and assigns interface IP addresses. Most of the common core interface settings are applied through configuration group **core-intf-settings**, enabling the ability to modify all core interface configuration at once if necessary.

Table 19: Core Physical Interface Configuration

Template ID	core-intf	**Device Role**	P and PE router, DCR, AS boundary router, access PE router, route reflector
Service	All	**Guidelines**	This configuration assumes that untagged Ethernet core interfaces are in use
Constants			
MTU is set to 9,192 (maximum)			
Parameters			
core_ifd	Core-facing physical interface name		
router_ipv4	Interface IPv4 address		
router_ipv6	Interface IPv6 address		
description	Interface description		

```
core-intf-settings {
    interfaces {
        <*> {
            /* MTU is set to maximum */
            mtu 9192;
            unit <*> {
                family inet {
                    filter {
                        /* Note, core interfaces are assigned group 1. This is used in the
                           loopback filter */
                        group 1;
                    }
                }
                /* Enable ISO family for IS-IS */
                family iso;
                family inet6 {
                    filter {
                        /* Note, core interfaces are assigned group 1. This is used later in
                           the loopback filter */
                        group 1;
                    }
                }
                /* Enable MPLS family */
                family mpls {
                    filter {
                        /* Note, core interfaces are assigned group 1. This is used later in
                           the loopback filter */
                        group 1;
                    }
                }
            }
        }
    }
}
interfaces {
    ${core_ifd} {
        /* Apply core-intf settings */
        apply-groups core-intf-settings;
        description ${core_description};
        /* Assuming that only one unit is enabled on the core interface */
        unit 0 {
            family inet {
                address ${router_ipv4};
            }
            family inet6 {
                address ${router_ipv6};
            }
        }
    }
}
```

It is important to note that the core interface is assigned to interface group 1. This means that packets received from the core interfaces are tagged internally as belonging to that interface group, which can be used in firewall filter match conditions. In this solution, the interface group is used in the loopback filter to provide different treatment to traffic coming from the network core (trusted interface) as opposed to customer attachment circuits (untrusted interfaces).

Edge Physical Interface Configuration

The template in Table 20 defines configuration for untagged Ethernet interfaces. The template configures only the physical interface MTU value.

Table 20: Edge Physical Interface Configuration

Template ID	edge-eth-phys-intf-untagged	Device Role	PE router, DCR, AS boundary router
Service	All	Guidelines	Untagged interface
Parameters			
cust_ifd	Physical interface name		
description	Interface description		

```
groups {
    edge-intf-untagged {
        interfaces {
            <*> {
                mtu 9118;
            }
        }
    }
}
interfaces {
    ${cust_ifd} {
        /* Apply edge-intf-untagged settings */
        apply-groups edge-intf-untagged;
        description ${description};
    }
}
```

The template in Table 21 defines configuration for tagged Ethernet interfaces. Single tagged interface configuration is slightly different. The MTU is increased by 4 bytes, plus VLAN tagging is enabled.

Table 21: Edge Tagged Interface Configuration—MTU and VLAN Tagging

Template ID	edge-eth-phys-intf-untagged	Device Role	PE router, DCR, AS boundary router
Service	All	Guidelines	Single tagged interface
Parameters			
cust_ifd	Physical interface name		
description	Interface description		

```
groups {
    edge-intf-tagged {
        interfaces {
            <*> {
                mtu 9122;
            }
        }
    }
}
interfaces {
    ${cust_ifd} {
        /* Apply edge-intf-untagged settings */
        apply-groups edge-intf-tagged;
        vlan-tagging;
        description ${description};
    }
}
```

Dual-tagged interface configuration follows the same principle. The MTU is increased by 4 additional bytes, ensuring a consistent IP MTU (9,100) across all interface types. Note the usage of the **flexible-vlan-tagging** keyword. This keyword enables termination of both single tagged and dual-tagged attachment circuits on the same interface.

The template in Table 22 defines configuration for dual-tagged Ethernet interfaces.

Table 22: Edge Dual-Tagged Interface Configuration

Template ID	edge-eth-phys-intf-double-tagged	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	Dual-tagged and single tagged
Parameters			
cust_ifd	Physical interface name		
description	Interface description		

```
groups {
    edge-intf-double-tagged {
        interfaces {
            <*> {
                mtu 9126;
            }
        }
    }
}
interfaces {
    ${cust_ifd} {
        /* Apply edge-intf-double-tagged settings */
        apply-groups edge-intf-double-tagged;
        flexible-vlan-tagging;
        description ${description};
    }
}
```

Some of the CoS profiles defined in the class-of-service section of this document rely on the per-VLAN queuing and shaping support, as well as hierarchical queuing (at the level of the individual interface and an interface group). This functionality should be enabled at the line-card level (see edge-chassis-pic-queuing) and at the interface level. The template in Table 23 is used to enable queuing at the interface level.

Table 23: Edge Ethernet Interface Queuing

Template ID	edge-eth-intf-queuing	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	Interfaces that have class-of-service profiles with queuing or shaping
Parameters			
cust_ifd	Physical interface name		

```
interfaces {
    ${cust_ifd} {
        hierarchical-scheduler;
    }
}
```

Aggregate Ethernet interfaces are configured a bit differently. Physical interface addresses are bundled in an aggregated interface and router IPv4 or IPv6 addresses are configured at the aggregated Ethernet interface level. Table 24 is used to configure aggregated Ethernet interface bundling.

Table 24: Aggregated Ethernet Interface Configuration

Template ID	agg-eth-intf	Device Role	P and PE routers, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	This configuration assumes that untagged core interfaces are in use
Parameters			
ae_ifd	Aggregated interface name		
member_link_ifd	An array of physical interface member links		

```
interfaces {
    interface ${member_ifd[i]} {
        gigether-options {
            802.3ad ${ae_ifd};
        }
    }
    ${ae_ifd} {
        unit 0;
    }
}
```

Ethernet OAM (link fault management) is enabled on all core interfaces and can be enabled on some of the edge interfaces. The template in Table 25 is used to configure link fault management (LFM).

NOTE: Once LFM adjacency loss is detected, the member link is declared down and it is not used for forwarding until LFM adjacency reestablishes. This behavior is implemented by configuring an action profile.

Table 25: Aggregated Ethernet Link Fault Management Configuration

Template ID	agg-eth-intf-lfm	Device Role	P and PE routers, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	This configuration assumes that untagged core interfaces are in use
Constants			
LFM PDU interval is set to 100 ms.			
The member link is declared down after three LFM PDUs are missed (pdu-threshold).			
Parameters			
member_ifd	An array of physical interface member links		

```
groups {
    oam-settings {
        protocols {
            oam {
                ethernet {
                    link-fault-management {
                        interface <*> {
                            apply-action-profile ap-intf-down;
                            pdu-interval 100;
                            pdu-threshold 3;
                        }
                    }
                }
            }
        }
    }
}
protocols {
    oam {
        ethernet {
            apply-groups oam-settings;
```

```
            link-fault-management {
                action-profile ap-intf-down {
                    event {
                        link-adjacency-loss;
                    }
                    action {
                        link-down;
                    }
                }
                /* every member interface should be listed below*/
                interface ${member_ifd[i]};
            }
        }
    }
}
```

IGP Routing Configuration

The template in Table 26 implements design goals presented in the "Core Design" section. This template is intended for initially configuring a network device.

Table 26: Interior Gateway Protocol Routing Configuration

Template ID	isis-common	Device Role	P and PE routers, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	
Parameters			
isis_auth_key	IS-IS authentication key		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
groups {
    isis-interface-settings {
        protocols {
            isis {
                interface <*> {
                    point-to-point;
                    bfd-liveness-detection {
                        minimum-interval ${bfd_min_interval};
                        multiplier ${bfd_multiplier};
                    }
                    level 2 {
                        hello-authentication-key ${isis_auth_key};
                        hello-authentication-type md5;
                    }
                }
            }
        }
    }
}

protocols {
    isis {
        apply-groups isis-interface-settings;
        level 1 disable;
        level 2 wide-metrics-only;
        interface lo0.0;
    }
}
```

In Table 26, IS-IS Level 1 is disabled. The IGP design is flat, meaning that there is no need to maintain two levels. Wide metrics are implemented at Level 2. A configuration group is configured to apply common interface settings such as authentication and BFD configuration.

The template in Table 27 illustrates core interface IS-IS configuration.

Table 27: Core Interface IS-IS Configuration

Template ID	isis-intf	Device Role	P and PE router, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	
Parameters			
core_ifd	Physical port of the core interface		
core_unit	Unit number of the core interface (typically 0)		
metric	IS-IS metric		

```
protocols {
    isis {
        interface ${core_ifd}.${core_unit} {
            level 2 metric ${metric};
        };
    }
}
```

The interface is listed under the IS-IS hierarchy and the Level 2 metric is set. How metrics are selected is outside of the scope of this solution. The service provider may choose different strategies to assign metrics (for example, to reflect bandwidth capacity or latency).

Transport Signaling Configuration

The template in Table 28 provides MPLS processing and RSVP-TE transport signaling interface configuration.

Table 28: Transport Signaling Configuration

Template ID	mpls-rsvp-te-common	Device Role	P and PE routers, DCR, AS boundary router, access PE router, route reflector
Service	All	Guidelines	
Constants			
Maximum interface subscription is set to 95 percent.			
Parameters			
core_ifd	Physical port of the core interface		
core_unit	Unit number of the core interface (typically 0)		

```
groups {
    rsvp-interface-settings {
        protocols {
            rsvp {
                interface <*> {
                    aggregate;
                    subscription 95;
                    link-protection;
                }
            }
        }
    }
}
protocols {
    rsvp {
        apply-groups rsvp-interface-settings;
        interface ${core_ifd}.${core_unit};
    }
    mpls {
        interface ${core_ifd}.${core_unit};
    }
}
```

The interfaces are listed explicitly. There is an alternative syntax available to enable RSVP-TE and MPLS on all interfaces with the MPLS address family configured (**interface all**). However, to enable individual interface MPLS and RSVP-TE parameter tuning if the need should arise, explicit listing is used instead.

The template in Table 29 is used to provision a set of 16 RSVP-TE LSPs between two routers. The template sets certain configuration parameters that enable auto-bandwidth adjustments on those LSPs.

Table 29: MPLS RSVP-TE Mesh Configuration

Template ID	mpls-rsvp-te-mesh	**Device Role**	PE router, DCR, AS boundary router, access PE router
Service	All	**Guidelines**	
Constants			
MPLS LSP statistics filename is “mpls-lsp.stat.”			
Size of the MPLS LSP statistics file is set to 16 Mb with 3 files maximum.			
MPLS LSP statistics collection interval is set to 300 seconds.			
MPLS LSP adjustment interval is set to 10,800 seconds or 3 hours.			
MPLS LSP adjustment threshold is set to 5 percent.			
MPLS LSP maximum bandwidth is set to 8 Gbps.			
MPLS LSP adjustment threshold overflow limit is set to 3 (number of consecutive samples when overflow condition occurs that triggers premature bandwidth adjustment).			
Parameters			
my_name	Hostname of this router		
target_name	Hostname of the RSVP-TE LSP target		
my_lo0_address	IPv4 address of this router loopback interface		
target_lo0_address	IPv4 address of the target router loopback interface		

```
groups {
    p2p-mpls-lsp-settings {
        protocols {
            mpls {
                label-switched-path <p2p-*> {
                    node-link-protection;
                    adaptive;
                    auto-bandwidth {
                        adjust-interval 10800;
                        adjust-threshold 5;
                        maximum-bandwidth 8g;
                        adjust-threshold-overflow-limit 3;
                    }
                }
            }
        }
    }
}

protocols {
    mpls {
        statistics {
            file mpls-lsp.stat size 16m files 3;
            interval 300;
            auto-bandwidth;
        }
        no-propagate-ttl;
        ipv6-tunneling;
        apply-groups p2p-mpls-lsp-settings;
        label-switched-path p2p-${my_name}-${target_name}-01 {
            from ${my_lo0_address};
            to ${target_lo0_address};
       }
```

```
        label-switched-path p2p-${my_name}-${target_name}-02 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-03 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-04 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-05 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-06 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-07 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-08 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-09 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-10 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-11 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-12 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-13 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-14 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-15 {
            from ${my_lo0_address};
            to ${target_lo0_address};
        }
        label-switched-path p2p-${my_name}-${target_name}-16 {
            from ${my_lo0_addres};
            to ${target_lo0_address};
        }
    }
}
```

You must use the template in Table 29 to configure a full mesh of LSPs among PE, DCR, and AS boundary routers in the network. There is also an assumption made that RSVP-TE LSPs have been established between the MPLS access PE router and the PE router. However, this might not be the case in typical deployments. It depends on the access and aggregation network design.

Other settings enabled in this template include the following:

- Using the **no-propagate-ttl** statement ensures that the TTL field value is not copied from the IP header to the MPLS shim header; this hides the service provider infrastructure from traceroutes.
- Using the **ipv6-tunneling** statement results in the installation of IPv6 mapped IPv4 addresses from the inet.3 table to the inet6.3 table. This installation results in enabling the resolution of IPv6 protocol next-hop addresses.

Route reflectors are configured differently than PE, DCR, or AS boundary routers. To support VPN route reflection, protocol next hops for VPN routes should be resolved to some valid next hop. One method of accomplishing this would be to configure a mesh of LSPs between reflectors and PE devices. However, these LSPs are not used for traffic forwarding, because route reflectors only propagate routes without changing the protocol next hop in this solution. Because of this function, the configuration can be simplified.

The template in Table 30 provides a configuration for use on dedicated route reflectors. In this configuration, two static routes are installed into the inet.3 and inet6.3 routing tables, with the only goal being to enable route resolution.

Table 30: Route Reflector Configuration

Template ID	mpls-rr	**Device Role**	Route reflector
Service	All	**Guidelines**	Use only on dedicated route reflectors

```
routing-options {
    rib inet.3 {
        static {
            route 0.0.0.0/0 discard;
        }
    }
    rib inet6.3 {
        static {
            route 0::0/0 discard;
        }
    }
}
```

BGP Routing Configuration

BGP routing is configured differently for PE devices and route reflectors.

The template in Table 31 provides BGP configuration for PE devices. In this template, we assume that each PE device peers with multiple (typically two) route reflectors. Normally, peering with two route reflectors would require just two IBGP sessions with all address families enabled on both. However, as of now there is no nonstop routing support for some of the address families (such as MPVN). If a Routing Engine switchover occurs, the session with the MPVN family goes down, and this action ultimately results in traffic loss. To minimize any adverse impact from connection loss, two sessions are configured between the peers. One session is used to support NSR-enabled address families and the other session is used to support non-NSR enabled address families. During any switchover event, only the latter session goes down.

Table 31: Edge BGP Routing Configuration

Template ID	pe-ibgp	Device Role	PE router, DCR, AS boundary router
Service	All	Guidelines	
Dependencies	dia-policy, dia-community		
Parameters			
kc_secret	Authentication key chain password		
kc_start_time	Authentication key chain activation time		
as	My autonomous system number		
lo0_v4_non_nsr	Secondary IPv4 address of this router loopback interface for IBGP session with non-NSR enabled families		
rr_v4_nsr	An array of primary IPv4 addresses of route reflector loopback interfaces for IBGP session with all NSR-enabled families		
rr_v4_non_nsr	An array of secondary IPv4 addresses of route reflector loopback interfaces for IBGP session with non-NSR enabled families		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
security {
    authentication-key-chains {
        key-chain kc-ibgp {
            key 0 {
                secret ${kc_secret};
                start-time ${kc_start_time};
            }
        }
    }
}
protocols {
    bgp {
        group v4-ibgp-nsr-enabled {
            type internal;
            path-selection external-router-id;
            /* Export all customer routes to RRs, including directs & statics, but not
               aggregates */
            export [ pol-export-directs pol-export-ibgp pol-next-hop-self pol-final ];
            local-address ${lo0_v4_nsr};
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family inet6 {
                unicast;
            }
            family inet6-vpn {
                unicast;
            }
            family route-target;
            bfd-liveness-detection {
                minimum-interval ${bfd_min_interval};
                multiplier ${bfd_multiplier};
            }
            multipath multiple-as;
            neighbor ${rr_v4_nsr[i]} {
                authentication-key-chain kc-ibgp;
            };
        }
```

```
        group v4-ibgp-non-nsr-enabled {
            type internal;
            path-selection external-router-id;
            /* Export all customer routes to RRs, including directs & statics, but not
               aggregates */
            export [ pol-export-directs pol-export-ibgp pol-next-hop-self pol-final ];
            local-address ${lo0_v4_non_nsr};
            family inet-mvpn {
                signaling;
            }
            family inet6-mvpn {
                signaling;
            }
            bfd-liveness-detection {
                minimum-interval ${bfd_min_interval};
                multiplier ${bfd_multiplier};
            }
            multipath multiple-as;
            neighbor ${rr_v4_non_nsr[i]} {
                authentication-key-chain kc-ibgp;
            };
        }
    }
}

policy-options {
    policy-statement pol-next-hop-self {
        from {
           protocol bgp;
           route-type external;
        }
        then next-hop self;
    }
    policy-statement pol-export-ibgp {
        /* This term skips aggregate routes */
        term skip-aggregates {
            from {
                protocol aggregate;
            }
            then {
                next policy;
            }
        }
        /* This term selects all BGP and Static routes marked with specific communities */
        term select-all {
            from community [ comm-all-peers comm-all-customers comm-all-upstreams ];
            then {
                community add comm-accept;
                next policy;
            }
        }
    }
}

routing-options {
    autonomous-system ${as};
}
```

In the template in Table 31, all route reflector neighbors should be listed in respective BGP groups.

Certain BGP export policies are configured to distribute local routes toward route reflectors.

For Direct Internet Access service (inet address family), all direct, peer, transit, and customer routes are distributed.

NOTE: The **next-hop** attribute is set to **self** for EBGP external routes only (and not for all exported routes). The reason for this setting is to support route reflector functionality being merged with the PE router functionality. In the route reflector case, some of the exported routes are reflected. For those routes, protocol next hops remain unchanged.

The rest of the configuration of BGP sessions follow the Junos OS software default with the exception of the path selection algorithm. To ensure consistent path selection before or after Routing Engine switchover events, the BGP path selection algorithm is configured to compare external router identifiers.

The template in Table 32 pertains to route reflectors. The configuration mirrors IBGP configuration on the PE router side with only the following exceptions:

- Route reflection is enabled (cluster id is set).
- The default route for the **route-target** family is advertised to signal to the peer that all VPN routes be sent to the reflector.

The rest of the configuration remains the same, including export policy configuration. In a dedicated route reflector case, you can simplify this part of the configuration. A dedicated route reflector need not advertise its direct routes, because it does not have any attachment circuits configured. In addition, a dedicated route reflector might just export all BGP routes without changing the next hop for EBGP routes. In this solution, however, a generic assumption is made that route reflectors and PE device functions can be merged, negating the need to implement those optimizations.

Table 32: Configuration for Merged Route Reflector and PE Routers

Template ID	rr-ibgp	**Device Role**	Route reflector
Service	All	**Guidelines**	May be used on route reflectors with other functions (AS boundary router, DCR, PE router)
Dependencies	dia-policy, dia-community		
Parameters			
kc_secret	Authentication key chain password		
kc_start_time	Authentication key chain activation time		
as	My autonomous system number		
lo0_v4_nsr	Primary IPv4 address of this router loopback interface for IBGP session with all NSR-enabled families		
lo0_v4_non_nsr	Secondary IPv4 address of this router loopback interface for IBGP session with non-NSR enabled families		
client_v4_nsr	Primary IPv4 address of the client I loopback interface for IBGP session with all NSR-enabled families		
client_non_nsr	Secondary IPv4 address of client I loopback interface for IBGP session with non-NSR enabled families		
cluster_id	IPv4 address identifying the route reflector cluster		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
security {
    authentication-key-chains {
        key-chain kc-ibgp {
            key 0 {
                secret ${kc_secret};
                start-time ${kc_start_time};
            }
        }
    }
}
protocols {
    bgp {
        group v4-ibgp-clients-nsr-enabled {
            type internal;
            path-selection external-router-id;
            /* Export all customer routes to RRs, including directs & statics, but not
               aggregates */
            export [ pol-export-directs pol-final ];
            local-address ${lo0_v4_nsr};
            family inet {
                unicast;
            }
```

```
            family inet-vpn {
                unicast;
            }
            family inet6 {
                unicast;
            }
            family inet6-vpn {
                unicast;
            }
            /* Route Reflector should receive all VPN routes */
            family route-target {
                advertise-default;
            }
            cluster ${cluster_id};
            bfd-liveness-detection {
                minimum-interval ${bfd_min_interval};
                multiplier ${bfd_multiplier};
            }
            multipath multiple-as;
            neighbor ${client_v4_nsr[i]} {
                authentication-key-chain kc-ibgp;
            };
        }
        group v4-ibgp-clients-non-nsr-enabled {
            type internal;
            path-selection external-router-id;
            /* Export all customer routes to RRs, including directs & statics, but not
               aggregates */
            export [ pol-export-directs pol-final ];
            local-address ${lo0_v4_non_nsr};
            family inet-mvpn {
                signaling;
            }
            family inet6-mvpn {
                signaling;
            }
            cluster ${cluster_id};
            bfd-liveness-detection {
                minimum-interval ${bfd_min_interval};
                multiplier ${bfd_multiplier};
            }
            multipath multiple-as;
            neighbor ${client_v4_non_nsr[i]} {
                authentication-key-chain kc-ibgp;
            };
        }
    }
}

policy-options {
    policy-statement pol-next-hop-self {
        from {
           protocol bgp;
           route-type external;
        }
        then next-hop self;
    }
    policy-statement pol-export-ibgp {
        /* This term skips aggregate routes */
        term skip-aggregates {
            from {
                protocol aggregate;
            }
            then {
```

```
                next policy;
            }
        }
        /* This term selects all BGP and Static routes marked with specific communities */
        term select-all {
            from community [ comm-all-peers comm-all-customers comm-all-upstreams ];
            then {
                community add comm-accept;
                next policy;
            }
        }
    }
}

routing-options {
    autonomous-system ${as};
}
```

Class-of-Service Configuration

The template in Table 33 defines common configuration for all router types used in the solution.

Table 33: Common Class-of-Service Configuration

Template ID	cos-common	Device Role	All
Service	All	Usage Guidelines	None

```
class-of-service {
   classifiers {
        dscp all-be-v4 {
            forwarding-class fc-be {
                loss-priority low code-points [ 000000 000001 000010 000011 000100 000101 000110
                    000111 001000 001001 001010 001011 001100 001101 001110 001111 010000 010001
                    010010 010011 010100 010101 010110 010111 011000 011001 011010 011011 011100
                    011101 011110 011111 100000 100001 100010 100011 100100 100101 100110 100111
                    101000 101001 101010 101011 101100 101101 101110 101111 110000 110001 110010
                    110011 110100 110101 110110 110111 111000 111001 111010 111011 111100 111101
                    111110 111111 ];
            }
        }
        dscp core-v4 {
            import all-be-v4;
            forwarding-class fc-nc {
                loss-priority low code-points [ 110000 111000 ];
            }
        }
        dscp-ipv6 all-be-v6 {
            forwarding-class fc-be {
                loss-priority low code-points [ 000000 000001 000010 000011 000100 000101 000110
                    000111 001000 001001 001010 001011 001100 001101 001110 001111 010000 010001
                    010010 010011 010100 010101 010110 010111 011000 011001 011010 011011 011100
                    011101 011110 011111 100000 100001 100010 100011 100100 100101 100110 100111
                    101000 101001 101010 101011 101100 101101 101110 101111 110000 110001 110010
                    110011 110100 110101 110110 110111 111000 111001 111010 111011 111100 111101
                    111110 111111 ];
            }
        }
        dscp-ipv6 core-v6 {
            import all-be-v6;
            forwarding-class fc-nc {
                loss-priority low code-points [ 110000 111000 ];
            }
        }
```

```
exp core {
            forwarding-class fc-be {
                loss-priority low code-points 000;
                loss-priority high code-points 001;
            }
            forwarding-class fc-ef {
                loss-priority low code-points 010;
                loss-priority high code-points 011;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points 100;
                loss-priority high code-points 101;
            }
            forwarding-class fc-nc {
                loss-priority low code-points 110;
                loss-priority high code-points 111;
            }
        }
    }
    host-outbound-traffic {
        forwarding-class fc-nc;
    }
    drop-profiles {
        dp-aggressive {
            interpolate {
                fill-level [ 50 75 100 ];
                drop-probability [ 0 50 100 ];
            }
        }
        dp-moderate {
            interpolate {
                fill-level [ 75 82 100 ];
                drop-probability [ 0 50 100 ];
            }
        }
    }
    forwarding-classes {
        class fc-be queue-num 0;
        class fc-ef queue-num 1;
        class fc-af1 queue-num 2;
        class fc-nc queue-num 3;
        class fc-af2 queue-num 4;
        class fc-af3 queue-num 5;
        class fc-af4 queue-num 6;
        class fc-af-extra queue-num 7;
    }
    rewrite-rules {
        exp core {
            forwarding-class fc-be {
                loss-priority low code-point 000;
                loss-priority high code-point 001;
            }
            forwarding-class fc-ef {
                loss-priority low code-point 010;
                loss-priority high code-point 011;
            }
            forwarding-class fc-af1 {
                loss-priority low code-point 100;
                loss-priority high code-point 101;
            }
            forwarding-class fc-nc {
                loss-priority low code-point 110;
                loss-priority high code-point 111;
            }
        }
```

```
    }
    scheduler-maps {
        core {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-20-high-rate-limit;
            forwarding-class fc-af1 scheduler sc-20-dp;
            forwarding-class fc-nc scheduler sc-5;
        }
    }
    schedulers {
        sc-remainder {
            transmit-rate {
                remainder;
            }
        }
        sc-20-dp {
            transmit-rate percent 20;
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-20-high-rate-limit {
                transmit-rate {
                percent 20;
                rate-limit;
            }
            priority high;
        }
        sc-5 {
            transmit-rate percent 5;
        }
    }
}
```

The configuration defines eight forwarding classes and establishes a mapping between these classes and the hardware queue numbers. The configuration also defines DSCP/traffic class classifiers (**all-be-v4** and **all-be-v6**) that classify all IPv4 and IPv6 traffic to the best-effort (BE) traffic class. Those classifiers are used later in some of the CoS profiles and they also serve as templates for **core-v4** and **core-v6** classifiers. The latter two classifiers are assigned to core interfaces. These classifiers classify certain traffic types as network control, with the only goal being to protect the control traffic from other traffic types. It is assumed that all IPv4 and IPv6 network control traffic is correctly classified as network control (this traffic might potentially be transit; for example, LDP or IBGP sessions over a P router), and all other IPv4 and IPv6 traffic is sent to the customers. This traffic is subject to output classification rules (which might differ between customers). Finally, there is no other IPv4 or IPv6 traffic.

The MPLS EXP classifier is also defined. This classifier is later enabled on core interfaces and maps traffic to four aggregated traffic classes. The MPLS EXP rewrite rule mirrors this classifier.

The template also configures host-bound traffic to use the **fc-nc** forwarding class (and queue 3, as a result). This forwarding class is used to protect control traffic from other traffic types in the egress direction.

The template also defines two drop profiles. One drop profile is aggressive (meaning that random early discards occur at low buffer fill levels). The other drop profile is moderate (with discards occurring at higher buffer fill levels). Aggressive drop profiles are used for out of profile customer traffic.

Finally, a core interface scheduler map is defined (named **core**). This scheduler map refers to four schedulers—one for each aggregate traffic class. Each scheduler establishes a proportion of traffic allocated to aggregate classes (with 5 percent allocated to network control, 20 percent for high priority expedited forwarding traffic class, 20 percent to assured forwarding class, and 55 percent to best-effort class).

NOTE: Expedited forwarding traffic is rate-limited to ensure jitter guarantees.

The primitives defined in the template in Table 34 are used in the next template which defines core interface CoS configuration.

The core interface configuration is comparatively simple; a scheduler, a rewrite rule, DSCP, and EXP classifiers are assigned to an interface.

Table 34: Core Interface Class-of-Service Configuration

Template ID	cos-core-intf	**Device Role**	P and PE router, DCR, AS boundary router, MPLS access PE router
Service	All	**Guidelines**	Apply this template to core-facing interfaces
Dependencies	cos-common		
Parameters			
core_ifd	Physical interface where core-facing interface resides		
core_ifl	Core-facing logical interface (typically equal to 0, as there is only one logical interface per core-facing physical interface)		

```
class-of-service {
    interfaces {
        ${core_ifd} {
            scheduler-map core;
            unit ${core_ifl} {
                classifiers {
                    dscp core-v4;
                    dscp-ipv6 core-v6;
                    exp core;
                }
                rewrite-rules {
                    exp core;
                }
            }
        }
    }
}
```

Global Service Configuration

In this solution, all customer interfaces refer to an apply group applicable to that service. These apply groups can be used to enable some functionality for all interfaces at once.

The template in Table 35 is applicable to all router roles.

Table 35: Global Service Configuration

Template ID	svc-pe-global	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	

```
groups {
    l3vpn-ifl-settings;
    dia-ifl-settings;
}
```

In this template, both groups are empty. However, the next template (applicable to AS boundarfy routers only) modifies the **dia-ifl-settings** group to enable input sampling.

In this solution, IPFIX export at the AS boundary router is supported. IPFIX is used to report information about flows traversing the router. The router randomly samples packets, identifies flows, and periodically exports flow information to the flow collector software.

There are multiple ways to enable sampling and IPFIX collection on Juniper Networks platforms. One method is to enable sampling only at peering interfaces in both the input and output directions. This is a valid approach; however, it might result in the duplication of statistics for flows between two peers. To export traffic only once, you can use the **sample-once** statement (enabled under **forwarding-options** sampling). This statement effectively disables output sampling for packets that were subject to input sampling. Another approach is to enable sampling in the forwarding table filter (input or output). This method requires minimal configuration without having to configure anything at the

interface level. However, packets discarded at earlier steps (because of using input filter discard) are not reported in the sampling process. Because of this lack of reporting, this approach is not implemented in this solution.

Yet another option is to enable sampling on input interfaces only (including core interfaces). This method provides uniform hardware resource utilization (flow tables) across ingress and egress PFEs and provides better scale as a result (instead of collecting all input and output flows at the PFEs with customer attachment circuits, the flows towards the customer are now collected at the PFEs hosting core interfaces). For these reasons, this solution uses this option for sampling. This method does require sampling be enabled on all core and direct Internet access customer interfaces. However, the use of configuration groups enables you to configure sampling on all interfaces at once.

The template in Table 36 enables input sampling at the core interfaces and DIA customer attachment circuits. This template also enables global sampling parameters, such as the destination address of the flow collector, flow timeouts, and the size of the IPv4 and IPv6 flow tables.

The current generation of the MPC is capable of storing 3.84 million flows in the flow memory. That memory is shared between IPv4 and IPv6 flows. The split between memory blocks consumed by IPv4 and IPv6 flows is static and set at the configuration time in units sized at 256,000 flows each. In this solution, 11 out of 15 units (73 percent) are allocated to IPv4 and 4 out of 15 units are allocated to IPv6 (27 percent). This selection is based on the assumption that the number of flows correlates with the traffic volume, and the split between IPv4 and IPv6 traffic volume is approximately 73 percent to 27 percent.

Active flow timeout is set to 180 seconds to enable periodic updates on the long-lived flows. Inactive timeout is set to 15 seconds. This setting enables the aging of flows every 15 seconds and maintains low utilization of the flow table. This setting should be fine-tuned in your actual deployment by monitoring flow table utilization. If flow table utilization reaches its maximum, the flow table active timeout should be reduced. However, shorter timeouts result in a higher export rate. These export rates might exceed flow collector capacity and the Packet Forwarding Engine export capacity. Because of this limitation, timeout values should be selected with care.

The sampling rate is set to 1:1000. This value is appropriate for most statistics collection or analytics applications.

We recommend enabling the sampling configuration on each MPC in the system.

AS Boundary Router Configuration for Direct Internet Access

Table 36: AS Boundary Router Configuration for Direct Internet Access

Template ID	dia-asbr	**Device Role**	AS boundary router
Service	DIA	**Guidelines**	
Constants			
Number of flow table memory units allocated to IPv4 is 11 (11 * 256K = 2816K entries).			
Number of flow table memory units allocated to IPv6 is 4 (4 * 256K = 1024K entries).			
Maximum flow export rate is 60 (60,000 flows per second). This limit is determined by the flow collector performance.			
Active flow timeout is 180 seconds. For an active flow, updates are sent every 180 seconds.			
Inactive flow timeout is 15 seconds. Flows become inactive if no data is seen in 15 seconds.			
Template refresh rate is 180 seconds.			
Option refresh rate is 180 seconds.			
Export rate is 1:1000 (enough for analytics).			
Parameters			
fpc	An array of FPCs where sampling is enabled. All core FPCs and FPCs with DIA customer or peer interfaces should be included.		
flow_server	Flow server IPv4 address		
flow_server_port	Flow server UDP port number		

```
groups {
    /* Note, dia-ifl-settings and core-intf-settings group configuration is expanded in ASBR
       case. Sampling is done on input.*/
    dia-ifl-settings {
        interfaces {
            <*> {
                unit <*> {
                    family inet {
                        sampling {
                            input;
```

```
                        }
                    }
                    family inet6 {
                        sampling {
                            input;
                        }
                    }
                }
            }
        }
    }
    core-intf-settings {
        interfaces {
            <*> {
                unit <*> {
                    family inet {
                        sampling {
                            input;
                        }
                    }
                    family inet6 {
                        sampling {
                            input;
                        }
                    }
                }
            }
        }
    }
}

chassis {
    /* In this case, sampling instance configuration is identical on all MPCs
       In production deployments, an operator may choose to have different instances with
       different flow server destinations in order to load balance the data between them */
    fpc ${fpc[i]} {
        sampling-instance sample-instance-default;
        inline-services {
            flow-table-size {
                ipv4-flow-table-size 11;
                ipv6-flow-table-size 4;
            }
        }
    }
}
forwarding-options {
    /* Configure loose mode discard (enables source remote triggered black hole functionality) */
    rpf-loose-mode-discard {
        family {
            inet;
            inet6;
        }
    }
    sampling {
        instance {
            sample-instance-default {
                input {
                    rate 1000;
                }
                family inet {
                    output {
                        flow-server ${flow_server} {
                            port ${flow_server_port};
                           version-ipfix {
```

```
                                template {
                                    v4;
                                }
                            }
                        }
                        inline-jflow {
                            source-address ${loopback_address};
                            /* 60 means 60K per second */
                            flow-export-rate 60;
                        }
                    }
                }
                family inet6 {
                    output {
                        flow-server ${flow_server} {
                            port ${flow_server_port};
                            version-ipfix {
                                template {
                                    v6;
                                }
                            }
                        }
                        inline-jflow {
                            source-address ${loopback_address};
                            /* 60 means 60K per second */
                            flow-export-rate 60;
                        }
                    }
                }
            }
        }
    }
}

services {
    flow-monitoring {
        version-ipfix {
            template v4 {
                flow-active-timeout 180;
                flow-inactive-timeout 15;
                template-refresh-rate {
                    seconds 180;
                }
                option-refresh-rate {
                    seconds 180;
                }
                ipv4-template;
            }
            template v6 {
                flow-active-timeout 180;
                flow-inactive-timeout 15;
                template-refresh-rate {
                    seconds 180;
                }
                option-refresh-rate {
                    seconds 180;
                }
                ipv6-template;
            }
        }
    }
}
```

Apart from the sampling configuration, the template in Table 36 also enables source remote triggered black-hole support by configuring **rpf-loose-mode-discard**. For each interface with RPF loose check enabled, the router consults the forwarding table using the source address of the packet. If there is an entry in the table pointing to the discard next hop, that packet is dropped. This configuration enables the dropping of packets from certain sources at peering locations.

Firewall/NAT Device Configuration

Geo-redundancy implementation is where firewall/NAT devices forming a redundant group elect a primary device. The primary device becomes active by propagating customer routes. In contrast, those routes are not propagated through other (backup) firewall/NAT devices. The advertisement is conditional.

The election process is based on the exchange of a special "beacon" route between firewall/NAT devices. A special infrastructure VPN is configured for that purpose on the DCR devices. Firewall/NAT devices establish a BGP session to the DCR in a dedicated VLAN used for control purposes.

VPN and attachment circuit configuration on the DCR side uses Layer 3 VPN service templates such as **edge-eth-phys-intf-tagged**, **l3vpn-instance**, and **l3vpn-bgp**.

The template in Table 37 provides the configuration for the firewall/NAT device. A special virtual router (**sys-vr-redundancy**) is used to exchange beacon routes.

Table 37: Firewall Configuration – Geographic Redundancy

Template ID	fwnat-geo-redundancy	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	
Parameters			
as	This router's autonomous system number		
pe_ifd	Physical interface towards PE router with a hub VRF		
pe_unit	Logical interface towards PE router with a hub VRF		
beacon_route	Beacon route, used to track redundancy state		
my_preference	This router preference; 171 for primary firewall/NAT device and 172 for backup firewall/NAT device		
my_community	Community that identifies the route from a primary or a secondary system; set to ${as}:20001 if primary and ${as}:20002 if secondary		
pe_ipv4	PE IPv4 address		
peer_as	Customer autonomous system		

```
policy-options {
    policy-statement pol-export-beacon {
        from {
            protocol static;
            route-filter ${beacon_route} exact;
        }
        then accept;
    }
    policy-statement pol-fwnat-conditional-export {
        term v4-export {
            from {
                protocol bgp;
                family inet;
                condition fwnat-active-device;
            }
            then accept;
        }
        term v6-export {
           from {
                protocol bgp;
                family inet6;
                condition fwnat-active-device;
            }
            then accept;
        }
        term reject-the-rest {
            then reject;
   }
```

```
    }
    policy-statement pol-set-preference-based-on-community {
        term match-comm-primary {
            from community comm-fwnat-primary;
            then {
                preference 171;
            }
        }
        term match-comm-secondary {
            from community comm-fwnat-secondary;
            then {
                preference 172;
            }
        }
    }
    policy-statement pol-use-static-beacon-only {
        term accept-static-beacon {
            from {
                protocol static;
                route-filter ${beacon_route} exact;
            }
            then accept;
        }
        term reject-the-rest {
            then reject;
        }
    }
    community comm-fwnat-primary members ${as}:20001;
    community comm-fwnat-secondary members ${as}:20002;
    condition fwnat-active-device {
        if-route-exists {
            0.0.0.0/0;
            table sys-vr-redundancy.inet.0;
        }
    }
}
routing-instances {
    sys-vr-redundancy {
        instance-type virtual-router;
        interface ${pe_ifd}.${pe_ifl};
        routing-options {
            static {
                route ${beacon_route} {
                    discard;
                    preference ${my_preference};
                    community ${my_community};
                }
            }
            aggregate {
                route 0.0.0.0/0 policy pol-use-static-beacon-only;
            }
        }
        protocols {
            bgp {
                group v4-redundancy-tracking {
                    metric-out ${my_preference};
                    import pol-set-preference-based-on-community;
                    export pol-export-beacon;
                    peer-as ${peer_as};
                    neighbor ${pe_ipv4};
                }
            }
        }
    }
}
```

Each firewall/NAT device has a beacon static route configured. The preference for this beacon route is set to 171 on a primary device and to 172 on a secondary device. Both values are higher than default preference for BGP routes (170), so if a BGP route is received, the static route is always less preferred.

Static routes are also marked with communities. Community **comm-fwnat-primary** identifies routes from a primary system, while community **comm-fwnat-secondary** identifies routes from a secondary system. These communities are used to properly assign route preference once imported from the DCR.

On export to the DCR, the route multiple exit discriminator (MED) attribute is set to 171 by the primary device and to 172 by the secondary device. This facilitates best route selection on the DCR side (routes from a primary device are more preferred).

Presence of an active static beacon route in the **sys-vr-redundancy** virtual router indicates that this firewall/NAT device is active. This indication can be used as a condition to propagate routes through the firewall/NAT device or stop that propagation.

Junos OS supports checking whether or not a route is present in a routing table. However, Junos OS does not support checking if a route of a certain type is present (in this case, a static route). To overcome this limitation, an aggregate 0/0 route is used. This aggregate route is only created if the contributing route is static (that is, the device is primary). The presence of this 0/0 route indicates that the static beacon route is active, and if a BGP beacon route is received, it is ignored.

Policy condition **fwnat-active-device** verifies whether or not this aggregate is present in the virtual router. It is used in the policy **pol-fwnat-conditional-export**, which is a policy used in the firewall/NAT device service configuration templates.

Service-Specific Design Configuration Templates

This section contains a collection of service-specific design configuration templates. These templates are used for the configuration of service-specific design components that are then applied to the routing devices used in the Juniper Networks business edge solution. The following subsections cover the various service-specific solution configurations:

- Service-Independent Configuration
- Class-of-Service
- Direct Internet Access Service-Specific Configuration
- Layer 3 VPN Service-Specific Configuration
- Multicast Layer 3 VPN Service-Specific Configuration
- Firewall and Network Address Translation Service-Specific Configuration

Service-Independent Configuration

Ethernet or Aggregated Ethernet

When configuring single-tagged or dual-tagged Ethernet and aggregated Ethernet attachment circuits, you must provision VLAN identifiers. The template in Table 38 defines how to configure single-tagged Ethernet attachment circuits.

Table 38: Link-Layer Configuration—Single-Tagged Ethernet Attachment Circuit

Template ID	edge-eth-link-tagged	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	Single-tagged circuits
Parameters			
cust_ifd	Physical interface name		
cust_ifl	Logical interface name		
vlan_id	VLAN identifier		

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            vlan-id ${vlan_id};
        }
    }
}
```

The template in Table 39 defines how to configure dual-tagged Ethernet attachment circuits.

Table 39: Link-Layer Configuration—Dual-Tagged Ethernet Attachment Circuit

Template ID	edge-eth-link-double-tagged	**Device Role**	PE router, DCR, AS boundary router
Service	All	**Guidelines**	Dual-tagged circuits
Parameters			
cust_ifd	Physical interface name		
cust_ifl	Logical interface name		
outer_vlan_id	Outer VLAN identifier		
inner_vlan_id	Inner VLAN identifier		

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            vlan-tags outer ${outer_vlan_id} inner ${inner_id};
        }
    }
}
```

MPLS Access Attachment Circuit

In this solution, the following two different MPLS PE router attachment circuit termination options are considered:

- Access PE router port traffic cross connected to a pseudowire interface that is terminated on the PE device
- Access PE router VLAN traffic from a specific port cross-connected to a pseudowire interface that is terminated on the PE device

In both cases, the pseudowire is signaled as type 4 without a VLAN ID. In the second case, the VLAN is stripped on the access PE device. This is the only difference between the two options and it is reflected in the configurations of the access PE device.

The template in Table 40 is designed to support the first (port) option.

Table 40: MPLS Attachment Circuit Configuration

Template ID	lt-termination-access-pe-port	**Device Role**	Access PE router
Service	All	**Guidelines**	Use in case of a port-based access
Constants			
BFD minimum interval for BFD sessions running on the pseudowires is set to 1,500.			
BFD multiplier for BFD sessions running on the pseudowires is set to 3.			
Pseudowire switchover delay is set to 0, causing the switchover to backup PE router to occur immediately.			
Pseudowire revert time is set 180, so that the switchover to primary PE router occurs 180 seconds after the primary pseudowire is set up.			
Parameters			
cust_ifd	Physical interface where customer interface resides		
primary_pe	IPv4 address of the primary PE router		
backup_pe	IPv4 address of the primary PE router		
vc_id	Virtual circuit identifier (note that the same identifier is used for both primary and backup PE devices); this is made under the assumption that a symmetric redundancy scheme is in place (same pair of routers is used to terminate all circuits from the MPLS access PE router)		

```
groups {
    l2ckt-access-pe-settings {
        protocols {
            l2circuit {
                neighbor <*> {
                    interface <*> {
                        /* Note, MTU is configured explicitly */
                        mtu 9086;
                        switchover-delay 0;
                        revert-time 180;
                        oam {
                            bfd-liveness-detection {
                                minimum-interval 1500;
                                multiplier 3;
                            }
                        }
                    }
                }
            }
        }
    }
}

interfaces {
    ${cust_ifd} {
        flexible-vlan-tagging;
        mtu 9100;
        encapsulation ethernet-ccc;
        unit 0;
    }
}
protocols {
    ldp {
        interface lo0.0;
    }
    l2circuit {
        neighbor ${pe_primary} {
            interface ${cust_ifd}.0 {
                apply-groups l2ckt-access-pe-settings;
                virtual-circuit-id ${vc_id};
                backup-neighbor ${pe_backup} {
                    virtual-circuit-id ${vc_id};
                }
            }
        }
    }
}
```

Note that pseudowire redundancy is configured in this template. Each pseudowire terminates at the primary PE device and, in case of a PE device failure, the pseudowire immediately switches over to the backup PE device. If the primary PE device becomes available, the pseudowire switches over to the primary PE device (revertive mode) after a configured delay of 180 seconds. This delay is configured to ensure that the primary PE device has enough time to learn routes from the rest of the network and program them into the forwarding tables.

On each pseudowire, the BFD session is enabled with a 1,5 second minimum interval. The template also enables LDP on the lo0.0 interface in order to support targeted LDP sessions.

The template in Table 41 is designed to support VLAN transport. This template is very similar to that found in Table 40. The only difference between the two is in the VLAN manipulation configuration. The access PE device strips off the VLAN tag on ingress and adds it on egress.

Table 41: MPLS Attachment Circuit Termination—VLAN Termination

Template ID	lt-termination-access-pe-vlan	**Device Role**	Access PE router
Service	All	**Guidelines**	Use in case of VLAN-based access
Constants			
BFD minimum interval for BFD sessions running on the pseudowires is set to 1,500.			
BFD multiplier for BFD sessions running on the pseudowires is set to 3.			
Pseudowire switchover delay is set to 0, causing the switchover to backup PE router to occur immediately.			
Pseudowire revert time is set 180, so that the switchover to primary PE router occurs 180 seconds after the primary pseudowire is set up.			
Parameters			
cust_ifd	Physical interface where customer interface resides		
cust_unit	Customer logical interface number		
vlan_id	Customer VLAN ID		
primary_pe	IPv4 address of the primary PE router		
backup_pe	IPv4 address of the primary PE router		
vc_id	Virtual circuit identifier (note that the same identifier is used for both primary and backup PEs); this is made under the assumption that a symmetric redundancy scheme is in place (same pair of routers is used to terminate all circuits from the MPLS access PE router)		

```
groups {
    access-port-settings {
        interfaces {
            <*> {
                unit <*> {
                    encapsulation vlan-ccc;
                    /* Strip VLAN tags on ingress / add on egress */
                    input-vlan-map pop;
                    output-vlan-map push;
                }
            }
        }
    }
    l2ckt-access-pe-settings {
        protocols {
            l2circuit {
                neighbor <*> {
                    interface <*> {
                        /* Note, MTU is configured explicitly */
                        mtu 9086;
                        switchover-delay 0;
                        revert-time 180;
                        oam {
                            bfd-liveness-detection {
                                minimum-interval 1500;
                                multiplier 3;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
interfaces {
    ${cust_ifd} {
        flexible-vlan-tagging;
        mtu 9104;
        encapsulation flexible-ethernet-services;
        unit ${cust_unit} {
            apply-groups access-port-settings;
            vlan-id ${vlan_id};
        }
    }
}
protocols {
    ldp {
        interface lo0.0;
    }
    l2circuit {
        neighbor ${pe_primary} {
            interface ${cust_ifd}.${cust_unit} {
                apply-groups l2ckt-access-pe-settings;
                virtual-circuit-id ${vc_id};
                backup-neighbor ${pe_backup} {
                    virtual-circuit-id ${vc_id};
                }
            }
        }
    }
}
```

The template in Table 42 supports pseudowire termination on the MPLS access PE device.

The template enables a logical tunnel interface device (under chassis hierarchy). The logical tunnel interface has a **per-unit-mac-disable** option to facilitate better logical tunnel interface scale (by default each logical tunnel interface consumes MAC addresses from a global chassis pool, limiting their number).

As in the template in Table 41, LDP is enabled on lo0.0.

Table 42: Logical Tunnel Termination on PE Router

Template ID	lt-termination-pe	**Device Role**	PE Router
Service	All	**Guidelines**	None
Constants			
BFD minimum interval for BFD sessions running on the pseudowires is set to 1,500.			
BFD multiplier for BFD sessions running on the pseudowires is set to 3.			
Parameters			
lt_fpc	Line-card number where logical tunnel interface resides		
lt_pic	Interface card where logical tunnel interface resides		
lt_port	Port where logical tunnel interface resides		
cust_unit	Customer logical interface number		
transport_unit	Corresponding transport logical interface number (peer unit)		
vc_id	Virtual circuit identifier		
access_pe	Access PE router IP address		

```
groups {
  l2ckt-settings {
       protocols {
           l2circuit {
               neighbor <*> {
                   interface <*> {
                       oam {
                           bfd-liveness-detection {
                              minimum-interval 1500;
```

```
                                multiplier 3;
                            }
                        }
                    }
                }
            }
        }
    }
}

chassis {
    fpc ${lt_fpc} {
        pic ${lt_pic} {
            /* Note, tunnel bandwidth is not limited */
            tunnel-services;
        }
    }
}

interfaces {
    lt-${lt_fpc}/${lt_pic}/${lt_port} {
        hierarchical-scheduler;
        mtu 9100;
        logical-tunnel-options {
            /* Disabling per unit mac address allocation facilitates lt- scaling */
            per-unit-mac-disable;
        }
        unit ${cust_unit} {
            encapsulation ethernet;
            peer-unit ${transport_unit};
        }
        unit ${transport_unit} {
            encapsulation ethernet-ccc;
            peer-unit ${cust_unit};
        }
    }
}
protocols {
    l2circuit {
        apply-groups l2ckt-settings;
        neighbor ${access_pe} {
            interface lt-${lt_fpc}/${lt_pic}/${lt_port}.${transport_unit} {
                virtual-circuit-id ${vc_id};
            }
        }
    }
    ldp {
        interface lo0.0;
    }
}
```

The Layer 2 circuit from the MPLS access PE device is terminated on one of the logical tunnel units (referred to as **transport_unit**), while actual service is provisioned on the **cust_unit**.

IP Layer Configuration

This template in Table 43 defines a DIA service-specific attachment circuit configuration. Router IP addresses are configured at the logical interface level and RPF check is enabled for both Layer 3 VPN and DIA services.

NOTE: The **service_name** parameter is used to identify an apply-group that is applicable for the service. Depending on the device role, the group might call for some additional interface configuration (for example, sampling at the AS boundary router when configuring DIA service).

Table 43: IP Layer Configuration—Attachment Circuit

Template ID	ip-attachment-circuit	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Use for customer attachment circuit (except DIA peer or upstream) and except Layer 3 VPN spoke site in hub-and-spoke Layer 3 VPN topology
Dependencies	svc-pe-global		
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
router_ipv4	Router IPv4 address		
router_ipv6	Router IPv4 address		
service_name	Service name parameter set to “l3vpn” or “dia”		

```
interfaces {
    ${cust_ifd} {
        /* Customer IFL configuration */
        unit ${cust_unit} {
            apply-groups ${service_name}-ifl-settings;
            family inet {
                rpf-check;
                address ${router_ipv4};
            }
            family inet6 {
                rpf-check;
                address ${router_ipv6};
            }
        }
    }
}
```

Class-of-Service

Table 44 shows the typical CoS profile application and parameter support for those profiles.

Table 44: CoS Profile Parameter Usage

Parameter	1	2	3	4	5	6	7	8
Typical application	DIA	VPN	DIA	VPN	VPN	VPN	VPN	VPN
N traffic classes (end user)	1	1	1	1	8	8	8	8
Actual N of classes	1	1	2	2	8	8	8	8
BA classifier	Y	Y	Y	Y	Y	Y	Y	Y
Ingress shaping/queuing at the interface set								Y
Ingress shaping/queuing						Y		Y
Ingress singe-rate two color policer per class	Y	Y	Y	Y	Y	Y	Y	Y
Hierarchical ingress policer					Y		Y	
DSCP bleaching	Y		Y					
Egress classification filter	Y	Y	Y	Y	Y	Y	Y	Y
Egress policer	Y	Y						
Egress queuing at the logical interface (IFL) level			Y	Y	Y	Y	Y	Y
Egress queuing at the IFL-set level							Y	Y

The first CoS profile is typically enabled for Direct Internet Access service attachment circuits. The profile enables DSCP/TC bleaching, and features ingress and egress policing. All ingress customer traffic is classified to the best-effort aggregate class. On egress, all traffic is treated as best effort.

The CoS template in Table 45 defines logical interface-specific policers and filters. This definition can be shared by multiple attachment circuits. A separate filter and a policer instance will be created for each attachment circuit; all

policers and filters are interface-specific. The only factor to take into consideration is that the policer attached to the aggregated Ethernet should have a shared bandwidth keyword enabled (which enables policer rate programming in proportion to the number of links terminated on the PFE), while policers and filters attached to regular interfaces must not have this setting (otherwise the commit will fail). A "-shared" suffix is used in the policer name (and a filter name that refers to that policer) in order to identify filters or policers that should be attached to the aggregated interfaces.

Table 45: Class-of-Service—Profile 1 Configuration

Template ID	cos-profile-1	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Typically used for DIA (because of DSCP bleaching)
Constants			
Policer burst size is set to 35 KB			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
rate_ingress	Ingress policing rate (for both address families)		
rate_egress	Egress policing rate (for both address families)		
suffix	Policer name suffix, set to "-shared" in case of the aggregated Ethernet customer IFD interface, otherwise left blank		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to "shared-bandwidth-policer" in case of aggregated Ethernet customer IFD interface, otherwise left blank		

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            family inet {
                filter {
                    input-list cos-${rate_ingress}${suffix}-bleach-ingress-v4;
                    output-list cos-${rate_egress}${suffix}-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list  cos-${rate_ingress}${suffix}-bleach-ingress-v6;
                    output-list cos-${rate_egress}${suffix}-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter cos-${rate_ingress}${suffix}-bleach-ingress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                    /* DSCP bleaching */
                    dscp be;
                }
            }
        }
        filter cos-${rate_egress}${suffix}-egress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_egress}${suffix};
                    /* Specify forwarding class on egress */
                    forwarding-class fc-be;
```

```
                }
            }
        }
    }
    family inet6 {
        filter cos-${rate_ingress}${suffix}-bleach-ingress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                    /* Traffic Class bleaching */
                    traffic-class be;
                }
            }
        }
        filter cos-${rate_egress}${suffix}-egress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_egress}${suffix};
                    forwarding-class fc-be;
                }
            }
        }
    }
    policer pol-${rate_ingress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_ingress};
            burst-size-limit 35k;
        }
        then discard;
    }
    policer pol-${rate_egress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_egress};
            burst-size-limit 35k;
        }
        then discard;
    }
}

class-of-service {
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                classifiers {
                    dscp all-be-v4;
                    dscp-ipv6 all-be-v6;
                }
            }
        }
    }
}
```

The CoS template in Table 46 defines the second class-of-service profile. This profile is similar to profile 1. The only difference is that it provides DSCP transparency.

Table 46: Class-of-Service—Profile 2 Configuration

Template ID	cos-profile-2	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Typically used for Layer 3 VPN (provides DSCP transparency)
Constants			
Policer burst size is set to 35 KB			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
rate_ingress	Ingress policing rate (for both address families)		
rate_egress	Egress policing rate (for both address families)		
suffix	Policer name suffix, set to "-shared" in case of the aggregated Ethernet customer IFD interface, otherwise left blank		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to "shared-bandwidth-policer" in case of aggregated Ethernet customer IFD interface, otherwise left blank		

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            family inet {
                filter {
                    /* Use uniform input-list syntax across all profiles, even it is not
                       required */
                    input-list cos-${rate_ingress}${suffix}-ingress-v4;
                    output-list  cos-${rate_egress}${suffix}-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list cos-${rate_ingress}${suffix}-ingress-v6;
                    output-list cos-${rate_egress}${suffix}-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter cos-${rate_ingress}${suffix}-ingress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                }
            }
        }
        filter cos-${rate_egress}${suffix}-egress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_egress}${suffix};
                    /* Specify forwarding class on egress */
                    forwarding-class fc-be;
                }
            }
        }
    }
    family inet6 {
        filter cos-${rate_ingress}${suffix}-ingress-v6 {
            interface-specific;
```

```
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                }
            }
        }
        filter cos-${rate_egress}${suffix}-egress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_egress}${suffix};
                    forwarding-class fc-be;
                }
            }
        }
    }
    policer pol-${rate_ingress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_ingress};
            burst-size-limit 35k;
        }
        then discard;
    }
    policer pol-${rate_egress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_egress};
            burst-size-limit 35k;
        }
        then discard;
    }
}

class-of-service {
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                classifiers {
                    dscp all-be-v4;
                    dscp-ipv6 all-be-v6;
                }
            }
        }
    }
}
```

The CoS template in Table 47 defines the third class-of-service profile. This profile features egress shaping, meaning that there is no need to police traffic on egress and that the egress filter is used for classification purposes only. The profile is used to support single traffic classes, it has DSCP bleaching enabled, and it is primarily used for DIA service. However, two queues are allocated for each attachment circuit where a separate queue is used for control traffic (host outbound traffic only).

Unlike the way policers operate, scheduling and shaping processes all Layer 1 and Layer 2 overheads for bandwidth measurements into account by default. To provide uniform bandwidth enforcement on ingress and egress, the egress traffic control profile includes overhead accounting adjustment configuration. The adjustment varies for different attachment circuit types. It is set to 38 bytes for untagged Ethernet attachment circuits, 42 bytes for single-tagged Ethernet attachment circuits, and 46 bytes for dual-tagged attachment circuits.

Table 47: Class-of-Service—Profile 3 Configuration

Template ID	cos-profile-3	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Typically used for DIA (because of DSCP bleaching)
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Best-effort queue bandwidth is set to 95 percent.			
Network control queue bandwidth is set to 5 percent.			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
rate_ingress	Ingress policing rate (for both address families)		
rate_egress	Egress shaping rate (for both address families)		
ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged		
suffix	Policer name suffix, set to "-shared" in case of the aggregated Ethernet customer IFD interface, otherwise left blank		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to "shared-bandwidth-policer" in case of aggregated Ethernet customer IFD interface, otherwise left blank		

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            family inet {
                rpf-check;
                filter {
                    input-list cos-${rate_ingress}${suffix}-bleach-ingress-v4;
                    output-list cos-unlimited-egress-v4;
                }
            }
            family inet6 {
                rpf-check;
                filter {
                    input-list cos-${rate_ingress}${suffix}-bleach-ingress-v6;
                    output-list cos-unlimited-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter cos-${rate_ingress}${suffix}-bleach-ingress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                    /* DSCP bleaching */
                    dscp be;
                }
            }
        }
        filter cos-unlimited-egress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    /* Specify forwarding class on egress */
                    forwarding-class fc-be;
```

```
                }
            }
        }
    }
    family inet6 {
        filter cos-${rate_ingress}${suffix}-bleach-ingress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                    /* Traffic Class bleaching */
                    traffic-class be;
                }
            }
        }
        filter cos-unlimited-egress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    forwarding-class fc-be;
                }
            }
        }
    }
    policer pol-${rate_ingress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_ingress};
            burst-size-limit 35k;
        }
        then discard;
    }
}
class-of-service {
    traffic-control-profiles {
        tcp-${rate_egress}-${ovhd_bytes}-2q-95-5 {
            scheduler-map edge-2q-95-5;
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${rate_egress};
        }
    }
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                output-traffic-control-profile tcp-${rate_egress}-${ovhd_bytes}-2q-95-5;
                classifiers {
                    dscp all-be-v4;
                    dscp-ipv6 all-be-v6;
                }
            }
        }
    }
 scheduler-maps {
        edge-2q-95-5 {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-nc scheduler sc-5-medium-exact;
        }
    }
    schedulers {
        sc-remainder {
               transmit-rate {
```

```
                remainder;
            }
        }
        sc-5-medium-exact {
            transmit-rate {
                percent 5;
                exact;
            }
            priority medium-high;
        }
    }
}
```

Interfaces of the same encapsulation type and bandwidth requirements might use the traffic control profiles defined in the template described in Table 47.

The CoS template in Table 48 defines the fourth class-of-service profile. This CoS profile is similar to the CoS template in Table 47, except that it supports DSCP transparency. For this reason, it is better suited for Layer 3 VPN deployments.

Table 48: Class-of-Service—Profile 4 Configuration

Template ID	cos-profile-4	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Typically used for Layer 3 VPN (provides DSCP transparency)
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Best-effort queue bandwidth is set to 95 percent.			
Network control queue bandwidth is set to 5 percent.			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
rate_ingress	Ingress policing rate (for both address families)		
rate_egress	Egress shaping rate (for both address families)		
ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged		
suffix	Policer name suffix, set to "-shared" in case of the aggregated Ethernet customer IFD interface, otherwise left blank		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to "shared-bandwidth-policer" in case of aggregated Ethernet customer IFD interface, otherwise left blank		

```
interfaces {
    ${cust_ifd {
        unit ${cust_unit} {
            family inet {
                filter {
                    input-list cos-${rate_ingress}${suffix}-ingress-v4;
                    output-list cos-unlimited-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list cos-${rate_ingress}${suffix}-ingress-v6;
                    output-list cos-unlimited-egress-v6;
                }
            }
        }
    }
}
```

```
firewall {
    family inet {
        filter cos-${rate_ingress}${suffix}-ingress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                }
            }
        }
        filter cos-unlimited-egress-v4 {
            interface-specific;
            term all-traffic {
                then {
                    /* Specify forwarding class on egress */
                    forwarding-class fc-be;
                }
            }
        }
    }
    family inet6 {
        filter cos-${rate_ingress}${suffix}-ingress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    policer pol-${rate_ingress}${suffix};
                }
            }
        }
        filter cos-unlimited-egress-v6 {
            interface-specific;
            term all-traffic {
                then {
                    forwarding-class fc-be;
                }
            }
        }
    }
    policer pol-${rate_ingress}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${rate_ingress};
            burst-size-limit 35k;
        }
        then discard;
    }
}
class-of-service {
    traffic-control-profiles {
        tcp-${rate_egress}-2q-95-5 {
            scheduler-map edge-2q-95-5;
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${rate_egress};
        }
    }
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                output-traffic-control-profile tcp-${rate_egress}-2q-95-5;
                classifiers {
                    dscp all-be-v4;
                    dscp-ipv6 all-be-v6;
```

```
                }
            }
        }
    }
    scheduler-maps {
        edge-2q-95-5 {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-nc scheduler sc-5-medium-exact;
        }
    }
    schedulers {
        sc-remainder {
            transmit-rate {
                remainder;
            }
        }
        sc-5-medium-exact {
            transmit-rate {
                percent 5;
                exact;
            }
            priority medium-high;
        }
    }
}
```

The CoS template in Table 49 defines the fifth class-of-service profile. This profile supports eight traffic classes with shaping on egress and policing in ingress.

In the ingress direction, the traffic class is policed to a certain committed ingress rate. In addition, a total rate limit is imposed on each attachment circuit. For all traffic classes except expedited forwarding (EF) and network control (NC), traffic that exceeds the committed rate limit is accepted but it is marked with a higher loss priority. Policers for assured forwarding (AF) traffic classes operate in color-aware mode. All traffic within a committed rate is treated as premium and not discarded. Hierarchical policers are used on ingress to enforce total ingress bandwidth limit. These policers are configured so that the premium traffic is accepted but traffic exceeding total allowed ingress bandwidth is discarded. It is worth noting that the burst size for the premium rate-limit burst size is equal to a sum of burst sizes of individual traffic class policers.

In the egress direction, attachment circuit traffic is shaped to a certain limit. Each traffic class is mapped to an egress queue and each queue receives some guaranteed bandwidth. All AF queues plus the BE queue are configured with the same scheduling priority. However, the AF queue has an explicit excess rate configuration, unlike the BE queue. This means that the AF queue receives priority for excess bandwidth over the BE queue. Medium scheduling priority is set and exact traffic rate enforcement is enabled on the network control traffic class. Finally, the EF traffic class receives highest priority. To maintain jitter guarantees, this queue is rate-limited.

Table 49: Class-of-Service—Profile 5 Configuration

Template ID	cos-profile-5	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Moderate drop profile (dp-moderate) starts dropping traffic at 75 percent fill level (linear function).			
Aggressive drop profile (dp-aggressive) starts dropping traffic at 50 percent fill level (linear function).			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
total_ ingress_rate	Ingress policing rate (for both address families)		
total_egress_rate	Egress shaping rate (for both address families)		

ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged
suffix	Policer name suffix, set to "-shared" in case of the aggregated Ethernet customer IFD interface, otherwise left blank
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to "shared-bandwidth-policer" in case of aggregated Ethernet customer IFD interface, otherwise left blank
af1x_ingress_rate	AF1 traffic class committed ingress rate
af1x_egress_rate	AF1 traffic class committed egress rate
af1x_egress_pct	AF1 traffic class committed egress rate in percent to the total_egress_rate
af2x_ingress_rate	AF2 traffic class committed ingress rate
af2x_egress_rate	AF2 traffic class committed egress rate
af2x_egress_pct	AF2 traffic class committed egress rate in percent to the total_egress_rate
af3x_ingress_rate	AF3 traffic class committed ingress rate
af3x_egress_rate	AF3 traffic class committed egress rate
af3x_egress_pct	AF3 traffic class committed egress rate in percent to the total_egress_rate
af4x_ingress_rate	AF4 traffic class committed ingress rate
af4x_egress_rate	AF4 traffic class committed egress rate
af4x_egress_pct	AF4 traffic class committed egress rate in percent to the total_egress_rate
af5_ingress_rate	AF5 traffic class committed ingress rate (note, this class is not defined in the RFC)
af5_egress_rate	AF5 traffic class committed egress rate (note, this class is not defined in the RFC)
af5_egress_rate	AF5 traffic class committed egress rate in percent to the total_egress_rate
nc_ingress_rate	NC traffic class committed ingress rate
nc_egress_rate	NC traffic class committed egress rate
nc_egress_pct	NC traffic class committed egress rate in percent to the total_egress_rate
ef_ingress_rate	EF traffic class committed ingress rate
ef_egress_rate	EF traffic class committed egress rate
ef_egress_rate	EF traffic class committed egress rate in percent to the total_egress_rate
premium_rate	Sum of af1x_ingress_rate, af2x_ingress_rate, af3x_ingress_rate, af4x_ingress_rate, af5_ingress_rate, nc_ingress_rate, and ef_ingress_rate; premium_rate should be less than total_ingress_rate (some bandwidth is allocated to the best-effort class).
bw_profile_id	A number that identifies this particular combination of traffic rates
mapping_profile_id	A number that identifies particular mapping of DSCP fields to forwarding classes

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            family inet {
                filter {
                    input-list cos-${total_ingress_rate}${suffix}-8-classes-mix-
                    ${bw_profile_id}-v4;
                    output-list cos-unlimited-8-classes-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list cos-${total_ingress_rate}${suffix}-8-classes-mix-
                    ${bw_profile_id}-v6;
                    output-list cos-unlimited-8-classes-egress-v6;
                }
            }
        }
    }
}
```

```
firewall {
    family inet {
        filter cos-${total_ingress_rate}${suffix}-8-classes-mix-${bw_profile_id}-v4 {
            interface-specific;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1;
                    }
                    next term;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af2_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                   three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
```

```
                        two-rate trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-5 {
                from {
                    forwarding-class fc-af-extra;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term expedited-forwarding {
                from {
                    forwarding-class fc-ef;
                }
                then {
                    /* police traffic unconditionally*/
                    policer pol-${ef_ingress_rate}${suffix}-ef;
                    next term;
                }
            }
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${nc_ingress_rate}${suffix}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term treat-in-contract-as-premium {
                from {
                    /* All traffic within the spec is declared premium */
                    forwarding-class-except fc-be;
                    loss-priority low;
                }
                then {
                    /* Note the usage of the force-premium knob */
                    force-premium;
                    next term;
                }
            }
            term h-police-all {
                then {
                    hierarchical-policer hpol-${premium_rate}-${total_ingress_rate}${suffix};
                    accept;
                }
            }
        }
```

```
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen for
           this solution. */
        filter cos-unlimited-8-classes-egress-v4 {
            interface-specific;
            term expedited-forwarding {
                from {
                    dscp ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
                }
            }
            /* set loss priority first */
            term set-loss-priority {
                from {
                    dscp [af12 af13 af32 af33 af42 af43];
                }
                then {
                    loss-priority medium-low;
                    next term;
                }
            }
            /* set forwarding classes */
            term fc-af1  {
                from {
                    dscp [ af11 af12 af13 ];
                }
                then {
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-1 {
                from {
                    dscp [ af21 af22 af23 ];
                }
                then {
                    forwarding-class fc-af2;
                    accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    dscp [ af31 af32 af33 ];
                }
                then {
                    forwarding-class fc-af3;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    dscp [ af41 af42 af43 ];
                }
                then {
                    forwarding-class fc-af4;
                    accept;
                }
            }
```

```
            term assured-forwarding-4 {
                from {
                    dscp cs5;
                }
                then {
                    forwarding-class fc-af-extra;
                    accept;
                }
            }
            term network-control {
                from {
                    dscp [ cs6 cs7 ];
                }
                then {
                    forwarding-class fc-nc;
                    accept;
                }
            }
            term accept-the-rest {
                /* the rest of the traffic is accepted, some host outbound network-control traffic
                   may hit this term, therefore forwarding class remains intact*/
                then accept;
            }
        }
    }
    family inet6 {
        filter cos-${total_ingress_rate}${suffix}-8-classes-mix-${bw_profile_id}-v6 {
            interface-specific;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1;
                    }
                    next term;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af2_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
```

```
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-5 {
                from {
                    forwarding-class fc-af-extra;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term expedited-forwarding {
                from {
                    forwarding-class fc-ef;
                }
                then {
                    /* police traffic unconditionally*/
                    policer pol-${ef_ingress_rate}${suffix}-ef;
                    next term;
                }
            }
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
```

```
                /* police traffic unconditionally */
                policer pol-${nc_ingress_rate}${suffix}-nc;
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                next term;
            }
        }
        term treat-in-contract-as-premium {
            from {
                /* All traffic within the spec is declared premium */
                forwarding-class-except fc-be;
                loss-priority low;
            }
            then {
                force-premium;
                next term;
            }
        }
        term h-police-all {
            then {
                hierarchical-policer hpol-${premium_rate}-${total_ingress_rate}${suffix};
                accept;
            }
        }
    }
    /* This output filter classifies traffic on PE egress. Another alternative would be to
       use dscp classifiers within a routing-instance, but this method only applies to VPN
       traffic output multifield classifiers are applicable to DIA service as well unlike
       routing-instance classifiers. Therefore they are chosen as a uniform mechanism. */
    filter cos-unlimited-8-classes-egress-v6 {
        interface-specific;
        term expedited-forwarding {
            from {
                traffic-class ef;
            }
            then {
                forwarding-class fc-ef;
                accept;
            }
        }
        /* set loss priority first */
        term set-loss-priority {
            from {
                traffic-class [ af12 af13 af32 af33 af42 af43 ];
            }
            then {
                loss-priority medium-low;
                next term;
            }
        }
        /* set forwarding classes */
        term fc-af1  {
            from {
                traffic-class [ af11 af12 af13 ];
            }
            then {
                forwarding-class fc-af1;
                accept;
            }
        }
       term assured-forwarding-1 {
            from {
                traffic-class [ af21 af22 af23 ];
```

```
                }
                then {
                    forwarding-class fc-af2;
                    accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    traffic-class [ af31 af32 af33 ];
                }
                then {
                    forwarding-class fc-af3;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    traffic-class [ af31 af32 af33 ];
                }
                then {
                    forwarding-class fc-af4;
                    accept;
                }
            }
            term assured-forwarding-4 {
                from {
                    traffic-class cs5;
                }
                then {
                    forwarding-class fc-af-extra;
                    accept;
                }
            }
            term network-control {
                from {
                    traffic-class [ cs6 cs7 ];
                }
                then {
                    forwarding-class fc-nc;
                    accept;
                }
            }
            term accept-the-rest {
                /* the rest of the traffic is accepted, some host outbound network-control
                   traffic may hit this term, therefore forwarding class remains intact*/
                then accept;
            }
        }
    }
    policer pol-${ef_ingress_rate}${suffix}-ef {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${ef_ingress_rate};
            burst-size-limit 35k;
        }
        then discard;
    }
    policer pol-${nc_ingress_rate}${suffix}-nc {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
```

```
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${nc_ingress_rate};
            burst-size-limit 35k;
        }
        then discard;
    }
    hierarchical-policer hpol-${premium_rate}-${total_ingress_rate}${suffix} {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        aggregate {
            if-exceeding {
                bandwidth-limit ${total_ingress_rate};
                /* set burst size limit to a sum of burst size limits of all policers*/
                burst-size-limit 280k;
            }
            then {
                discard;
            }
        }
        premium {
            if-exceeding {
                bandwidth-limit ${premium_rate};
                /* set burst size limit to a sum of burst size limits of policers
                   used for premium traffic*/
                burst-size-limit 245k;
            }
            then {
                discard;
            }
        }
    }
    three-color-policer trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            committed-information-rate ${af1_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            committed-information-rate ${af2_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
```

```
        three-color-policer trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3 {
            logical-interface-policer;
            /* Enable shared-bandwidth policing for AE interfaces only */
            ${shared_bandwidth};
            action {
                loss-priority high then discard;
            }
            two-rate {
                color-aware;
                committed-information-rate ${af3_ingress_rate};
                committed-burst-size 35k;
                peak-information-rate ${total_ingress_rate};
                peak-burst-size 35k;
            }
        }
        three-color-policer trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4 {
            logical-interface-policer;
            /* Enable shared-bandwidth policing for AE interfaces only */
            ${shared_bandwidth};
            action {
                loss-priority high then discard;
            }
            two-rate {
                color-aware;
                committed-information-rate ${af4_ingress_rate};
                committed-burst-size 35k;
                peak-information-rate ${total_ingress_rate};
                peak-burst-size 35k;
            }
        }
        three-color-policer trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5 {
            logical-interface-policer;
            /* Enable shared-bandwidth policing for AE interfaces only */
            ${shared_bandwidth};
            action {
                loss-priority high then discard;
            }
            two-rate {
                color-aware;
                committed-information-rate ${af5_ingress_rate};
                committed-burst-size 35k;
                peak-information-rate ${total_ingress_rate};
                peak-burst-size 35k;
            }
        }
    }

    class-of-service {
        classifiers {
            dscp edge-8-classes-${mapping_profile_id}-v4 {
                import all-be-v4;
                forwarding-class fc-ef {
                    loss-priority low code-points ef;
                }
                forwarding-class fc-af1  {
                    loss-priority low code-points af11;
                    loss-priority medium-low code-points [ af12 af13 ];
                }
                forwarding-class fc-af2 {
                    loss-priority low code-points af21;
                    loss-priority medium-low code-points [ af22 af23 ];
                }
```

```
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
        dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6 {
            import all-be-v6;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1  {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
    }
    traffic-control-profiles {
        tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id} {
            scheduler-map edge-8q-mix-${bw_profile_id};
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_egress_rate};
        }
    }
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                output-traffic-control-profile
                    tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id};
                classifiers {
                    dscp edge-8-classes-${mapping_profile_id}-v4;
                    dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6;
                }
            }
        }
    }
```

```
    scheduler-maps {
        edge-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-${ef_egress_pct}-high-rate-limit;
            forwarding-class fc-af1  scheduler sc-${af1_egress_pct}-dp-excess;
            forwarding-class fc-af2 scheduler sc-${af2_egress_pct}-dp-excess;
            forwarding-class fc-af3 scheduler sc-${af3_egress_pct}-dp-excess;
            forwarding-class fc-af4 scheduler sc-${af4_egress_pct}-dp-excess;
            forwarding-class fc-af-extra scheduler sc-${af5_egress_pct}-dp-excess;
            forwarding-class fc-nc scheduler sc-${nc_egress_pct}-medium-exact;
        }
    }
    schedulers {
        sc-${nc_conn_i_egress_pct}-medium-exact {
            transmit-rate {
                percent ${nc_conn_i_egress_pct};
                exact;
            }
            priority medium-high;
        }
        sc-${af1_egress_pct}-dp-excess {
            transmit-rate percent ${af1_egress_pct};
            excess-rate percent ${af1_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_egress_pct}-dp-excess {
            transmit-rate percent ${af2_egress_pct};
            excess-rate percent ${af2_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_egress_pct}-dp-excess {
            transmit-rate percent ${af3_egress_pct};
            excess-rate percent ${af3_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_egress_pct}-dp-excess {
            transmit-rate percent ${af4_egress_pct};
            excess-rate percent ${af4_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_egress_pct}-dp-excess {
            transmit-rate percent ${af5_egress_pct};
            excess-rate percent ${af5_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        /* note! this scheduled does not have excess-rate configured, therefore this traffic is
                 preempted by other traffic classes with excess-rate config */
        sc-remainder {
```

```
            transmit-rate {
                remainder;
            }
        }
        sc-${ef_egress_pct}-high-rate-limit {
            transmit-rate {
                percent ${ef_egress_pct};
                rate-limit;
            }
            priority high;
        }
    }
}
```

The CoS template in Table 50 defines the sixth class-of-service profile. This profile is similar to the profile defined in Table 49, except that instead of ingress policing, the profile features ingress queuing and shaping.

NOTE: This profile is not supported for MPLS access attachment circuits.

Ingress and egress shaping and queuing configurations are similar, having only a minor difference in the EF queue configuration. Rate limiting is not supported in the ingress direction. To maintain jitter guarantees, a 1 ms delay buffer is configured instead.

Ingress policers are still enabled, but they are used to classify and mark high loss priority traffic only.

Table 50: Class-of-Service—Profile 6 Configuration

Template ID	cos-profile-6	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	Not supported for MPLS access attachment circuits
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Moderate drop profile (dp-moderate) starts dropping traffic at 75 percent fill level (linear function).			
Aggressive drop profile (dp-aggressive) starts dropping traffic at 50 percent fill level (linear function).			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
total_ ingress_rate	Ingress policing rate (for both address families)		
total_egress_rate	Egress shaping rate (for both address families)		
ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged		
suffix	Policer name suffix, set to “-shared” in case of the aggregated Ethernet customer IFD interface, otherwise left blank		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to “shared-bandwidth-policer” in case of aggregated Ethernet customer IFD interface, otherwise left blank		
af1x_ingress_rate	AF1 traffic class committed ingress rate		
af1x_egress_rate	AF1 traffic class committed egress rate		
af1x_egress_pct	AF1 traffic class committed egress rate in percent to the total_egress_rate		
af2x_ingress_rate	AF2 traffic class committed ingress rate		
af2x_egress_rate	AF2 traffic class committed egress rate		
af2x_egress_pct	AF2 traffic class committed egress rate in percent to the total_egress_rate		
af3x_ingress_rate	AF3 traffic class committed ingress rate		
af3x_egress_rate	AF3 traffic class committed egress rate		
af3x_egress_pct	AF3 traffic class committed egress rate in percent to the total_egress_rate		
af4x_ingress_rate	AF4 traffic class committed ingress rate		

af4x_egress_rate	AF4 traffic class committed egress rate
af4x_egress_pct	AF4 traffic class committed egress rate in percent to the total_egress_rate
af5_ingress_rate	AF5 traffic class committed ingress rate (note, this class is not defined in the RFC)
af5_egress_rate	AF5 traffic class committed egress rate (note, this class is not defined in the RFC)
af5_egress_rate	AF5 traffic class committed egress rate in percent to the total_egress_rate
nc_ingress_rate	NC traffic class committed ingress rate
nc_egress_rate	NC traffic class committed egress rate
nc_egress_pct	NC traffic class committed egress rate in percent to the total_egress_rate
ef_ingress_rate	EF traffic class committed ingress rate
ef_egress_rate	EF traffic class committed egress rate
ef_egress_rate	EF traffic class committed egress rate in percent to the total_egress_rate
premium_rate	Sum of af1x_ingress_rate, af2x_ingress_rate, af3x_ingress_rate, af4x_ingress_rate, af5_ingress_rate, nc_ingress_rate, and ef_ingress_rate; premium_rate should be less than total_ingress_rate (some bandwidth is allocated to the best-effort class).
bw_profile_id	A number that identifies this particular combination of traffic rates
mapping_profile_id	A number that identifies particular mapping of DSCP fields to forwarding classes

```
interfaces {
    ${cust_ifd} {
        unit ${cust_unit} {
            family inet {
                filter {
                    input-list
                      cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v4;
                    output-list cos-unlimited-8-classes-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list
                      cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v6;
                    output-list cos-unlimited-8-classes-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-1-v4 {
            interface-specific;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1;
                    }
                    accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
```

```
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af2_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-5 {
                from {
                    forwarding-class fc-af-extra;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
```

```
            term expedited-forwarding {
                from {
                    forwarding-class fc-ef;
                }
                then {
                    /* police traffic unconditionally*/
                    policer pol-${ef_ingress_rate}${suffix}-ef;
                    accept;
                }
            }
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${nc_ingress_rate}${suffix}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term accept-the-rest {
                then {
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen for
           this solution. */
        filter cos-unlimited-8-classes-egress-v4 {
            interface-specific;
            term expedited-forwarding {
                from {
                    dscp ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
                }
            }
            /* set loss priority first */
            term set-loss-priority {
                from {
                    dscp [af12 af13 af32 af33 af42 af43];
                }
                then {
                    loss-priority medium-low;
                    next term;
                }
            }
            /* set forwarding classes */
            term assured-forwarding-1 {
                from {
                    dscp [ af11 af12 af13 ];
                }
                then {
                    forwarding-class fc-af1;
                    accept;
                }
```

```
            }
            term assured-forwarding-2 {
                from {
                    dscp [ af21 af22 af23 ];
                }
                then {
                    forwarding-class fc-af2;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    dscp [ af31 af32 af33 ];
                }
                then {
                    forwarding-class fc-af3;
                    accept;
                }
            }
            term assured-forwarding-4 {
                from {
                    dscp [ af41 af42 af43 ];
                }
                then {
                    forwarding-class fc-af4;
                    accept;
                }
            }
            term assured-forwarding-5 {
                from {
                    dscp cs5;
                }
                then {
                    forwarding-class fc-af-extra;
                    accept;
                }
            }
            term network-control {
                from {
                    dscp [ cs6 cs7 ];
                }
                then {
                    forwarding-class fc-nc;
                    accept;
                }
            }
            term accept-the-rest {
                /* the rest of the traffic is accepted, some host outbound fc-nc traffic might
                   hit this term, therefore forwarding class remains intact*/
                then accept;
            }
        }
    }
    family inet6 {
        filter cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v6 {
            interface-specific;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                      allowed total rate */
```

```
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1;
                }
                accept;
            }
        }
        term assured-forwarding-2 {
            from {
                forwarding-class fc-af2;
            }
            then {
                /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                   allowed total rate */
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate trtcm-${af2_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-3 {
            from {
                forwarding-class fc-af3;
            }
            then {
                /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                   allowed total rate */
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-4 {
            from {
                forwarding-class fc-af4;
            }
            then {
                /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                   allowed total rate */
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-5 {
            from {
                forwarding-class fc-af-extra;
            }
```

```
                then {
                    /* mark traffic exceeding traffic class as yellow, discard traffic exceeding
                       allowed total rate */
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term expedited-forwarding {
                from {
                    forwarding-class fc-ef;
                }
                then {
                    /* police traffic unconditionally*/
                    policer pol-${ef_ingress_rate}${suffix}-ef;
                    accept;
                }
            }
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${nc_ingress_rate}${suffix}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term accept-the-rest {
                then {
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. Therefore they are chosen as a uniform mechanism. */
        filter cos-unlimited-8-classes-egress-v6 {
            interface-specific;
            term expedited-forwarding {
                from {
                    traffic-class ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
                }
            }
            /* set loss priority first */
            term set-loss-priority {
                from {
                    traffic-class [af12 af13 af32 af33 af42 af43];
                }
                then {
                   loss-priority medium-low;
```

```
                next term;
            }
        }
        /* set forwarding classes */
        term assured-forwarding-1 {
            from {
                traffic-class [ af11 af12 af13 ];
            }
            then {
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-2 {
            from {
                traffic-class [ af21 af22 af23 ];
            }
            then {
                forwarding-class fc-af2;
                accept;
            }
        }
        term assured-forwarding-3 {
            from {
                traffic-class [ af31 af32 af33 ];
            }
            then {
                forwarding-class fc-af3;
                accept;
            }
        }
        term assured-forwarding-4 {
            from {
                traffic-class [ af31 af32 af33 ];
            }
            then {
                forwarding-class fc-af4;
                accept;
            }
        }
        term assured-forwarding-5 {
            from {
                traffic-class cs5;
            }
            then {
                forwarding-class fc-af-extra;
                accept;
            }
        }
        term network-control {
            from {
                traffic-class [ cs6 cs7 ];
            }
            then {
                forwarding-class fc-nc;
                accept;
            }
        }
        term accept-the-rest {
            /* the rest of the traffic is accepted, some host outbound fc-nc traffic
               may hit this term, therefore forwarding class remains intact*/
            then accept;
        }
```

```
        }
    }
    policer pol-${ef_ingress_rate}${suffix}-ef {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${ef_ingress_rate};
            /* Note! burst size is set to 35K to match shaper burst size */
            burst-size-limit 35k;
        }
        then discard;
    }
    policer pol-${nc_ingress_rate}${suffix}-nc {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${nc_ingress_rate};
            /* Note! burst size is set to 35K to match shaper burst size */
            burst-size-limit 35k;
        }
        then discard;
    }
    three-color-policer trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af1 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af1_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af1_ingress_rate}-${total_ingress_rate}${suffix}-ca-af2 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af2_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af3_ingress_rate}-${total_ingress_rate}${suffix}-ca-af3 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
       }
```

```
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af3_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af4_ingress_rate}-${total_ingress_rate}${suffix}-ca-af4 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af4_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af5_ingress_rate}-${total_ingress_rate}${suffix}-ca-af5 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af5_ingress_rate};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
}

class-of-service {
    classifiers {
        dscp edge-8-classes-${mapping_profile_id}-v4 {
            import all-be-v4;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
          forwarding-class fc-af4 {
```

```
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
        dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6 {
            import all-be-v6;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
    }
    traffic-control-profiles {
        tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id} {
            scheduler-map edge-8q-mix-${bw_profile_id};
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_egress_rate};
        }
        tcp-input-${total_ingress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id} {
            scheduler-map edge-input-8q-mix-${bw_profile_id};
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_ingress_rate};
        }
    }
    interfaces {
        ${cust_ifd} {
            unit ${cust_unit} {
                input-traffic-control-profile
                      tcp-input-${total_ingress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id};
                output-traffic-control-profile
                      tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id};
                classifiers {
                    dscp edge-8-classes-${mapping_profile_id}-v4;
                    dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6;
                }
            }
```

```
        }
    }
    scheduler-maps {
        edge-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-${ef_egress_pct}-high-rate-limit;
            forwarding-class fc-af1 scheduler sc-${af1_egress_pct}-dp-excess;
            forwarding-class fc-af2 scheduler sc-${af2_egress_pct}-dp-excess;
            forwarding-class fc-af3 scheduler sc-${af3_egress_pct}-dp-excess;
            forwarding-class fc-af4 scheduler sc-${af4_egress_pct}-dp-excess;
            forwarding-class fc-af-extra scheduler sc-${af5_egress_pct}-dp-excess;
            forwarding-class fc-nc scheduler sc-${nc_egress_pct}-medium-exact;
        }
        edge-input-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-${ef_ingress_pct}-high-small-buffer;
            forwarding-class fc-af1 scheduler sc-${af1_ingress_pct}-dp-excess;
            forwarding-class fc-af2 scheduler sc-${af2_ingress_pct}-dp-excess;
            forwarding-class fc-af3 scheduler sc-${af3_inggress_pct}-dp-excess;
            forwarding-class fc-af4 scheduler sc-${af4_inggress_pct}-dp-excess;
            forwarding-class fc-af-extra scheduler sc-${af5_ingress_pct}-dp-excess;
            forwarding-class fc-nc scheduler sc-${nc_ingress_pct}-medium-exact;
        }
    }
    schedulers {
        sc-${nc_conn_i_egress_pct}-medium-exact {
            transmit-rate {
                percent ${nc_conn_i_egress_pct};
                exact;
            }
            priority medium-high;
        }
        sc-${af1_egress_pct}-dp-excess {
            transmit-rate percent ${af1_egress_pct};
            excess-rate percent ${af1_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_egress_pct}-dp-excess {
            transmit-rate percent ${af2_egress_pct};
            excess-rate percent ${af2_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_egress_pct}-dp-excess {
            transmit-rate percent ${af3_egress_pct};
            excess-rate percent ${af3_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_egress_pct}-dp-excess {
            transmit-rate percent ${af4_egress_pct};
            excess-rate percent ${af4_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
```

```
        sc-${af5_egress_pct}-dp-excess {
            transmit-rate percent ${af5_egress_pct};
            excess-rate percent ${af5_egress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af1_ingress_pct}-dp-excess {
            transmit-rate percent ${af1_ingress_pct};
            excess-rate percent ${af1_ingress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_ingress_pct}-dp-excess {
            transmit-rate percent ${af2_ingress_pct};
            excess-rate percent ${af2_ingress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_ingress_pct}-dp-excess {
            transmit-rate percent ${af3_ingress_pct};
            excess-rate percent ${af3_ingress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_ingress_pct}-dp-excess {
            transmit-rate ${af4_ingress_pct};
            excess-rate ${af4_ingress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_ingress_pct}-dp-excess {
            transmit-rate ${af5_ingress_pct};
            excess-rate ${af5_ingress_pct};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        /* note! this scheduled does not have excess-rate configured, therefore this traffic
           is preempted by other traffic classes with excess-rate config */
        sc-remainder {
            transmit-rate {
                remainder;
            }
        }
        sc-${ef_egress_pct}-high-rate-limit {
            transmit-rate {
                percent ${ef_egress_pct};
                rate-limit;
            }
            priority high;
        }
       sc-${ef_ingress_pct}-high-small-buffer {
```

```
            transmit-rate percent ${ef_ingress_pct};
             shaping-rate percent ${ef_ingress_pct};
            /* Set buffer size to 1ms */
             buffer-size {
                 temporal 1000;
             }
             priority high;
        }
    }
}
```

The CoS template in Table 51 defines the seventh class-of-service profile. This profile targets a use case where several attachment circuits (or connections) share the same ingress and egress total bandwidth limits.

In this profile, per class and per connection limits are enforced individually in both ingress and egress directions, much like in the profile 5 CoS template in Table 49. However, aggregate bandwidth ingress rate limit is enforced for a group of connections. In other words, each connection should have its own connection-specific policers (shared between IPv4 and IPv6 address families) and one global policer for a group of the connections. This can only be implemented by configuring a unique set of filters and policers for each connection. At the same time, each filter should refer to the global policer that is specific to the connection group (these group names start with **hpol-**). This global policer must be shared. For this reason, it is declared as physical interface-specific. In its turn, all filters that refer to this policer are declared as physical interface filters.

It is assumed that the group of connections is identified by the unique **cust_id** parameter.

Table 51: Class-of-Service—Profile 7 Configuration

Template ID	cos-profile-7	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Moderate drop profile (dp-moderate) starts dropping traffic at 75 percent fill level (linear function).			
Aggressive drop profile (dp-aggressive) starts dropping traffic at 50 percent fill level (linear function).			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	An array of logical interfaces where attachment circuit is terminated		
total_ ingress_rate	Ingress policing rate (for both address families)		
total_egress_rate	Egress shaping rate (for both address families)		
ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to “shared-bandwidth-policer” in case of aggregated Ethernet customer IFD interface, otherwise left blank		
total_conn_ingress_rate	An array of total guaranteed ingress rates per connection		
total_conn_egress_rate	An array of total guaranteed egress rates per connection		
af1x_conn_ingress_rate	An array of AF1 traffic class committed ingress rate per connection		
af1x_conn_ingress_pct	An array of AF1 traffic class committed ingress rate in percent to the total ingress rate per connection		
af1x_conn_egress_rate	An array of AF1 traffic class committed egress rate per connection		
af1x_conn_egress_pct	An array of AF1 traffic class committed egress rate in percent to the total ingress rate per connection		
af2x_conn_ingress_rate	An array of AF2 traffic class committed ingress rate per connection		
af2x_conn_ingress_pct	An array of AF2 traffic class committed ingress rate in percent to the total ingress rate per connection		
af2x_conn_egress_rate	An array of AF2 traffic class committed egress rate per connection		
af2x_conn_egress_pct	An array of AF2 traffic class committed egress rate in percent to the total ingress rate per connection		
af3x_conn_ingress_rate	An array of AF3 traffic class committed ingress rate per connection		
af3x_conn_ingress_pct	An array of AF3 traffic class committed ingress rate in percent to the total ingress rate per connection		

af3x_conn_egress_rate	An array of AF3 traffic class committed egress rate per connection
af3x_conn_egress_pct	An array of AF3 traffic class committed egress rate in percent to the total ingress rate per connection
af4x_conn_ingress_rate	An array of AF4 traffic class committed ingress rate per connection
af4x_conn_ingress_pct	An array of AF4 traffic class committed ingress rate in percent to the total ingress rate per connection
af4x_conn_egress_rate	An array of AF4 traffic class committed egress rate per connection
af4x_conn_egress_pct	An array of AF4 traffic class committed egress rate in percent to the total ingress rate per connection
af5x_conn_ingress_rate	An array of AF5 traffic class committed ingress rate per connection
af5x_conn_ingress_pct	An array of AF5 traffic class committed ingress rate in percent to the total ingress rate per connection
af5x_conn_egress_rate	An array of AF5 traffic class committed egress rate per connection
af5x_conn_egress_pct	An array of AF5 traffic class committed egress rate in percent to the total ingress rate per connection
nc_conn_ingress_rate	An array of NC traffic class committed ingress rate per connection
nc_conn_ingress_pct	An array of NC traffic class committed ingress rate in percent to the total ingress rate per connection
nc_conn_egress_rate	An array of NC traffic class committed egress rate per connection
nc_conn_egress_pct	An array of NC traffic class committed egress rate in percent to the total ingress rate per connection
ef_conn_ingress_rate	An array of EF traffic class committed ingress rate per connection
ef_conn_ingress_pct	An array of EF traffic class committed ingress rate in percent to the total ingress rate per connection
ef_conn_egress_rate	An array of EF traffic class committed egress rate per connection
ef_conn_egress_pct	An array of EF traffic class committed egress rate in percent to the total ingress rate per connection
premium_rate	Sum of all ingress rate for all assured forwarding classes, network control and expedited forwarding for all connections
bw_profile_id	A number that identifies this particular combination of traffic rates
mapping_profile_id	A number that identifies particular mapping of DSCP fields to forwarding classes
cust_id	A number that identifies a group of connections
conn_id	An array of connection identifiers (1..N)

```
interfaces {
    interface-set ${cust_id} {
        interface ${cust_ifd} {
            /* all interface units pertaining to this connection group (ifl set)
               should be listed here */
            unit ${cust_unit[i]};
        }
    }
    ${cust_ifd} {
        unit ${cust_unit[i]} {
            family inet {
                filter {
                    input-list cos-${cust_id}-${conn_id}-v4;
                    output-list cos-unlimited-8-classes-egress-v4;
                }
            }
            family inet6 {
                filter {
                    input-list cos-${cust_id}-${conn_id}-v6;
                    output-list cos-unlimited-8-classes-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        /* A separate instance of this filter is created per each connection */
        filter cos-${cust_id}-${conn_id[i]}-v4 {
            /* declare it as physical interface filter in order to refer to a physical
               interface policer */
           physical-interface-filter;
```

```
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding ${af1_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate
            trtcm-${cust_id}-${conn_id[i]}-${af1_conn_ingress_rate[i]}-${total_ingress_rate}
            -ca-af1;
                    }
                    next term;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
                    /* mark traffic exceeding ${af2_conn_i_ingress_rate} as yellow,
                       discard traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate
            trtcm-${cust_id}-${conn_id[i]}-${af2_conn_ingress_rate[i]}-${total_ingress_rate}
            -ca-af2;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
                }
                then {
                    /* mark traffic exceeding ${af3_conn_i_ingress_rate} as yellow,
                       discard traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate
            trtcm-${cust_id}-${conn_id[i]}-${af3_conn_ingress_rate[i]}-${total_ingress_rate}
            -ca-af3;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
                then {
                    /* mark traffic exceeding ${af4_conn_i_ingress_rate} as yellow,
                       discard traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
```

```
                /* every term uses a dedicated policer, as logical interface policers
                   are filter-specific */
                two-rate
trtcm-${cust_id}-${conn_id[i]}-${af4_conn_ingress_rate[i]}-${total_ingress_rate}
-ca-af4;
            }
            /* set aggregated forwarding class */
            forwarding-class fc-af1;
            next term;
        }
    }
    term assured-forwarding-5 {
        from {
            forwarding-class fc-af-extra;
        }
        then {
            /* mark traffic exceeding ${af5_conn_i_ingress_rate} as yellow, discard
               traffic exceeding ${total_ingress_rate}*/
            three-color-policer {
                /* every term uses a dedicated policer, as logical interface policers
                   are filter-specific */
                two-rate
trtcm-${cust_id}-${conn_id[i]}-${af5_conn_ingress_rate[i]}-${total_ingress_rate}
-ca-af5;
            }
            /* set aggregated forwarding class */
            forwarding-class fc-af1;
            next term;
        }
    }
    term expedited-forwarding {
        from {
            forwarding-class fc-ef;
        }
        then {
            /* police traffic unconditionally*/
            policer pol-${cust_id}-${conn_id[i]}-${ef_conn_ingress_rate[i]}-ef;
            next term;
        }
    }
    term network-control {
        from {
            forwarding-class fc-nc;
        }
        then {
            /* police traffic unconditionally */
            policer pol-${cust_id}-${conn_id[i]}-${nc_conn_ingress_rate[i]}-nc;
            /* set aggregated forwarding class */
            forwarding-class fc-af1;
            next term;
        }
    }
     term treat-in-contract-as-premium {
        from {
            /* All traffic within the spec is declared premium */
            forwarding-class-except fc-be;
            loss-priority low;
        }
        then {
            force-premium;
            next term;
        }
    }
```

```
            term h-police-all {
                then {
                    /* Note! this policer is specific to a group of connections */
                    hierarchical-policer hpol-${cust_id}-${premium_rate}-${total_ingress_rate};
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic.
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen
           for this solution. */
        filter cos-unlimited-8-classes-egress-v4 {
            interface-specific;
            term expedited-forwarding {
                from {
                    dscp ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
                }
            }
            /* set loss priority first */
            term set-loss-priority {
                from {
                    dscp [af12 af13 af32 af33 af42 af43];
                }
                then {
                    loss-priority medium-low;
                    next term;
                }
            }
            /* set forwarding classes */
            term assured-forwarding-1 {
                from {
                    dscp [ af11 af12 af13 ];
                }
                then {
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    dscp [ af21 af22 af23 ];
                }
                then {
                    forwarding-class fc-af2;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    dscp [ af31 af32 af33 ];
                }
                then {
                    forwarding-class fc-af3;
                    accept;
                }
            }
```

```
            term assured-forwarding-4 {
                from {
                    dscp [ af41 af42 af43 ];
                }
                then {
                    forwarding-class fc-af4;
                    accept;
                }
            }
            term assured-forwarding-5 {
                from {
                    dscp cs5;
                }
                then {
                    forwarding-class fc-af-extra;
                    accept;
                }
            }
            term network-control {
                from {
                    dscp [ cs6 cs7 ];
                }
                then {
                    forwarding-class fc-nc;
                    accept;
                }
            }
            term accept-the-rest {
                /* the rest of the traffic is accepted, some host outbound network-control
                   traffic may hit this term, therefore forwarding class remains intact*/
                then accept;
            }
        }
    }
    family inet6 {
        /* A separate instance of this filter is created per each connection */
        filter cos-${cust_id}-${conn_id[i]}-v6 {
            /* declare it as physical interface filter in order to refer to a physical
               interface policer */
            physical-interface-filter;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding ${af1_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                        two-rate
            trtcm-${cust_id}-${conn_id[i]}-${af1_conn_ingress_rate[i]}-${total_ingress_rate}
            -ca-af1;
                    }
                    next term;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
```

```
                /* mark traffic exceeding ${af2_conn_i_ingress_rate} as yellow, discard
                   traffic exceeding ${total_ingress_rate}*/
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate
trtcm-${cust_id}-${conn_id[i]}-${af2_conn_ingress_rate[i]}-${total_ingress_rate}
-ca-af2;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                next term;
            }
        }
        term assured-forwarding-3 {
            from {
                forwarding-class fc-af3;
            }
            then {
                /* mark traffic exceeding ${af3_conn_i_ingress_rate} as yellow, discard
                   traffic exceeding ${total_ingress_rate}*/
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate
trtcm-${cust_id}-${conn_id[i]}-${af3_conn_ingress_rate[i]}-${total_ingress_rate}
-ca-af3;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                next term;
            }
        }
        term assured-forwarding-4 {
            from {
                forwarding-class fc-af4;
            }
            then {
                /* mark traffic exceeding ${af4_conn_i_ingress_rate} as yellow, discard
                   traffic exceeding ${total_ingress_rate}*/
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate
trtcm-${cust_id}-${conn_id[i]}-${af4_conn_ingress_rate[i]}-${total_ingress_rate}
-ca-af4;
                }
                /* set aggregated forwarding class */
                forwarding-class fc-af1;
                next term;
            }
        }
        term assured-forwarding-5 {
            from {
                forwarding-class fc-af-extra;
            }
            then {
                /* mark traffic exceeding ${af5_conn_i_ingress_rate} as yellow,
                   discard traffic exceeding ${total_ingress_rate}*/
                three-color-policer {
                    /* every term uses a dedicated policer, as logical interface policers
                       are filter-specific */
                    two-rate
```

```
            trtcm-${cust_id}-${conn_id[i]}-${af5_conn_ingress_rate[i]}-${total_ingress_rate}
            -ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
            term expedited-forwarding {
                from {
                    forwarding-class fc-ef;
                }
                then {
                    /* police traffic unconditionally*/
                    policer pol-${cust_id}-${conn_id[i]}-${ef_conn_ingress_rate[i]}-ef;
                    next term;
                }
            }
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${cust_id}-${conn_id[i]}-${nc_conn_ingress_rate[i]}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    next term;
                }
            }
             term treat-in-contract-as-premium {
                from {
                    /* All traffic within the spec is declared premium */
                    forwarding-class-except fc-be;
                    loss-priority low;
                }
                then {
                    force-premium;
                    next term;
                }
            }
            term h-police-all {
                then {
                    /* Note! this policer is specific to the group of connections */
                    hierarchical-policer hpol-${cust_id}-${premium_rate}-${total_ingress_rate};
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen for
           this solution. */
        filter cos-unlimited-8-classes-egress-v6 {
            interface-specific;
            term expedited-forwarding {
                from {
                    dscp ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
```

```
            }
        }
        /* set loss priority first */
        term set-loss-priority {
            from {
                dscp [ af12 af13 af32 af33 af42 af43];
            }
            then {
                loss-priority medium-low;
                next term;
            }
        }
        /* set forwarding classes */
        term assured-forwarding-1 {
            from {
                dscp [ af11 af12 af13 ];
            }
            then {
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-2 {
            from {
                dscp [ af21 af22 af23 ];
            }
            then {
                forwarding-class fc-af2;
                accept;
            }
        }
        term assured-forwarding-3 {
            from {
                dscp [ af31 af32 af33 ];
            }
            then {
                forwarding-class fc-af3;
                accept;
            }
        }
        term assured-forwarding-4 {
            from {
                dscp [ af41 af42 af43 ];
            }
            then {
                forwarding-class fc-af4;
                accept;
            }
        }
        term assured-forwarding-5 {
            from {
                dscp cs5;
            }
            then {
                forwarding-class fc-af-extra;
                accept;
            }
        }
        term network-control {
            from {
                dscp [ cs6 cs7 ];
            }
            then {
```

```
                forwarding-class fc-nc;
                accept;
            }
        }
        term accept-the-rest {
            /* the rest of the traffic is accepted, some host outbound network-control
               traffic may hit this term, therefore forwarding class remains intact*/
            then accept;
        }
    }
}
/* Policers below are declared per each customer connection */
policer pol-${cust_id}-${conn_id[i]}-${ef_conn_ingress_rate[i]}-ef {
    physical-interface-policer;
    /* Enable shared-bandwidth policing for AE interfaces only */
    ${shared_bandwidth};
    if-exceeding {
        bandwidth-limit ${ef_conn_ingress_rate[i]};
        /* Note! burst size is set to 35K to match shaper burst size */
        burst-size-limit 35k;
    }
    then discard;
}
policer pol-${cust_id}-${conn_id[i]}-${nc_conn_ingress_rate[i]}-nc {
    physical-interface-policer;
    /* Enable shared-bandwidth policing for AE interfaces only */
    ${shared_bandwidth};
    if-exceeding {
        bandwidth-limit ${nc_conn_ingress_rate[i]};
        /* Note! burst size is set to 35K to match shaper burst size */
        burst-size-limit 35k;
    }
    then discard;
}
three-color-policer
    trtcm-${cust_id}-${conn_id[i]}-${af1_conn_ingress_rate[i]}-${total_ingress_rate[i]}
    -ca-af1 {
    physical-interface-policer;
    /* Enable shared-bandwidth policing for AE interfaces only */
    ${shared_bandwidth};
    action {
        loss-priority high then discard;
    }
    two-rate {
        color-aware;
        /* Note! burst size is set to 35K to match shaper burst size */
        committed-information-rate ${af1_conn_ingress_rate[i]};
        committed-burst-size 35k;
        peak-information-rate ${total_ingress_rate};
        peak-burst-size 35k;
    }
}
three-color-policer
    trtcm-${cust_id}-${conn_id[i]}-${af1_conn_ingress_rate[i]}-${total_ingress_rate}
    -ca-af2 {
    physical-interface-policer;
    /* Enable shared-bandwidth policing for AE interfaces only */
    ${shared_bandwidth};
    action {
        loss-priority high then discard;
    }
    two-rate {
        color-aware;
```

```
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af2_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer
        trtcm-${cust_id}-${conn_id[i]}-${af3_conn_ingress_rate[i]}-${total_ingress_rate}
        -ca-af3 {
        physical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af3_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer
        trtcm-${cust_id}-${conn_id[i]}-${af4_conn_ingress_rate[i]}-${total_ingress_rate}
        -ca-af4 {
        physical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af4_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer
        trtcm-${cust_id}-${conn_id}-${af5_conn_ingress_rate[i]}-${total_ingress_rate}-ca-af5 {
        physical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af5_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
}
```

```
class-of-service {
    classifiers {
        dscp edge-8-classes-${mapping_profile_id}-v4 {
            import all-be-v4;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
        dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6 {
            import all-be-v6;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
    }
    traffic-control-profiles {
        /* As many egress profiles are declared as needed. All possible combinations of the
           guaranteed rates should be covered*/
```

```
        tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id} {
            scheduler-map edge-8q-mix-${bw_profile_id};
            overhead-accounting bytes -${ovhd_bytes};
            guaranteed-rate ${total_conn_i_egress_rate};
            shaping-rate ${total_egress_rate};
        }
        tcp-${total_egress_rate}-${ovhd_bytes} {
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_egress_rate};
        }
    }
    interfaces {
        interface-set ${cust_id} {
            output-traffic-control-profile tcp-${total_egress_rate}-${ovhd_bytes};
        }
        ${cust_ifd} {
        /* This configuration is repeated for all connections */
            unit ${cust_unit[i]} {
      output-traffic-control-profile tcp-${total_egress_rate}-${ovhd_bytes}-8q-mix-${bw_profile_id};
                classifiers {
                    dscp edge-8-classes-${mapping_profile_id}-v4;
                    dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6;
                }
            }
        }
    }
    scheduler-maps {
        edge-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-${ef_conn_egress_pct[i]}-high-rate-limit;
            forwarding-class fc-af1 scheduler sc-${af1_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af2 scheduler sc-${af2_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af3 scheduler sc-${af3_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af4 scheduler sc-${af4_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af-extra scheduler sc-${af5_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-nc scheduler sc-${nc_egress_pct[i]}-medium-exact;
        }
    }
    schedulers {
        sc-${nc_conn_egress_pct[i]}-medium-exact {
            transmit-rate {
                percent ${nc_conn_egress_pct[i]};
                exact;
            }
            priority medium-high;
        }
        sc-${af1_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af1_conn_egress_pct[i]};
            excess-rate percent ${af1_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af2_conn_egress_pct[i]};
            excess-rate percent ${af2_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
       sc-${af3_conn_egress_pct[i]}-dp-excess {
```

```
            transmit-rate percent ${af3_conn_egress_pct[i]};
            excess-rate percent ${af3_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af4_conn_egress_pct[i]};
            excess-rate percent ${af4_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af5_conn_egress_pct[i]};
            excess-rate percent ${af5_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af1_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af1_conn_ingress_pct[i]};
            excess-rate percent ${af1_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af2_conn_ingress_pct[i]};
            excess-rate percent ${af2_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af3_conn_ingress_pct[i]};
            excess-rate percent ${af3_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_conn_ingress_pct[i]}-dp-excess {
            transmit-rate ${af4_conn_ingress_pct[i]};
            excess-rate ${af4_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_conn_ingress_pct[i]}-dp-excess {
            transmit-rate ${af5_conn_ingress_pct[i]};
            excess-rate ${af5_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
```

```
        /* note! this scheduled does not have excess-rate configured, therefore this traffic is
           preempted by other traffic classes with excess-rate config */
        sc-remainder {
            transmit-rate {
                remainder;
            }
        }
        sc-${ef_egress_pct[i]}-high-rate-limit {
            transmit-rate {
                percent ${ef_egress_pct[i]};
                rate-limit;
            }
            priority high;
        }
    }
}
```

The profile 8 CoS template in Table 52 is similar to that of profile 7 in Table 51. The difference is that ingress shaping and queuing, instead of ingress policing, is enabled in profile 8.

In this profile, there is a need to maintain a shared global policer for a group of connections, because the enforcement is performed in the ingress traffic control profile at the IFL-set (group of connections) level. Ingress filters are still configured. However, like in profile 6 (Table 50), they are configured in order to identify high loss priority traffic and map edge traffic classes to the aggregate core classes. Because no global policer is required in the ingress filter, a number of connection groups can reference filters that share the same combination of ingress traffic rates. That combination is identified by the **bw_profile_id** parameter.

Table 52: Class-of-Service—Profile 8 Configuration

Template ID	cos-profile-8	**Device Role**	PE router, DCR, AS boundary router
Service	DIA, Layer 3 VPN	**Guidelines**	
Dependencies	edge-chassis-pic-queuing, edge-eth-intf-queuing		
Constants			
Policer burst size is set to 35 KB.			
Moderate drop profile (dp-moderate) starts dropping traffic at 75 percent fill level (linear function).			
Aggressive drop profile (dp-aggressive) starts dropping traffic at 50 percent fill level (linear function).			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
total_ ingress_rate	Ingress policing rate (for both address families)		
total_egress_rate	Egress shaping rate (for both address families)		
ovhd_bytes	L1+L2 overhead bytes, set to 42 for single VLAN tagged, to 46 for double-tagged, and to 38 for untagged		
shared_bandwidth	Junos OS command enabling shared bandwidth policer configuration, set to “shared-bandwidth-policer” in case of aggregated Ethernet customer IFD interface, otherwise left blank		
total_conn_ingress_rate	An array of total guaranteed ingress rates per connection		
total_conn_egress_rate	An array of total guaranteed egress rates per connection		
af1x_conn_ingress_rate	An array of AF1 traffic class committed ingress rate per connection		
af1x_conn_ingress_pct	An array of AF1 traffic class committed ingress rate in percent to the total ingress rate per connection		
af1x_conn_egress_rate	An array of AF1 traffic class committed egress rate per connection		
af1x_conn_egress_pct	An array of AF1 traffic class committed egress rate in percent to the total ingress rate per connection		
af2x_conn_ingress_rate	An array of AF2 traffic class committed ingress rate per connection		
af2x_conn_ingress_pct	An array of AF2 traffic class committed ingress rate in percent to the total ingress rate per connection		
af2x_conn_egress_rate	An array of AF2 traffic class committed egress rate per connection		
af2x_conn_egress_pct	An array of AF2 traffic class committed egress rate in percent to the total ingress rate per connection		
af3x_conn_ingress_rate	An array of AF3 traffic class committed ingress rate per connection		

af3x_conn_ingress_pct	An array of AF3 traffic class committed ingress rate in percent to the total ingress rate per connection
af3x_conn_egress_rate	An array of AF3 traffic class committed egress rate per connection
af3x_conn_egress_pct	An array of AF3 traffic class committed egress rate in percent to the total ingress rate per connection
af4x_conn_ingress_rate	An array of AF4 traffic class committed ingress rate per connection
af4x_conn_ingress_pct	An array of AF4 traffic class committed ingress rate in percent to the total ingress rate per connection
af4x_conn_egress_rate	An array of AF4 traffic class committed egress rate per connection
af4x_conn_egress_pct	An array of AF4 traffic class committed egress rate in percent to the total ingress rate per connection
af5x_conn_ingress_rate	An array of AF5 traffic class committed ingress rate per connection
af5x_conn_ingress_pct	An array of AF5 traffic class committed ingress rate in percent to the total ingress rate per connection
af5x_conn_egress_rate	An array of AF5 traffic class committed egress rate per connection
af5x_conn_egress_pct	An array of AF5 traffic class committed egress rate in percent to the total ingress rate per connection
nc_conn_ingress_rate	An array of NC traffic class committed ingress rate per connection
nc_conn_ingress_pct	An array of NC traffic class committed ingress rate in percent to the total ingress rate per connection
nc_conn_egress_rate	An array of NC traffic class committed egress rate per connection
nc_conn_egress_pct	An array of NC traffic class committed egress rate in percent to the total ingress rate per connection
ef_conn_ingress_rate	An array of EF traffic class committed ingress rate per connection
ef_conn_ingress_pct	An array of EF traffic class committed ingress rate in percent to the total ingress rate per connection
ef_conn_egress_rate	An array of EF traffic class committed egress rate per connection
ef_conn_egress_pct	An array of EF traffic class committed egress rate in percent to the total ingress rate per connection
premium_rate	Sum of all ingress rate for all assured forwarding classes, network control and expedited forwarding for all connections
bw_profile_id	A number that identifies this particular combination of traffic rates
mapping_profile_id	A number that identifies particular mapping of dscp fields to forwarding classes
cust_id	A number that identifies a group of connections
conn_id	An array of connection identifiers (1..N)

```
interfaces {
    interface-set ${cust_id} {
        interface ${cust_ifd} {
            /* all interface units pertaining to this connection group (ifl set)
               should be listed here */
            unit ${cust_unit[i]};
        }
    }
    ${cust_ifd} {
        unit ${cust_unit[i]} {
            family inet {
                filter {
        input-list cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v4;
                    output-list cos-unlimited-8-classes-egress-v4;
                }
            }
            family inet6 {
                filter {
        input-list cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v6;
                    output-list cos-unlimited-8-classes-egress-v6;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v4 {
            interface-specific;
            term assured-forwarding-1 {
```

```
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding ${af1_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af1_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af1;
                    }
                    accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
                    /* mark traffic exceeding ${af2_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af2_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af2;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
                }
                then {
                    /* mark traffic exceeding ${af3_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af3_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af3;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
                then {
                    /* mark traffic exceeding ${af4_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af4_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af4;
                    }
```

```
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-5 {
                from {
                    forwarding-class fc-af-extra;
                }
                then {
                    /* mark traffic exceeding ${af5_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af5_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            /* network-control and expedited-forwarding traffic is shaped by the input shaper,
               need only to reclassify */
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${nc_conn_ingress_rate[i]}${suffix}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term accept-the-rest {
                then {
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen for
           this solution. */
        filter cos-unlimited-8-classes-egress-v4 {
            interface-specific;
            term expedited-forwarding {
               from {
                    dscp ef;
                }
                then {
                    forwarding-class fc-ef;
                    accept;
                }
            }
            /* set loss priority first */
            term set-loss-priority {
                    from {
                            dscp [af12 af13 af32 af33 af42 af43];
                       }
```

```
                then {
                        loss-priority medium-low;
                         next term;
                }
        }
        /* set forwarding classes */
        term assured-forwarding-1 {
            from {
                dscp [ af11 af12 af13 ];
            }
            then {
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-2 {
            from {
                dscp [ af21 af22 af23 ];
            }
            then {
                forwarding-class fc-af2;
                accept;
            }
        }
        term assured-forwarding-3 {
            from {
                dscp [ af31 af32 af33 ];
            }
            then {
                forwarding-class fc-af3;
                accept;
            }
        }
        term assured-forwarding-4 {
            from {
                dscp [ af41 af42 af43 ];
            }
            then {
                forwarding-class fc-af4;
                accept;
            }
        }
        term assured-forwarding-5 {
            from {
                dscp cs5;
            }
            then {
                forwarding-class fc-af-extra;
                accept;
            }
        }
        term network-control {
            from {
                dscp [ cs6 cs7 ];
            }
            then {
                forwarding-class fc-nc;
                accept;
            }
        }
        term accept-the-rest {
           /* the rest of the traffic is accepted, some host outbound network-control
              traffic may hit this term, therefore forwarding class remains intact*/
```

```
                then accept;
            }
        }
    }
    family inet6 {
        filter cos-${total_ingress_rate}${suffix}-no-hpol-8-classes-mix-${bw_profile_id}-v6 {
            interface-specific;
            term assured-forwarding-1 {
                from {
                    forwarding-class fc-af1;
                }
                then {
                    /* mark traffic exceeding ${af1_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af1_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af1;
                        }
                        accept;
                }
            }
            term assured-forwarding-2 {
                from {
                    forwarding-class fc-af2;
                }
                then {
                    /* mark traffic exceeding ${af2_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af2_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af2;
                        }
                        /* set aggregated forwarding class */
                        forwarding-class fc-af1;
                        accept;
                }
            }
            term assured-forwarding-3 {
                from {
                    forwarding-class fc-af3;
                }
                then {
                    /* mark traffic exceeding ${af3_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af3_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af3;
                        }
                        /* set aggregated forwarding class */
                        forwarding-class fc-af1;
                        accept;
                }
            }
            term assured-forwarding-4 {
                from {
                    forwarding-class fc-af4;
                }
```

```
                then {
                    /* mark traffic exceeding ${af4_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af4_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af4;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term assured-forwarding-5 {
                from {
                    forwarding-class fc-af-extra;
                }
                then {
                    /* mark traffic exceeding ${af5_conn_i_ingress_rate} as yellow, discard
                       traffic exceeding ${total_ingress_rate}*/
                    three-color-policer {
                        /* every term uses a dedicated policer, as logical interface policers
                           are filter-specific */
                       two-rate trtcm-${af5_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
                       -ca-af5;
                    }
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            /* network-control and expedited-forwarding traffic is shaped by the input shaper,
               need only to reclassify */
            term network-control {
                from {
                    forwarding-class fc-nc;
                }
                then {
                    /* police traffic unconditionally */
                    policer pol-${nc_conn_ingress_rate[i]}-nc;
                    /* set aggregated forwarding class */
                    forwarding-class fc-af1;
                    accept;
                }
            }
            term accept-the-rest {
                then {
                    accept;
                }
            }
        }
        /* This output filter classifies traffic on PE egress. Another alternative would be to use
           dscp classifiers within a routing-instance, but this method only applies to VPN traffic
           Output multifield classifiers are applicable to DIA service as well unlike
           routing-instance classifiers. They are uniform and therefore they are chosen for
           this solution. */
        filter cos-unlimited-8-classes-egress-v6 {
            interface-specific;
            term expedited-forwarding {
                from {
                    dscp ef;
                }
                then {
```

```
                forwarding-class fc-ef;
                accept;
            }
        }
        /* set loss priority first */
        term set-loss-priority {
            from {
                dscp [ af12 af13 af32 af33 af42 af43];
            }
            then {
                loss-priority medium-low;
                next term;
            }
        }
        /* set forwarding classes */
        term assured-forwarding-1 {
            from {
                dscp [ af11 af12 af13 ];
            }
            then {
                forwarding-class fc-af1;
                accept;
            }
        }
        term assured-forwarding-2 {
            from {
                dscp [ af21 af22 af23 ];
            }
            then {
                forwarding-class fc-af2;
                accept;
            }
        }
        term assured-forwarding-3 {
            from {
                dscp [ af31 af32 af33 ];
            }
            then {
                forwarding-class fc-af3;
                accept;
            }
        }
        term assured-forwarding-4 {
            from {
                dscp [ af41 af42 af43 ];
            }
            then {
                forwarding-class fc-af4;
                accept;
            }
        }
        term assured-forwarding-5 {
            from {
                dscp cs5;
            }
            then {
                forwarding-class fc-af-extra;
                accept;
            }
        }
        term network-control {
            from {
                dscp [ cs6 cs7 ];
```

```
                }
                then {
                    forwarding-class fc-nc;
                    accept;
                }
            }
            term accept-the-rest {
                /* the rest of the traffic is accepted, some host outbound network-control
                   traffic may hit this term, therefore forwarding class remains intact*/
                then accept;
            }
        }
    }
    policer pol-${ef_conn_ingress_rate[i]}${suffix}-ef {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${ef_conn_ingress_rate[i]};
            /* Note! burst size is set to 35K to match shaper burst size */
            burst-size-limit 35k;
        }
        then discard;
    }
    policer pol-${nc_conn_ingress_rate[i]}${suffix}-nc {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        if-exceeding {
            bandwidth-limit ${nc_conn_ingress_rate[i]};
                        /* Note! burst size is set to 35K to match shaper burst size */
            burst-size-limit 35k;
        }
        then discard;
    }
    three-color-policer trtcm-${af1_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
    -ca-af1 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af1_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af1_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
    -ca-af2 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af2_conn_ingress_rate[i]};
```

```
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af3_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
    -ca-af3 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af3_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af4_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
    -ca-af4 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af4_conn_ingress_rate[i]};
            committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
    three-color-policer trtcm-${af5_conn_ingress_rate[i]}-${total_ingress_rate}${suffix}
    -ca-af5 {
        logical-interface-policer;
        /* Enable shared-bandwidth policing for AE interfaces only */
        ${shared_bandwidth};
        action {
            loss-priority high then discard;
        }
        two-rate {
            color-aware;
            /* Note! burst size is set to 35K to match shaper burst size */
            committed-information-rate ${af5_conn_ingress_rate[i]};
           committed-burst-size 35k;
            peak-information-rate ${total_ingress_rate};
            peak-burst-size 35k;
        }
    }
}
class-of-service {
    classifiers {
        dscp edge-8-classes-${mapping_profile_id}-v4 {
            import all-be-v4;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
```

```
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
        dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6 {
            import all-be-v6;
            forwarding-class fc-ef {
                loss-priority low code-points ef;
            }
            forwarding-class fc-af1 {
                loss-priority low code-points af11;
                loss-priority medium-low code-points [ af12 af13 ];
            }
            forwarding-class fc-af2 {
                loss-priority low code-points af21;
                loss-priority medium-low code-points [ af22 af23 ];
            }
            forwarding-class fc-af3 {
                loss-priority low code-points af31;
                loss-priority medium-low code-points [ af32 af33 ];
            }
            forwarding-class fc-af4 {
                loss-priority low code-points af41;
                loss-priority medium-low code-points [ af42 af43 ];
            }
            forwarding-class fc-af-extra {
                loss-priority low code-points cs5;
            }
            forwarding-class fc-nc {
                loss-priority low code-points [ cs6 cs7 ];
            }
        }
    }
    traffic-control-profiles {
        /* As many egress profiles are declared as needed. All possible combinations of the
           guaranteed rates should be covered */
        tcp-${total_egress_rate}-8q-mix-${bw_profile_id} {
            scheduler-map edge-8q-mix-${bw_profile_id};
            overhead-accounting bytes -${ovhd_bytes};
            guaranteed-rate ${total_conn_egress_rate[i]};
            shaping-rate ${total_egress_rate};
        }
        tcp-input-${total_conn_ingress_rate[i]}-${total_ingress_rate}-8q-mix-${bw_profile_id} {
            scheduler-map edge-input-8q-mix-${bw_profile_id};
```

```
            shaping-rate ${total_ingress_rate};
            overhead-accounting bytes -${ovhd_bytes};
            guaranteed-rate ${total_conn_ingress_rate[i]};
        }
        tcp-${total_egress_rate} {
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_egress_rate};
        }
        tcp-${total_ingress_rate} {
            overhead-accounting bytes -${ovhd_bytes};
            shaping-rate ${total_ingress_rate};
        }
    }
    interfaces {
        interface-set ${cust_id} {
            output-traffic-control-profile tcp-${total_egress_rate};
            input-traffic-control-profile tcp-${total_ingress_rate};
        }
        ${cust_ifd} {
            unit ${cust_unit} {
                output-traffic-control-profile tcp-${total_egress_rate}-8q-mix-${bw_profile_id};
                classifiers {
                    dscp edge-8-classes-${mapping_profile_id}-v4;
                    dscp-ipv6 edge-8-classes-${mapping_profile_id}-v6;
                }
            }
        }
    }
    scheduler-maps {
        edge-input-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-10-high-small-buffer;
            forwarding-class fc-af1 scheduler sc-10-excess;
            forwarding-class fc-af2 scheduler sc-10-excess;
            forwarding-class fc-af3 scheduler sc-10-excess;
            forwarding-class fc-af4 scheduler sc-10-excess;
            forwarding-class fc-af-extra scheduler sc-10-excess;
            forwarding-class fc-nc scheduler sc-5-medium-exact;
        }
        edge-8q-mix-${bw_profile_id} {
            forwarding-class fc-be scheduler sc-remainder;
            forwarding-class fc-ef scheduler sc-${ef_conn_egress_pct[i]}-high-rate-limit;
            forwarding-class fc-af1 scheduler sc-${af1_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af2 scheduler sc-${af2_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af3 scheduler sc-${af3_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af4 scheduler sc-${af4_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-af-extra scheduler sc-${af5_conn_egress_pct[i]}-dp-excess;
            forwarding-class fc-nc scheduler sc-${nc_egress_pct}-medium-exact;
        }
    }
    schedulers {
        sc-${ef_conn_ingress_pct[i]}-high-small-buffer {
            transmit-rate percent ${ef_conn_ingress_pct[i]};
            shaping-rate percent ${ef_conn_ingress_pct[i]};
            /* Set buffer size to 1ms */
            buffer-size {
                temporal 1000;
            }
            priority high;
        }
        sc-${nc_conn_egress_pct[i]}-medium-exact {
            transmit-rate {
               percent ${nc_conn_egress_pct[i]};
```

```
                exact;
            }
            priority medium-high;
        }
        sc-${af1_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af1_conn_egress_pct[i]};
            excess-rate percent ${af1_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af2_conn_egress_pct[i]};
            excess-rate percent ${af2_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af3_conn_egress_pct[i]};
            excess-rate percent ${af3_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af4_conn_egress_pct[i]};
            excess-rate percent ${af4_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_conn_egress_pct[i]}-dp-excess {
            transmit-rate percent ${af5_conn_egress_pct[i]};
            excess-rate percent ${af5_conn_egress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af1_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af1_conn_ingress_pct[i]};
            excess-rate percent ${af1_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af2_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af2_conn_ingress_pct[i]};
            excess-rate percent ${af2_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af3_conn_ingress_pct[i]}-dp-excess {
            transmit-rate percent ${af3_conn_ingress_pct[i]};
            excess-rate percent ${af3_conn_ingress_pct[i]};
```

```
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af4_conn_ingress_pct[i]}-dp-excess {
            transmit-rate ${af4_conn_ingress_pct[i]};
            excess-rate ${af4_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        sc-${af5_conn_ingress_pct}-dp-excess {
            transmit-rate ${af5_conn_ingress_pct[i]};
            excess-rate ${af5_conn_ingress_pct[i]};
            drop-profile-map loss-priority low protocol any drop-profile dp-moderate;
            drop-profile-map loss-priority medium-low protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority medium-high protocol any drop-profile dp-aggressive;
            drop-profile-map loss-priority high protocol any drop-profile dp-aggressive;
        }
        /* note! this scheduled does not have excess-rate configured, therefore this traffic is
           preempted by other traffic classes with excess-rate config */
        sc-remainder {
            transmit-rate {
                remainder;
            }
        }
        sc-${ef_egress_pct[i]}-high-rate-limit {
            transmit-rate {
                percent ${ef_egress_pct[i]};
                rate-limit;
            }
            priority high;
        }
    }
}
```

Direct Internet Access Service Specific Configuration

Upstream and Peer Attachment Circuit

The template in Table 53 is applied to the DIA peer and upstream attachment circuit.

Table 53: DIA Upstream and Peer Attachment Circuit Configuration

Template ID	ip-attachment-circuit-peer-upstream	**Device Role**	AS boundary router
Service	DIA	**Guidelines**	Use for peer or upstream attachment circuit
Dependencies	svc-pe-global		
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
router_ipv4	Router IPv4 address		
router_ipv6	Router IPv4 address		

```
interfaces {
    ${peer_ifd} {
        /* Peer IFL configuration */
        unit ${peer_ifl} {
            /* Apply DIA specific configuration */
```

```
            apply-groups dia-ifl-settings;
            family inet {
                /* RPF check is done in loose mode */
                rpf-check {
                    mode loose;
                }
                address ${router_ipv4};
            }
            family inet6 {
                /* RPF check is done in loose mode */
                rpf-check {
                    mode loose;
                }
                address ${router_ipv6};
            }
        }
    }
}
```

Community Definitions

The template in Table 54 defines BGP communities used in the various BGP policies defined later in this document. This template should be applied to all AS boundary, PE, DCR, and router reflector routers.

Table 54: DIA Community Definitions Configuration

Template ID	dia-community	Device Role	AS boundary router, PE router, DCR, route reflector
Service	DIA	Guidelines	
Dependencies			
Parameters			
as	Autonomous system number of our autonomous system		
region_id	Three-digit string identifying a region to which this router belongs		

```
policy-options {
    /* Matches any community */
    community comm-all members *:*;

    /* Matches all communities related to my AS number */
    community comm-all-my-as members "^${as}:*";

    /* Identifies customer prefixes */
    community comm-customer members ${as}:3000;
    /* Identifies peer prefixes */
    community comm-peer members ${as}:2000;
    /* Identifies upstream prefixes */
    community comm-upstream members ${as}:1000;

    /* Matches all customer communities */
    community comm-all-type members "^${as}:.000$";

    /* Accept community, used internally within the router during policy processing,
       signals that the route should be accepted by the final policy in the list */
    community comm-accept members ${as}:10000;

    /* Identifier of the region this router belongs to. Region-specific policies are not
       used in this solution, they are specific to a particular deployment.
       This is provided for completeness */
    community comm-region-id members ${as}:${region_id};

    /* Provided as an example in RFC 5635 and now used by some ISPs to
       signal remote triggered black hole */
    community comm-remote-triggered-black-hole members ${as}:666;
```

```
    /* RFC 1998 communities to signal local preference from CE device to PE device */
    community comm-lclpref-70 members ${as}:70;
    community comm-lclpref-80 members ${as}:80;
    community comm-lclpref-90 members ${as}:90;

    /* Standard no-export community */
    community comm-no-export members no-export;
}
```

Static and Aggregate Routes

The template in Table 55 defines default routes and aggregate routes. The default routes (0/0 and ::/0) might later be advertised to DIA customers using BGP.

IPv4 route 192.0.2.1 and IPv6 mapped route ::ffff:192.0.2.1/128 are used as next hops for black-hole routes installed by customers. Those routes point to discard interfaces. They are allocated from the address block of nonroutable addresses and should not create any conflict with normal Internet routes.

Finally, a number of aggregate routes are defined. In the BGP policy templates, those aggregate routes are announced to peers and upstream interfaces, and more specific contributing (customer) routes are discarded.

Table 55: DIA Routing Configuration—Static and Aggregate Routes

Template ID	dia-routes	**Device Role**	AS boundary router, PE router, DCR, route reflector
Service	DIA	**Guidelines**	
Parameters			
as	Autonomous system number of our autonomous system		
my_ipv4_aggregate	Array of aggregate IPv4 routes		
my_ipv6_aggregate	Array of aggregate IPv6 routes		

```
routing-options {
    static {
        route 0/0 {
            discard;
            /* Do not install the route into the forwarding table */
            no-install;
        }
        /* IPv4 black hole routes should be set up with the 192.0.2.1 next-hop
           The 192.0.2.0/24 network is only used for documentation and testing,
           therefore it is safe to configure addresses from this network as
           black hole destination.
           There is a guarantee that there will be no valid destination with
           this prefix assigned */
        route 192.0.2.1/32 {
            discard;
            no-readvertise;
        }
    }
    aggregate {
        route ${my_ipv4_aggregate[i]} {
            /* Mark aggregates with a customer community */
            community ${as}:3000;
            /* Include longest common sequences from contributing paths */
            brief;
            /* Retain inactive */
            passive;
        }
    }
    rib inet6.0 {
        static {
```

```
            route ::/0 {
                discard;
                /* Do not install the route into forwarding table */
                no-install;
            }
            /* IPv6 black hole routes should be set up with the ::ffff:192.0.2.1 next-hop
               The 192.0.2.0/24 network is only used for documentation and testing
               (same applies to IPv6 mapped prefix), therefore it is safe to configure
               addresses from this network as black hole destination.
               There is a guarantee that there will be no valid destination with this
               prefix assigned */
            route ::ffff:192.0.2.1/128 {
                discard;
                no-readvertise;
            }
        }
        aggregate {
            route ${my_ipv6_aggregate[i]} {
                /* Mark aggregates with a customer community */
                community ${as}:3000;
                /* Include longest common sequences from contributing paths */
                brief;
                /* Retain inactive */
                passive;
            }
        }
    }
}
```

Common BGP Configuration

The template in Table 56 defines several routing policies that are used by other configuration templates defined later in this document.

Table 56: DIA Routing Configuration—Common BGP Configuration

Template ID	dia-policy	Device Role	AS boundary router, PE router, DCR
Service	DIA	Guidelines	
Dependencies	dia-community, address-space		

```
policy-options {
    policy-statement pol-reject-martians {
        /* Reject bogon prefixes */
        term reject-martians-v4 {
            from {
                /* Any 0/X prefix, X == 0..32 */
                route-filter 0.0.0.0/0 through 0.0.0.0/32 reject;

                /* Loopback address */
                route-filter 127.0.0.0/8 orlonger reject;

                /* RFC 1918 private address space */
                route-filter 10.0.0.0/8 orlonger reject;
                route-filter 172.16.0.0/12 orlonger reject;
                route-filter 192.168.0.0/16 orlonger reject;

                /* Link local prefixes */
                route-filter 169.254.0.0/16 orlonger reject;

                /* RFC 6333 DS-Lite private address space */
                route-filter 192.0.0.0/29 orlonger reject;

                /* RFC 2544, prefixes for testing */
                route-filter 198.18.0.0/15 orlonger reject;
```

```
            /* RFC 6598 Private address space for CG NAT applications */
            route-filter 100.64.0.0/10 orlonger reject;

            /* RFC 5737, prefixes for documentation */
            route-filter 192.0.2.0/24 orlonger reject;
            route-filter 198.51.100.0/24 orlonger reject;
            route-filter 203.0.113.0/24 orlonger reject;

            /* Multicast group addresses */
            route-filter 224.0.0.0/4 orlonger reject;
            route-filter 240.0.0.0/4 orlonger reject;
        }
        then reject;
    }
    term reject-martians-v6 {
        from {
            /* Reject default */
            route-filter ::/0 exact reject;

            /* Reject unspecified address */
            route-filter ::/128 exact reject;

            /* Reject loopbacks address */
            route-filter ::1/128 exact reject;

            /* Reject IPv4-mapped address */
            route-filter ::ffff:0.0.0.0/96 orlonger reject;

            /* Reserved by IETF, includes IPv4 compatible address space */
            route-filter 0000::/8 orlonger reject;

            /* RFC 4913 Unique Local Unicast Address Space */
            route-filter fc00::/7 orlonger reject;

            /* Link Local Unicast Address */
            route-filter fe80::/10 orlonger reject;

            /* Deprecated site-local address prefix, see RFC 3879 */
            route-filter fec0::/10 orlonger reject;

            /* 6bone Decommissioned Address Space */
            route-filter 3ffe::/16 orlonger reject;

            /* IPv6 Documentation Prefix */
            route-filter 2001:db8::/32 orlonger reject;

            /* Multicast */
            route-filter ff00::/8 orlonger reject;

            /* 6to4 mapped IPv4 Default 0.0.0.0/X, X=0..32 */
            route-filter 2002:0000::/16 orlonger reject;

            /* 6to4 mapped IPv4 Loopback 127.0.0.0/8 */
            route-filter 2002:7f00::/24 orlonger reject;

            /* 6to4 mapped RFC 1918 private address space */
            /* 10.0.0.0/8 */
            route-filter 2002:0a00::/24 orlonger reject;
            /* 172.16.0.0/12 */
            route-filter 2002:ac10::/28 orlonger reject;
            /* 192.168.0.0/16 */
           route-filter 2002:c0a8::/32 orlonger reject;
```

```
                /* 6to4 mapped Link local 169.254.0.0/16 prefix */
                route-filter 2002:a9fe::/32 orlonger reject;

                /* 6to4 mapped RFC 6333 DS-Lite private address space */
                route-filter 2002:c000::/45 orlonger reject;

                /* 6to4 mapped RFC 2544, 198.18.0.0/15 prefix for testing */
                route-filter 2002:c612::/31 orlonger reject;

                /* 6to4 mapped RFC 6598, 100.64.0.0/10 prefix for CG NAT application */
                route-filter 2002:6440::/26 orlonger reject;

                /* 6to4 mapped RFC 5737, prefixes for documentation */
                /* 192.0.2.0/24 */
                route-filter 2002:c000:200::/40 orlonger reject;
                /* 198.51.100.0/24 */
                route-filter 2002:c633:6400::/40 orlonger reject;
                /* 203.0.113.0/24 */
                route-filter 2002:cb00:7100::/40 orlonger reject;

                /* 6to4 mapped IPv4 Multicast 224.0.0.0/4 */
                route-filter 2002:e000::/20 orlonger reject;
                /* 6to4 mapped IPv4 Multicast 240.0.0.0/4 */
                route-filter 2002:f000::/20 orlonger reject;
            }
            then reject;
        }
    }
    policy-statement pol-export-directs {
        /* Reject direct routes from the V4 infrastructure range,
           they should not be reachable */
        term reject-infrastructure-directs-v4 {
            from {
                protocol direct;
                prefix-list-filter pl-infra-v4 orlonger;
            }
            then reject;
        }
        /* Select direct routes from our V4 address range */
        term select-my-directs-v4 {
            from {
                protocol direct;
                prefix-list-filter pl-my-as-routes-v4 orlonger;
            }
            then {
                community add comm-accept;
                next policy;
            }
        }
        /* Reject direct routes from the V6 infrastructure range,
           they should not be reachable */
        term reject-infrastructure-directs-v6 {
            from {
                protocol direct;
                prefix-list-filter pl-infra-v6 orlonger;
            }
            then reject;
        }
        /* Select direct routes from our V6 address range */
        term select-my-directs-v6 {
            from {
                protocol direct;
                prefix-list-filter pl-my-as-routes-v6 orlonger;
```

```
            }
            then {
                community add comm-accept;
                next policy;
            }
        }
    }
    policy-statement pol-export-peer-or-upstream {
        /* Suppress specific V4 routes within our range */
        term supress-specifics-v4 {
            from {
                protocol [direct static bgp];
                prefix-list-filter pl-my-as-routes-v4 orlonger;
            }
            then next policy;
        }
        /* Suppress specific V6 routes within our range */
        term supress-specifics-v6 {
            from {
                protocol [direct static bgp];
                prefix-list-filter pl-my-as-routes-v6 orlonger;
            }
            then next policy;
        }
        /* Export other customer routes (not from our AS) */
        term select-customer-routes {
            /* Select only customer routes for export */
            from community comm-customer;
            then {
                /* Remove communities identifying internal route type */
                community delete comm-all-type;
                /* Mark them with comm-accept community, final accept is
                   happening in the pol-final */
                community add comm-accept;
                next policy;
            }
        }
    }
    policy-statement pol-export-v4-default {
        term accept-default {
            from route-filter 0/0 exact;
            then accept;
        }
        term reject-the-rest {
            then reject;
        }
    }
    policy-statement pol-export-v6-default {
        term accept-default {
            from route-filter ::/0 exact;
            then accept;
        }
        term reject-the-rest {
            then reject;
        }
    }
    policy-statement pol-final {
        /* Accept routes marked with comm-accept community,
           these routes are selected for export at preceding steps */
        term final-action {
            from community comm-accept;
            then {
                community delete comm-accept;
```

```
                accept;
            }
        }
        /* Other routes are rejected */
        term reject-the-rest {
            then reject;
        }
    }
    policy-statement pol-import-customer-default {
        /* Reject our own infrastructure networks */
        term reject-my-networks-v4 {
            from {
                prefix-list-filter pl-infra-v4 orlonger;
            }
            then reject;
        }
        term reject-my-networks-v6 {
            from {
                prefix-list-filter pl-infra-v6 orlonger;
            }
            then reject;
        }
        /* Set local preference to the routes according to the customer's request, RFC 1998 */
        term set-lcl-pref-70 {
            from {
                community comm-lclpref-70;
            }
            then {
                local-preference 70;
                next term;
            }
        }
        term set-lcl-pref-80 {
            from {
                community comm-lclpref-80;
            }
            then {
                local-preference 80;
                next term;
            }
        }
        term set-lcl-pref-90 {
            from {
                community comm-lclpref-90;
            }
            then {
                local-preference 90;
                next term;
            }
        }
        /* Assign communities / accept the route */
        term set-defaults {
            then {
                community delete comm-all-my-as;
                community add comm-customer;
                community add comm-region-id;
                community add comm-accept;
                /* Allow route processing in next policies */
                next policy;
            }
```

```
        }
    }
    policy-statement pol-import-peer-default {
        /* Reject our own infrastructure networks */
        term reject-my-networks-v4 {
            from {
                prefix-list-filter pl-infra-v4 orlonger;
            }
            then reject;
        }
        term reject-my-networks-v6 {
            from {
                prefix-list-filter pl-infra-v6 orlonger;
            }
            then reject;
        }
        /* Assign communities / accept the route */
        term set-defaults {
            then {
                community delete comm-all-my-as;
                community add comm-peer;
                community add comm-region-id;
                community add comm-accept;
                /* Allow route processing in next policies */
                next policy;
            }
        }
    }
    policy-statement pol-import-upstream-default {
        /* Reject our own infrastructure networks */
        term reject-my-networks-v4 {
            from {
                prefix-list-filter pl-infra-v4 orlonger;
            }
            then reject;
        }
        term reject-my-networks-v6 {
            from {
                prefix-list-filter pl-infra-v6 orlonger;
            }
            then reject;
        }
        /* Assign communities / accept the route */
        term set-defaults {
            then {
                community delete comm-all-my-as;
                community add comm-upstream;
                community add comm-region-id;
                community add comm-accept;
                /* Allow route processing in next policies */
                next policy;
            }
        }
    }
}
```

Attachment Circuit Routing—Static Routing

The template in Table 57 defines static routing configuration for DIA service.

Table 57: DIA Attachment Circuit Routing—Static Configuration

Template ID	dia-customer-static	Device Role	PE router, DCR
Service	DIA	Guidelines	Static routing for customer attachment circuits
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
route_v4	Array of IPv4 static routes (multiple routes possible)		
route_v6	Array of IPv6 static routes (multiple routes possible)		

```
routing-options {
    /* IPv4 routes */
    static {
        route ${route_v4[i]} {
            /* Mark with the customer and region id community */
            community [ ${as}:3000 ${as}:${region_id} ];
            next-hop ${ce_v4};
        }
    }
    /* IPv6 routes */
    rib inet6.0 {
        static {
            route ${route_v6[i]} {
                /* Mark with the customer community */
                community [ ${as}:3000 ${as}:${region_id} ];
                next-hop ${ce_v6};
            }
        }
    }
}
```

Static routes are defined in the inet.0 routing table. Note that multiple routes pointing to the same CE device might exist. A “customer” community and “region ID” community is assigned to each static route. The customer community is used to identify routes in various BGP import and export policies.

The template in Table 58 enables BFD for static routes.

Table 58: DIA Attachment Circuit Routing—Static and BFD Configuration

Template ID	dia-customer-static-bfd	Device Role	PE router, DCR
Service	DIA	Guidelines	Static routing for customer attachment circuits
Parameters			
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
route_v4	Array of IPv4 static routes (multiple routes possible)		
route_v6	Array of IPv6 static routes (multiple routes possible)		

```
groups {
    bfd-static-route-settings {
        routing-options {
            static route <*> {
                bfd-liveness-detection {
                    minimum-interval ${bfd_min_interval};
                    multiplier ${bfd_multiplier};
                }
            }
            rib inet6.0 {
                static route <*> {
                    bfd-liveness-detection {
                        minimum-interval ${bfd_min_interval};
                        multiplier ${bfd_multiplier};
                    }
                }
            }
        }
    }
}
routing-options {
    /* IPv4 routes */
    static {
        route ${route_v4[i]} {
            /* Configure BFD */
            apply-groups bfd-static-route-settings;
        }
    }
    /* IPv6 routes */
    rib inet6.0 {
        static {
            route ${route_v6[i]} {
                /* Configure BFD */
                apply-groups bfd-static-route-settings;
            }
        }
    }
}
```

Attachment Circuit Routing—BGP Routing

BGP routing configuration at the attachment circuit is provisioned in two steps:

1. Depending on the required routing export policy, either the **dia-customer-bgp-full-view** or **dia-customer-bgp-default-only** template is chosen.
2. Depending on the route import policy, one of the four BGP import policy templates is chosen.

The template in Table 59 defines BGP routing configuration for an attachment circuit with export of the full IPv4/IPv6 Internet routing table. In this template, two BGP sessions are established to the CE device; one for IPv4 and another for IPv6. BFD is enabled for both sessions. There is a limit of 1,000 prefixes that are accepted on each session.

Export policies are configured in such a way as to enable the export of direct routes (that is, all customer, uplink, and peer routes).

NOTE: The BGP neighbor group name does follow a special meaning convention. The name used identifies peers with the Generalized TTL Security Mechanism enabled. Neighbors from groups with names that start with **v4-gtsm** or **v6-gtsm** are included in the prefix list and matched in the loopback filter (see Table 15, Network Element Control Plane Protection).

Table 59: DIA Attachment Circuit Routing—BGP Routing

Template ID	dia-customer-bgp-full-view	Device Role	PE router, DCR
Service	DIA	Guidelines	BGP routing for customer attachment circuits, full view export
Dependencies	dia-policy		
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Peer AS number		
gtsm	Set to "gtsm-" for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		

```
groups {
    inet-max-1000-prefixes {
        protocols {
            bgp {
                group <*> {
                    neighbor <*> {
                        family inet {
                            unicast {
                                prefix-limit {
                                    maximum 1000;
                                  teardown 75;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    inet6-max-1000-prefixes {
        protocols {
            bgp {
                group <*> {
                    neighbor <*> {
                        family inet6 {
                            unicast {
                                prefix-limit {
                                    maximum 1000;
                                    teardown 75;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

protocols {
    bgp {
        /* Neighbors in this group receive all Internet IPv4 routes */
        group v4-${gtsm}customer-full-view {
            type external;
            /* Export all routes to the customer, including direct routes */
            export [ pol-export-directs pol-export-customer pol-final ];
            /* Remove private AS from the AS-PATH */
            remove-private;
```

```
            neighbor ${ce_v4} {
                apply-groups [ inet-max-1000-prefixes ];
                family inet {
                    unicast;
                }
                /* Route Import Policy is specified in a separate template */
                peer-as ${peer_as};
            }
        }
        /* Neighbors in this group receive all Internet IPv6 routes */
        group v6-${gtsm}customer-full-view {
            type external;
            /* Export all routes to the customer */
            export [ pol-export-directs pol-export-customer pol-final ];
            /* Remove private AS from the AS-PATH */
            remove-private;
            neighbor ${ce_v6} {
                /* Limit number of accepted prefixes to 1000 */
                apply-groups [ inet6-max-1000-prefixes ];
                family inet6 {
                    unicast;
                }
                /* Route Import Policy is specified in a separate template */
                peer-as ${peer_as};
            }
         }
    }
}

policy-options {
    policy-statement pol-export-customer {
        /* This term select all BGP and Static routes marked with specific communities */
        term select-all {
            from community [ comm-all-peers comm-all-customers comm-all-upstreams ];
            then {
                community delete comm-all-type;
                community add comm-accept;
                next policy;
            }
        }
    }
}
```

The template in Table 60 is used to provision BGP routing configuration for customers that only expect default routes from the carrier. The only difference to the previous template is the BGP export policy, which allows only default routes. The rest of the configuration remains the same.

Table 60: DIA Attachment Circuit Routing—BGP Export Default Route Policy

Template ID	dia-customer-bgp-default-only	**Device Role**	PE router, DCR
Service	DIA	**Guidelines**	BGP routing for customer attachment circuits, default route export
Dependencies	dia-policy		
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Peer AS number		
gtsm	Set to “gtsm-” for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		

```
groups {
    inet-max-1000-prefixes {
        protocols {
            bgp {
                group <*> {
                    neighbor <*> {
                        family inet {
                            unicast {
                                prefix-limit {
                                    maximum 1000;
                                    teardown 75;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    inet6-max-1000-prefixes {
        protocols {
            bgp {
                group <*> {
                    neighbor <*> {
                        family inet6 {
                            unicast {
                                prefix-limit {
                                    maximum 1000;
                                    teardown 75;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

protocols {
    bgp {
        /* Neighbors in this group receive only default IPv4 route */
        group v4-${gtsm}customer-default-only {
            type external;
            /* Export only default routes to the customer */
            export pol-export-v4-default;
            /* Remove private AS from the AS-PATH */
            remove-private;
            neighbor ${ce_v4} {
                apply-groups [ inet-max-1000-prefixes ];
                family inet {
                    unicast;
                }
                /* Route Import Policy is specified in a separate template */
                peer-as ${peer_as};
            }
        }
        /* Neighbors in this group receive only default IPv6 route */
        group v6-${gtsm}customer-default-only {
            type external;
            /* Export only default routes to the customer */
            export pol-export-v6-default;
            /* Remove private AS from the AS-PATH */
            remove-private;
```

```
            neighbor ${ce_v6} {
                /* Limit number of accepted prefixes to 1000 */
                apply-groups [ inet6-max-1000-prefixes ];
                family inet6 {
                    unicast;
                }
                /* Route Import Policy is specified in a separate template */
                peer-as ${peer_as};
            }
        }
    }
}
```

BFD can be enabled on BGP sessions using the template in Table 61.

Table 61: DIA Attachment Circuit Routing—BGP Routing with BFD Configuration

Template ID	dia-bgp-bfd	**Device Role**	PE router, DCR
Service	DIA	**Guidelines**	BGP routing for customer attachment circuits, default route export
Dependencies	dia-policy		
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		
gtsm	Set to “gtsm-” for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		

```
groups {
    bgp-bfd-settings {
        protocols {
            bgp {
                group <*> {
                    neighbor <*> {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
            }
        }
    }
}

protocols {
    bgp {
        group v4-${gtsm}customer-default-only {
            neighbor ${ce_v4} {
                apply-groups [ bgp-bfd-settings ];
            }
        }
       group v6-${gtsm}customer-default-only {
            neighbor ${ce_v6} {
                apply-groups [ bgp-bfd-settings ];
            }
        }
    }
}
```

The following four templates define various BGP import policies.

The template in Table 62 unconditionally imports all routes from the customer with some minimal route validation (it discards martian routes and routes from the address space belonging to the carrier). The template refers to policy definitions from the **dia-policy** template. The only difference between peer, customer, and uplink policies is the route community tagging.

Table 62: DIA Attachment Circuit Routing—BGP Routing Import Policy Configuration

Template ID	dia-bgp-import-policy-1	**Device Role**	PE router, DCR
Service	DIA	**Guidelines**	Unconditional import of all routes
Dependencies	dia-policy		
Parameters			
peer_as	Peer AS number		
gtsm	Set to "gtsm-" for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		
group_suffix	Set to "-full-view" for neighbors receiving full view BGP routing table, set to "-default-only" for neighbors receiving default only route		
peer_type	Set to "customer" if the peer is the of a customer type, set to "peer" if the peer type is a peer, set to "upstream" if the peer type is an upstream		

```
protocols {
    bgp {
        /* Neighbors in this group receive all Internet IPv4 routes */
        group v4-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v4} {
                /*  Example configuration of the profile #1 - all prefixes are accepted with
                    minimal constraints */
                import [ pol-reject-martians pol-import-${peer_type}-default pol-final ];
            }
        }
        /* Neighbors in this group receive all Internet IPv6 routes */
        group v6-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v6} {
                /* Example configuration of the profile #1 - all prefixes are accepted with
                   minimal constraints */
                import [ pol-reject-martians pol-import-${peer_type}-default pol-final ];
            }
        }
    }
}
```

The template in Table 63 is used for peers with strict prefix validation. Prefixes expected from that peer are explicitly listed in a prefix list and that prefix list is used in the peer-specific import policy.

Table 63: DIA Attachment Circuit Routing—BGP Routing Import with Strict Prefix Validation

Template ID	dia-bgp-import-policy-2	**Device Role**	PE router, DCR
Service	DIA	**Guidelines**	Import of a select set of customer routes
Dependencies	dia-policy		
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Peer AS number		
allowed_v4_prefix	Array of IPv4 prefixes allowed from that peer		
allowed_v6_prefix	Array of IPv6 prefixes allowed from that peer		

gtsm	Set to “gtsm-” for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank
group_suffix	If the peer type is customer, set to “-full-view” for neighbors receiving full view BGP routing table, set to “-default-only” for neighbors receiving default only route; if the peer type is upstream or peer, leave blank
peer_type	Set to “customer” if the peer is of a customer type, set to “peer” if the peer type is a peer, set to “upstream” if the peer type is an upstream

```
protocols {
    bgp {
        group v4-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v4} {
                /* Use customer specific import policy */
                import [ pol-prefix-allow-${peer_as} pol-import-${peer_type}-default pol-final ];
            }
        }
        /* Neighbors in this group receive all Internet IPv6 routes */
        group v6-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v6} {
                /* Use customer specific import policy */
                import [ pol-prefix-allow-${peer_as} pol-import-${peer_type}-default pol-final ];
            }
        }
    }
}

policy-options {
    /* Prefix list containing all V4 prefixes allowed to accept from the peer AS */
    prefix-list pl-allowed-${peer_as}-v4 {
        ${allowed_v4_prefix[i]};
    }
    /* Prefix list containing all V6 prefixes allowed to accept from the peer AS */
    prefix-list pl-allowed-${peer_as}-v6 {
        ${allowed_v6_prefix[i]};
    }
    /* This policy is customer specific. It allows only certain routes from the customer */
    policy-statement pol-prefix-allow-${peer_as} {
        term allow-v4 {
            from {
                family inet;
                /* Allow only exact matches, suppress specifics */
                prefix-list-filter pl-allowed-${peer_as}-v4 exact;
            }
            then next policy;
        }
        term allow-v6 {
            from {
                family inet6;
                /* Allow only exact matches, suppress specifics */
                prefix-list-filter pl-allowed-${peer_as}-v6 exact;
            }
            then next policy;
        }
        term reject-the-rest {
            then reject;
        }
    }
}
```

The template in Table 64 defines an import policy that enables destination black-hole route signaling along with verification of the prefixes upon import (including black-hole routes).

Table 64: DIA Attachment Circuit Routing—BGP Routing with Route Import Verification

Template ID	dia-bgp-import-policy-3	**Device Role**	PE router, DCR
Service	DIA	**Guidelines**	Import of a select set of customer routes plus allow destination black-hole signaling
Dependencies	dia-policy		
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Peer AS number		
allowed_v4_prefix	Array of IPv4 prefixes allowed from that peer		
allowed_v6_prefix	Array of IPv6 prefixes allowed from that peer		
gtsm	Set to “gtsm-” for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		
group_suffix	If the peer type is customer, set to “-full-view” for neighbors receiving full view BGP routing table, set to “-default-only” for neighbors receiving default only route, if the peer type is upstream or peer, leave blank		
peer_type	Set to “customer” if the peer is of a customer type, set to “peer” if the peer type is a peer, set to “upstream” if the peer type is an upstream		

```
protocols {
    bgp {
        group v4-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v4} {
                /*  In Profile #3 black hole route installation is enabled,
                    pol-prefix-allow-${peer_as}-bh is used. We also accept next-hops other than
                    the peer address (this is to enable next-hop change to 192.0.2.1) */
                accept-remote-nexthop;
                import [ pol-prefix-allow-${peer_as}-bh pol-import-${peer_type}-default
                pol-final ];
            }
        }
        group v6-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v6} {
                /*  In Profile #3 black hole route installation is enabled,
                    pol-prefix-allow-${peer_as}-bh is used. We also accept next-hops other than
                    the peer address (this is to enable next-hop change to ::ffff:192.0.2.1) */
                accept-remote-nexthop;
                import [ pol-prefix-allow-${peer_as}-bh pol-import-${peer_type}-default
                pol-final ];
            }
        }
    }
}

policy-options {
    /* Prefix list containing all V4 prefixes allowed to accept from the peer AS */
    prefix-list pl-allowed-${peer_as}-v4 {
        ${allowed_v4_prefix[i]};
    }
    /* Prefix list containing all V6 prefixes allowed to accept from the peer AS */
    prefix-list pl-allowed-${peer_as}-v6 {
        ${allowed_v6_prefix[i]};
    }
    /*  This policy is customer specific. It allows only certain routes from the customer
        It also allows routes with a black hole community */
    policy-statement pol-prefix-allow-${peer_as}-bh {
```

```
        term allow-v4-blackhole {
            from {
                family inet;
                /* Accept longer prefixes with the black hole community
                   All allowed prefixes are listed here.*/
                prefix-list-filter pl-allowed-${peer_as}-v4 orlonger;
                community comm-remote-triggered-black-hole;            }
           then {
                /* Set the next-hop address that is resolved through the discard route */
                next-hop 192.0.2.1;
                /* Attach no-export community */
                community add no-export;
                next policy;
            }
        }
        term allow-v6-blackhole {
            from {
                family inet6;
                /* Accept longer prefixes with the black hole community
                   All allowed prefixes are listed here.*/
                prefix-list-filter pl-allowed-${peer_as}-v6 orlonger;
                community comm-remote-triggered-black-hole;
            }
            then {
                 /* Set the next-hop address that is resolved through the discard route */
                next-hop ::ffff:192.0.2.1;
                /* Attach no-export community */
                community add no-export;
                next policy;
            }
        }
        term allow-v4 {
            from {
                family inet;
                /* All allowed prefixes are listed here, suppress specifics. */
                prefix-list-filter pl-allowed-${peer_as}-v4 exact;
            }
            then next policy;
        }
        term allow-v6 {
            from {
                family inet6;
                /* All allowed prefixes are listed here, suppress specifics. */
                prefix-list-filter pl-allowed-${peer_as}-v6 exact;
            }
            then next policy;
        }
        term reject-the-rest {
            then reject;
        }
    }
}
```

The template in Table 65 defines an import policy that enables source black hole route installation. This policy should normally be enabled for trusted peers only. Source black hole route installation is sensitive because source black hole routing takes effect for each carrier subscriber; even those subscribers not affected by the attack. Because the peer is trusted, there is no need to limit route import to certain prefixes. The rest of the configuration remains similar to the previous policies.

Table 65: DIA Attachment Circuit Routing—BGP Routing Policy with Source Black-Hole Route Installation

Template ID	dia-bgp-import-policy-4	Device Role	PE router, DCR
Service	DIA	Guidelines	Import of a select set of customer routes plus allow source black-hole signaling
Parameters			
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Peer AS number		
gtsm	Set to “gtsm-” for GTSM-enabled peers (this is to construct loopback filter automatically), otherwise leave blank		
group_suffix	If the peer type is customer, set to “-full-view” for neighbors receiving full view BGP routing table, set to “-default-only” for neighbors receiving default only route; if the peer type is upstream or peer, leave blank		
peer_type	Set to “customer” if the peer is of a customer type, set to “peer” if the peer type is a peer, set to “upstream” if the peer type is an upstream		
peer_type	Set to “customer” if the peer is of a customer type, set to “peer” if the peer type is a peer, set to “upstream” if the peer type is an upstream		

```
protocols {
    bgp {
        /* Neighbors in this group receive all Internet IPv4 routes */
        group v4-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v4} {
                /*  In Profile #4 source black hole route installation is enabled,
                    pol-prefix-allow-source-bh is used. We also accept next-hops other than
                    the peer address (this is to enable next-hop change to 192.0.2.1). Minimal
                    sanity checks are done as well (pol-reject-martians) */
                accept-remote-nexthop;
                import [ pol-reject-martians pol-prefix-allow-source-bh
                         pol-import-${peer_type}-default pol-final ];
            }
        }
        /* Neighbors in this group receive all Internet IPv6 routes */
        group v6-${gtsm}${peer_type}${group_suffix} {
            neighbor ${ce_v6} {
                /* In Profile #4 source black hole route installation is enabled,
                   pol-prefix-allow-source-bh is used. We also accept next-hops other than the
                   peer address (this is to enable next-hop change to 192.0.2.1).
                   Minimal sanity checks are done as well (pol-reject-martians) */
                accept-remote-nexthop;
                import [ pol-reject-martians pol-prefix-allow-source-bh
                         pol-import-${peer_type}-default pol-final ];
                peer-as ${peer_as};
            }
        }
    }
}

policy-options {

    /*  This policy allows all routes from the customer and enables
        source remote triggered black hole installation */
    policy-statement pol-prefix-allow-source-bh {
        /* Set next-hop for family inet black hole routes */
        term allow-source-blackhole-inet {
            from {
                family inet;
                /* Sources can be anywhere on Internet, no prefix checks are made */
                community comm-remote-triggered-black-hole;
            }
            then {
```

```
                /* Set the next-hop address that is resolved through the discard route */
                next-hop 192.0.2.1;
                /* Attach no-export community */
                community add comm-no-export;
                next policy;
            }
        }
        /* Set next-hop for family inet6 black hole routes */
        term allow-source-blackhole-inet6 {
            from {
                family inet6;
                /* Sources can be anywhere on Internet, no prefix checks are made */
                community comm-remote-triggered-black-hole;
            }
            then {
                /* Set the next-hop address that is resolved through the discard route */
                next-hop ::ffff:192.0.2.1;
                /* Attach no-export community */
                community add comm-no-export;
                next policy;
            }
        }
    }
}
```

Layer 3 VPN Service-Specific Configuration

Layer 3 VPN service attachment circuit provisioning at the router requires the following two steps:

1. Initial configuration of the Layer 3 VPN routing instance (run only once).
2. Actual configuration of the attachment circuit, including routing protocol and other parameters (BFD detection).

Routing instance configuration depends on the VPN topology. The next two sections describe routing instance configuration for any-to-any topology and for hub-and-spoke topology.

Routing Instance—Any-to-Any VPN

The template in Table 66 is used to configure a Layer 3 VPN routing instance. The route distinguisher identifier and a VRF target must be configured to provision a VPN instance. Route distinguishers are allocated on a per-VPN basis or per-VPN and per-PE device basis. In this solution, the latter option is used. The easiest way to achieve this configuration is to use the PE router loopback address as a part of a route distinguisher and the locally unique VPN identifier (some operators might choose to use globally unique identifiers to simplify operations).

Table 66: Layer 3 VPN Service Configuration—Any-to-Any VPN

Template ID	l3vpn-instance	**Device Role**	PE device, DCR
Service	Layer 3 VPN	**Guidelines**	Use to provision any-to-any instance
Constants			
Each instance supports up to 10,000 IPv6 prefixes and 100,000 IPv4 prefixes. Once the 75 percent threshold limit is reached, a message is logged.			
Parameters			
vrf_target	VRF route target		
vrf_id	An integer number identifying this particular VRF, which is used to produce a route distinguisher identifier		
ipv4_loopback	IPv4 loopback address, used to create a route distinguisher identifier		
cust_name	Customer name, used to produce VRF name identifiers		

```
groups {
    l3vpn-instance-settings {
        routing-instances {
            <*> {
                routing-options {
                    rib <*.inet6.0> {
```

```
                        maximum-prefixes 10000 threshold 75;
                    }
                    multipath {
                        vpn-unequal-cost;
                    }
                    maximum-prefixes 100000 threshold 75;
                }
            }
        }
    }
}
routing-instances {
    /* l3vpn- prefix is used in the routing instance name to selectively apply l3vpn groups to
       l3vpn any-to-any service if needed later */
    l3vpn-${cust_name} {
        apply-groups [ l3vpn-instance-settings ];
        instance-type vrf;
        vrf-target ${vrf_target};
        route-distinguisher ${ipv4_loopback}:${vrf_id};
        routing-options {
            /* Need to explicitly define inet6 in order for the prefix limits from the
               l3vpn-instance-settings to apply */
            rib l3vpn-${cust_name}.inet6.0;
        }
        vrf-table-label;
    }
}
```

In the template in Table 66, the VRF route import and export rules are configured automatically by using vrf-target construct.

Per-VRF label allocation is enabled. In addition, per-VRF route limits are enforced (10,000 IPv4 routes and 100,000 IPv6 routes).

The **cust_name** parameter can match the **vrf_id** value. However, to simplify troubleshooting, you can use a short name to identify the customer.

Routing Instance—Hub-and-Spoke VPN

Hub-and-spoke VPN configuration adheres to the high-level design of the solution. The template in Table 67 is used at the hub site.

Table 67: Layer 3 VPN Service Configuration – Hub-and-Spoke VPN

Template ID	l3vpn-hs-instance-hub	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Use to provision hub-and-spoke instance at the hub site
Constants			
Each instance supports up to 10,000 IPv6 prefixes and 100,000 IPv4 prefixes. Once the 75 percent threshold limit is reached, a message is logged.			
Parameters			
vrf_target_hub	VRF route target for hub routes		
vrf_target_spoke	VRF route target for spoke routes		
vrf_id	An integer number identifying this particular VRF, which is used to produce a route distinguisher identifier		
ipv4_loopback	IPv4 loopback address, used to create a route distinguisher identifier		
cust_name	Customer name, used to produce VRF name identifiers		

```
groups {
    l3vpn-instance-settings {
        routing-instances {
            <*> {
                routing-options {
```

```
                    rib <*.inet6.0> {
                        maximum-prefixes 10000 threshold 75;
                    }
                    maximum-prefixes 100000 threshold 75;
                }
            }
        }
    }
}
routing-instances {
    /* l3vpn_hub- prefix is used in the routing instance name to selectively apply l3vpn groups
       to l3vpn hub instances if needed */
    /* This routing instance should include all hub interfaces. It does import spoke prefixes,
       hub prefixes and exports only hub prefixes. Note that auto-export is enabled. This is to
       account for a case when same PE router has both spoke interfaces and hub interfaces */
    l3vpn_hub-${cust_name} {
        apply-groups [ l3vpn-instance-settings ];
        instance-type vrf;
        vrf-target export ${vrf_target_hub};
        vrf-import pol-hub-import-${cust_name};
        route-distinguisher ${ipv4_loopback}:${vrf_id};
        routing-options {
            /* Need to explicitly define inet6 in order for the prefix limits from the
               l3vpn-instance-settings to apply */
            rib l3vpn_hub-spoke-routes-${cust_name}.inet6.0;
            auto-export;
        }
        /* Note : VRF table label is not configured */
    }
}

policy-options {
    /* Community definitions for hub & spoke routes */
    community comm-hub-${cust_name} members ${vrf_target_hub};
    community comm-spoke-${cust_name} members ${vrf_target_spoke};

    /* Hub instance route import policy */
    policy-statement pol-hub-import-${cust_name} {
        term accept-hub-and-spoke-routes {
            from {
                community [ comm-hub-${cust_name} comm-spoke-${cust_name} ];
            }
            then accept;
        }
        term reject-the-rest {
            then reject;
        }
    }
}
```

Unlike any-to-any VPN configuration, route import and export rules are asymmetric at the hub-and-spoke site of the hub-and-spoke VPN. In this configuration, hub routes are exported to the rest of the network with a community that identifies the hub site (**vrf_target_hub**). On the other hand, all hub and all spoke routes are accepted at the hub, whereas the hub site can reach all destinations. In order to avoid direct spoke-to-spoke communication bypassing the CE device, this VPN instance does not have the **vrf-table-label** statement configured. As a result, the router defaults to per next-hop label allocation. In this mode, IP address lookup is not made for the packets coming from the core. Packets are transmitted to the CE router instead, preventing direct spoke-to-spoke communication.

NOTE: The usage of the **auto-export** keyword is used when configuring the VRF routing options. There could be cases when a PE device has attachment circuits from hub sites and from spoke sites. In those cases, the PE device must have a different set of VRFs configured. The auto-export option ensures that routes are exported from the hub VRF to the spoke VRF on the same router.

The template in Table 68 represents a hub-and-spoke VPN configuration at the spoke site.

Table 68: Layer 3 VPN Service Configuration—Hub-and-Spoke Configuration at Spoke Site

Template ID	l3vpn-hs-instance-spoke	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Use to provision hub-and-spoke instance at the spoke site
Constants			
Each instance supports up to 10,000 IPv6 prefixes and 100,000 IPv4 prefixes. Once the 75 percent threshold limit is reached, a message is logged.			
Parameters			
vrf_target_hub	VRF route target for hub routes		
vrf_target_spoke	VRF route target for spoke routes		
hub_routes_vrf_id	An integer number identifying hub routes VRF, which is used to produce a route distinguisher identifier		
spoke_routes_vrf_id	An integer number identifying hub routes VRF, which is used to produce a route distinguisher identifier		
ipv4_loopback	IPv4 loopback address, used to create a route distinguisher identifier		
cust_name	Customer name, used to produce VRF name identifiers		

```
groups {
    l3vpn-instance-settings {
        routing-instances {
            <*> {
                routing-options {
                    rib <*.inet6.0> {
                        maximum-prefixes 10000 threshold 75;
                    }
                    maximum-prefixes 100000 threshold 75;
                }
            }
        }
    }
}
routing-instances {
    /* This routing instance should not have any interfaces configured, traffic to this instance
       is delivered from the spoke_routes vpn via 0/0 route or an output interface filer.
       Only routes from hub sites are imported into this instance */
    l3vpn_spoke-hub_routes-${cust_name} {
        apply-groups [ l3vpn-instance-settings ];
        instance-type vrf;
        vrf-target ${vrf_target_hub};
        route-distinguisher ${ipv4_loopback}:${hub_routes_vrf_id};
        routing-options {
            /* Need to explicitly include this in order for the l3vpn-max-routes group to
               apply prefix limits */
            rib l3vpn_spoke-hub_routes-${cust_name}.inet6.0;
        }
        /* Note : VRF table label is not configured */
    }
    /* l3vpn_spoke- prefix is used in the routing instance name to selectively apply l3vpn
       groups to l3vpn spoke instances if needed */

    /* This routing instance should include all spoke interfaces, it does not import any routes,
       however it does export routes learned from spoke sites. Note that auto-export is enabled.
       This is to account for a case when same PE router has both spoke interfaces
       and hub interfaces */
    l3vpn_spoke-spoke_routes-${cust_name} {
        apply-groups [ l3vpn-instance-settings ];
        instance-type vrf;
        vrf-target export ${vrf_target_spoke};
```

```
        /* All routes from other sites are rejected, traffic should be forwarded only through
           the corresponding hub_routes instance */
        vrf-import pol-reject-all;
        route-distinguisher ${ipv4_loopback}:${spoke_routes_vrf_id};
        routing-options {
            rib l3vpn_spoke-spoke_routes-${cust_name}.inet6.0 {
                static {
                    /* Route all traffic through the hub, spoke to spoke traffic is redirected
                    via an output filter at the interface */
                    route ::/0 next-table l3vpn_spoke-hub_routes-${cust_name}.inet6.0;
                }
            }
            static {
                /* Route all traffic through the hub, spoke to spoke traffic is redirected via
                   an output filter at the interface */
                route 0.0.0.0/0 next-table l3vpn_spoke-hub_routes-${cust_name}.inet.0;
            }
            auto-export;
        }
        vrf-table-label;
    }
}

policy-options {
    policy-statement pol-reject-all {
        then reject;
    }
}
```

In this template, two VRFs are configured. VRF **l3vpn_spoke-spoke_routes** stores routes from the spoke locations (customer sites). Another VRF (**l3vpn_hub-hub_routes**) stores routes from the remote hub locations. Spoke interfaces are added to the spoke VRF (this is a part of the attachment circuit configuration template defined later). The hub VRF does not contain any interfaces. Traffic to that VRF is forwarded through either a default route from the spoke VRF or through a filter (defined later). The spoke VRF only stores local routes received from the local peers. Those routes are exported to the rest of the network with the spoke VRF community (therefore they are imported at the hub sites). The spoke VRF does not import any routes, because all forwarding is supposed to be done through a hub VRF site (using a default route).

The hub VRF imports hub VRF routes but it does not export any routes, because no routes are originated from that VRF (it should not have any attachment circuits, all attachment circuits are configured under spoke VRF).

In this design, the packet forwarding path from the customer spoke site occurs as follows:

- The packet is received in the spoke VRF context.
- A route lookup takes place. If the packet is destined for the hub or a remote spoke site, it should hit the default route. Otherwise, the packet is sent to the local interface. If sent to the local interface, it is caught by an output filter (part of the attachment circuit template) and still delivered to the hub VRF route.
- The route lookup occurs in the hub VRF context and the packet is sent to the hub.

The following path is taken in the opposite direction:

- The packet destined for the spoke site is received in the spoke VRF context (spoke routes are exported from this VRF).
- The packet is sent to its ultimate destination (the local site). The packet is not forwarded to the hub VRF site by the output filter; only traffic from the local sites is forwarded (see filter definition in the attachment circuit template).

Attachment Circuit Configuration

No special configuration occurs for an attachment circuit belonging to the any-to-any Layer 3 VPN or an attachment circuit at the hub sites of hub-and-spoke VPNs. However, the spoke site attachment circuit has additional configuration that prevents spoke-to-spoke communication in the event multiple spokes are terminated at the same PE router.

The template in Table 69 shows this additional configuration.

Table 69: Layer 3 VPN Service Configuration—Spoke Site Attachment Circuit Configuration

Template ID	l3vpn-hs-attachment-circuit-spoke-site	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Layer 3 VPN spoke site in hub-and-spoke Layer 3 VPN topology
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		

```
interfaces {
    ${cust_ifd} {
        /* Customer IFL configuration */
        unit ${cust_unit} {
            apply-groups l3vpn-ifl-settings;
            /* Note that rpf check is enabled for the Layer 3 VPN interfaces too
               that might not be required in some cases if this function is done at the CPE */
            family inet {
                rpf-check;
                address ${router_ipv4};
                filter {
                    /* All traffic from local spoke sites is redirected to the hub VRF.
                       All other traffic (from hub sites) is accepted.
                       Replace sec-accept-all-v4 with CoS classification filter if needed */
                    output-list [ sec-via-hub-${cust_name}-v4 sec-accept-all-v4];
                    /* All spoke interfaces belong to group 4, this is used as a match
                      condition in an output filter to identify traffic from local spoke sites */
                    group 4;
                }
            }
            family inet6 {
                rpf-check;
                address ${router_ipv6};
                filter {
                    /* All traffic from local spoke sites is redirected to the hub VRF
                       All other traffic (from hub sites) is accepted.
                       Replace sec-accept-all-v6 with CoS classification filter if needed */
                    output-list [ sec-via-hub-${cust_name}-v6 sec-accept-all-v6];
                    /* All spoke interfaces belong to group 4, this is used as a match
                      condition in an output filter to identify traffic from local spoke sites */
                    group 4;
                }
            }
        }
    }
}

firewall {
    family inet {
        filter sec-via-hub-${cust_name}-v4 {
            /* Select traffic from local spoke sites and redirect it to the hub VRF */
            term from-spokes {
                from {
                    interface-group 4;
                }
                then {
```

```
                    routing-instance l3vpn_spoke-hub_routes-${cust_name};
                }
            }
        }
        /* This filter is provided for illustration purposes, it should typically be replaced
            by the egress classification filter */
       filter sec-accept-all-v4 {
            term all-traffic {
                then {
                    accept;
                }
            }
        }
    }
    family inet6 {
        filter sec-via-hub-${cust_name}-v6 {
            /* Select traffic from local spoke sites and redirect it to the hub VRF */
            term from-spokes {
                from {
                    interface-group 4;
                }
                then {
                    routing-instance l3vpn_spoke-hub_routes-${cust_name};
                }
            }
        }
        /* This filter is provided for illustration purposes, it should typically be replaced
           by the egress classification filter */
        filter sec-accept-all-v6 {
            term all-traffic {
                then {
                    accept;
                }
            }
        }
    }
}
```

All Layer 3 VPN spoke interfaces are assigned interface group 4. An output filter is configured at each interface. This filter forwards traffic to the hub VRF in the event traffic is coming from a spoke interface, blocking spoke-to-spoke communication.

Attachment Circuit Routing—Static

The template in Table 70 defines a static routing configuration. The template is used to provision IPv4 and IPv6 static routes pointing to the CE device. This template can be used for both any-to-any Layer 3 VPN and hub-and-spoke Layer 3 VPNs.

Table 70: Layer 3 VPN Service Configuration—Static Routing Configuration

Template ID	l3vpn-static	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Static routing
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
route_v4	An array of IPv4 static routes (multiple routes possible)		
route_v6	An array of IPv6 static routes (multiple routes possible)		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        interface ${cust_ifd}.${cust_unit};
        routing-options {
            rib l3vpn${suffix}-${cust_name}.inet6.0 {
                static {
                    /* The following config is repeated for every static route I */
                    route ${route_v6[i]} {
                        next-hop ${ce_v6};
                    }
                }
            }
            static {
                /* The following config is repeated for every static route I */
                route ${route_v4[i]} {
                    next-hop ${ce_v4};
                }
            }
        }
    }
}
```

The template in Table 71 should be used to enable BFD for the static routes defined above.

Table 71: Layer 3 VPN Service Configuration—BFD for Static Routing Configuration

Template ID	l3vpn-static-bfd	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Static routing
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
route_v4	An array of IPv4 static routes (multiple routes possible)		
route_v6	An array of IPv6 static routes (multiple routes possible)		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        routing-options {
            rib l3vpn${suffix}-${cust_name}.inet6.0 {
                static {
                    /* The following config is repeated for every static route I */
                    route ${route_v6[i]} {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
            }
            static {
                /* The following config is repeated for every static route I */
                route ${route_v4[i]} {
                    bfd-liveness-detection {
```

```
                        minimum-interval ${bfd_min_interval};
                        multiplier ${bfd_multiplier};
                    }
                }
            }
        }
    }
}
```

Attachment Circuit Routing—BGP Routing

The template in Table 72 defines BGP routing configuration between PE and CE devices. This template can be used for both any-to-any Layer 3 VPN and hub-and-spoke Layer 3 VPNs. Two BGP sessions are configured for each peer, one is IPv4 and another one IPv6.

Table 72: Layer 3 VPN Service Configuration—BGP Routing Configuration

Template ID	l3vpn-bgp	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	BGP routing
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
peer_as	Customer autonomous system		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        interface ${cust_ifd}.${cust_ifl};
        protocols {
            bgp {
                path-selection external-router-id;
                group v4-peers {
                    type external;
                    neighbor ${ce_v4} {
                        family inet {
                            unicast;
                        }
                        peer-as ${peer_as};
                    }
                }
                group v6-peers {
                    type external;
                    neighbor ${ce_v6} {
                        family inet6 {
                            unicast;
                        }
                        peer-as ${peer_as};
                    }
                }
            }
        }
    }
}
```

The template in Table 73 enables BFD on the BGP sessions.

Table 73: Layer 3 VPN Service Configuration—BFD for BGP Routing Configuration

Template ID	l3vpn-bgp-bfd	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	BGP routing
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        protocols {
            bgp {
                group v4-peers {
                    neighbor ${ce_v4} {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
                group v6-peers {
                    neighbor ${ce_v6} {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
            }
        }
    }
}
```

In some scenarios, the service provider might need to enable the AS number override for the routes received from the Layer 3 VPN BGP peer. This is required when Layer 3 VPN is provisioned between sites that belong to the same autonomous system.

The template in Table 74 defines the AS override configuration.

Table 74: Layer 3 VPN Service Configuration – BGP Routing Configuration with AS Override

Template ID	l3vpn-bgp-as-override	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	BGP routing
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
ce_v4	CE router IPv4 address		
ce_v6	CE router IPv6 address		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        protocols {
            bgp {
                group v4-peers {
                    type external;
                    neighbor ${ce_v4} {
                        as-override;
                    }
                }
                group v6-peers {
                    type external;
                    neighbor ${ce_v6} {
                        as-override;
                    }
                }
            }
        }
    }
}
```

Attachment Circuit Routing–OSPF Routing

The template in Table 75 defines OSPF routing configuration between the PE and CE devices. You can use this template for both any-to-any Layer 3 VPNs and hub-and-spoke Layer 3 VPNs.

Table 75: Layer 3 VPN Service Configuration – OSPF Routing Configuration

Template ID	l3vpn-ospf	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	OSPF and OSPF3 routing
Constants			
OSPF Area is set to 0.0.0.0			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        interface ${cust_ifd}.${cust_unit};
        protocols {
            ospf {
                area 0.0.0.0 {
                    interface ${cust_ifd}.${cust_unit};
                }
            }
            ospf3 {
                area 0.0.0.0 {
                    interface ${cust_ifd}.${cust_unit};
                }
            }
        }
    }
}
```

The template in Table 76 enables BFD for OSPF.

Table 76: Layer 3 VPN Service Configuration—BFD for OSPF Routing Configuration

Template ID	l3vpn-ospf-bfd	Device Role	PE router, DCR
Service	Layer 3 VPN	Guidelines	OSPF and OSPF3 routing
Constants			
OSPF Area is set to 0.0.0.0			
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        protocols {
            ospf {
                area 0.0.0.0 {
                    interface ${cust_ifd}.${cust_unit} {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
            }
            ospf3 {
                area 0.0.0.0 {
                    interface ${cust_ifd}.${cust_unit} {
                        bfd-liveness-detection {
                            minimum-interval ${bfd_min_interval};
                            multiplier ${bfd_multiplier};
                        }
                    }
                }
            }
        }
    }
}
```

Attachment Circuit Routing—RIP Routing

The template in Table 77 defines RIP routing configuration between the PE and CE devices. You can use this template for both any-to-any Layer 3 VPNs and hub-and-spoke Layer 3 VPNs.

Table 77: Layer 3 VPN Service Configuration—RIP Routing Configuration

Template ID	l3vpn-rip	Device Role	PE router, DCR
Service	Layer 3 VPN	Guidelines	RIP and RIPng routing
Parameters			
cust_ifd	Physical interface where attachment circuit is terminated		
cust_unit	Logical interface where attachment circuit is terminated		
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        interface ${cust_ifd}.${cust_unit};
        protocols {
            rip {
                group rip-to-ce {
                    export pol-vpn-export-to-rip;
                    interface ${cust_ifd}.${cust_unit};
                }
            }
            ripng {
                group ripng-to-ce {
                    export pol-export-to-rip;
                    interface ${cust_ifd}.${cust_unit};
                }
            }
        }
    }
}

policy-options {
    policy-statement pol-vpn-export-to-rip {
        term accept-statics-bgp-rip {
            from protocol [ bgp rip ripng static ospf ospf3 ];
            then accept;
        }
        term reject-others {
            then reject;
        }
    }
}
```

All static, direct, BGP, OSPF and OSPFv3, RIP, and RIPng routes are exported to RIP/RIPng neighbors.

BFD is enabled for RIPv2 using the template in Table 78.

NOTE: BFD is not currently supported for RIPng neighbors.

Table 78: Layer 3 VPN Service Configuration—BFD for RIPv2 Routing Configuration

Template ID	l3vpn-rip-bfd	Device Role	PE router, DCR
Service	Layer 3 VPN	Guidelines	RIP and RIPng routing
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
suffix	Part of the routing instance name, which is set to "_spoke-spoke-routes" for Layer 3 VPN spoke sites; set to "_hub" for Layer 3 VPN hub sites; and left blank for any-to-any Layer 3 VPNs		
bfd_min_interval	Minimum BFD interval for the BFD session		
bfd_multiplier	BFD multiplier for the BFD session		

```
routing-instances {
    l3vpn${suffix}-${cust_name} {
        protocols {
            rip {
                group rip-to-ce {
                    bfd-liveness-detection {
                        minimum-interval ${bfd_min_interval};
                        multiplier ${bfd_multiplier};
                    }
                }
            }
        }
    }
}
```

Multicast Layer 3 VPN Service Specific Configuration

Receiver Site Configuration

The template in Table 79 is applied at the receiver site of the MVPN. It enables PIM protocol support on all VRF interfaces, sets PIM mode to sparse, and configures version 2. The **mvpn** statement enables support for BGP signaling and point-to-multipoint (P2MP) transport.

Table 79: Multicast Layer 3 VPN Service—Receiver Site Configuration

Template ID	mvpn-receiver	Device Role	PE router, DCR
Service	MVPN	Guidelines	Receiver site
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		

```
routing-instances {
    l3vpn-${cust_name} {
        protocols {
            pim {
                interface all {
                    mode sparse;
                    version 2;
                }
            }
            mvpn;
        }
    }
}
```

The template in Table 80 should be applied in addition to the previous one if PIM any source multicast is required. The template configures rendezvous point IPv4/IPv6 addresses, which are required for any source multicast operation.

Table 80: Multicast Layer 3 VPN Service—Receiver Site Configuration PIM Any Source Multicast

Template ID	mvpn-receiver-sm	Device Role	PE router, DCR
Service	MVPN	Guidelines	Receiver site, any source multicast
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
rp_v4	Rendezvous point IPv4 address		
rp_v6	Rendezvous point IPv6 address		

```
routing-instances {
    l3vpn-${cust_name} {
        protocols {
            pim {
                rp {
                    static {
                        address ${rp_v4} {
                            version 2;
                        }
                        address ${rp_v6} {
                            version 2;
                        }
                    }
                }
            }
        }
    }
}
```

The template in Table 81 enables two virtual tunnel (VT) interfaces, one is declared primary and the other one is backup. We recommend distributing those tunnel interfaces between two different line cards, for example, uplink cards.

Table 81: Multicast Layer 3 VPN Service—Receiver Site Configuration with Multiple Virtual Tunnel Interfaces

Template ID	mvpn-receiver-vt	Device Role	PE router, DCR
Service	MVPN	Guidelines	Use in the P/PE device to avoid traffic duplication at uplinks
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
vt_primary_fpc	FPC of the primary virtual tunnel interface		
vt_primary_pic	PIC of the primary virtual tunnel interface		
vt_primary_port	Port of the primary virtual tunnel interface		
vt_primary_unit	Unit number of the primary virtual tunnel interface		
vt_secondary_fpc	FPC of the secondary virtual tunnel interface		
vt_secondary_pic	PIC of the secondary virtual tunnel interface		
vt_secondary_port	Port of the secondary virtual tunnel interface		
vt_secondary_unit	Unit number of the secondary virtual tunnel interface		

```
chassis {
    fpc ${vt_primary_fpc} {
        pic ${vt_primary_pic} {
            tunnel-services {
                bandwidth ${tunnel_bw};
            }
        }
    }
    fpc ${vt_backup_fpc} {
        pic ${vt_backup_pic} {
            tunnel-services {
                bandwidth ${tunnel_bw};
            }
        }
    }
}

interfaces {
    vt-${vt_primary_fpc}/${vt_primary_pic}/${vt_primary_port} {
        unit ${vt_primary_unit} {
            family inet;
            family inet6;
        }
    }
    vt-${vt_backup_fpc}/${vt_backup_pic}/${vt_primary_port} {
        unit ${vt_primary_unit} {
            family inet;
            family inet6;
        }
    }
}
routing-instances {
    /* l3vpn- prefix is used in the routing instance name to selectively apply l3vpn groups to
       l3vpn any-to-any service if needed later */
    l3vpn-${cust_name} {
        interface vt-${vt_primary_fpc}/${vt_primary_pic}/${vt_primary_port}.${vt_primary_unit} {
            multicast;
            primary;
        }
        interface vt-${vt_backup_fpc}/${vt_backup_pic}/${vt_backup_port}.${vt_backup_unit} {
            multicast;
        }
    }
}
```

This template must only be applied if the PE router functions as a P router (for example, in square redundant PE router topologies).

NOTE: Multicast traffic duplication at the PE router uplink should be avoided.

Sender Site Configuration

The template in Table 82 defines MVPN configuration at the sender site. This template is used to enable multicast distribution in aggregate trees, when multicast traffic to all PIM groups share the same tree. In addition to the PIM configuration and MVPN signaling, this template also defines the RSVP-TE signaled point-to-multipoint LSP template, which is used to instantiate LSPs from the sender site to the receivers.

NOTE: Link protection is requested for the LSP and it is also signaled with a preconfigured bandwidth.

Table 82: Multicast Layer 3 VPN Service—Sender Site Configuration

Template ID	mvpn-sender	**Device Role**	PE router, DCR
Service	MVPN	**Guidelines**	Aggregate tree distribution
Constants			
P2MP LSP optimize bandwidth is set to 50.			
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
mcast_bw	Expected multicast stream bandwidth (for reservation)		

```
protocols {
    mpls {
        label-switched-path p2mp-template-mvpn-${mcast_bw}  {
            template;
            bandwidth ${mcast_bw};
            optimize-timer 50;
            link-protection;
            p2mp;
        }
    }
}

routing-instances {
    l3vpn-${cust_name} {
        provider-tunnel {
            rsvp-te {
                label-switched-path-template {
                    p2mp-template-mvpn-${mcast_bw};
                }
            }
        }
        protocols {
            pim {
                interface all {
                    mode sparse;
                    version 2;
                }
            }
            mvpn;
        }
    }
}
```

Unlike the template in Table 82, the template in Table 83 enables traffic distribution in selective trees. Each multicast group can be mapped to a selective tree. This approach is more optimal in cases where receivers are only subscribed to a subset of multicast groups, and not subscribed to all of them.

Table 83: Multicast Layer 3 VPN Service—Sender Site Configuration with Selective Tree Mapping

Template ID	mvpn-sender-selective-tree	**Device Role**	PE router, DCR
Service	MVPN	**Guidelines**	Selective tree distribution
Constants			
P2MP LSP optimize bandwidth is set to 50.			
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
mcast_bw	Expected multicast stream bandwidth (for reservation)		
group	Multicast group IPv4 or IPv6 address		
source	Multicast source IPv4 or IPv6 address		

```
protocols {
    mpls {
        label-switched-path p2mp-template-mvpn-${mcast_bw} {
            template;
            bandwidth ${mcast_bw};
            optimize-timer 50;
            link-protection;
            p2mp;
        }
    }
}

routing-instances {
    l3vpn-${cust_name} {
        provider-tunnel {
            group ${group} {
                source ${source} {
                    rsvp-te {
                        label-switched-path-template {
                            p2mp-template-mvpn-${mcast_bw};
                        }
                    }
                }
            }
        }
        protocols {
            pim {
                interface all {
                    mode sparse;
                    version 2;
                }
            }
            mvpn;
        }
    }
}
```

The carrier can choose to support rendezvous point functionality in the context of the customer VPN. The best location for the rendezvous point (RP) configuration is the sender site. In this case, the rendezvous point function shares its fate with the multicast traffic.

To provide RP redundancy, we assume that other sender sites exist and that the RP is configured at these other sender sites as well. Those rendezvous points form an anycast RP set.

The template in Table 84 provides local RP configuration at the sender site and enables redundant RP configuration by specifying a peer RP. RP configuration requires a loopback interface in the routing instance context; this loopback is enabled as well.

Table 84: Multicast Layer 3 VPN Service—Sender Site Configuration with Local Rendezvous Point Configuration

Template ID	mvpn-sender-local-rp	**Device Role**	PE router, DCR
Service	MVPN	**Guidelines**	Aggregate tree distribution
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
lo_unit	VPN loopback unit number		
vpn_lo_v4	VPN loopback IPv4 address		
vpn_lo_v6	VPN loopback IPv6 address		
anycast_rp_v4	Anycast IPv4 address		
anycast_rp_v6	Anycast IPv6 address		
peer_rp_v4	Peer RP IPv4 address		
peer_rp_v6	Peer RP IPv6 address		

```
interfaces {
    lo0 {
        unit ${lo_unit} {
            family inet {
                address ${vpn_lo_v4} {
                    primary;
                };
                address ${anycast_rp_v4};
            }
            family inet6 {
                address ${vpn_lo_v6} {
                    primary;
                };
                address ${anycast_rp_v6};
            }
        }
    }
}
routing-instances {
    l3vpn-${cust_name} {
        interface lo0.${lo_unit};
        protocols {
            pim {
                rp {
                    local {
                        family inet {
                            address ${anycast_rp_v4};
                            anycast-pim {
                                rp-set {
                                    ${peer_rp_v4};
                                }
                                local-address ${vpn_lo_v4};
                            }
                        }
                        family inet6 {
                            address ${anycast_rp_v6};
                            anycast-pim {
                                rp-set {
                                    ${peer_rp_v6};
```

```
                                }
                                local-address ${vpn_lo_v6};
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Extranet Receiver Configuration

In an MVPN extranet configuration, sender sites belong to one sender VPN, whereas receiver sites belong to other VPNs. Receivers should be able to join sources in the single VPN. The template in Table 85 defines VPN configuration where receiver sites are located.

Table 85: Multicast Layer 3 VPN Service—Extranet Receiver Configuration

Template ID	mvpn-extranet-receiver	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Use to configure VPN with extranet multicast receivers
Constants			
P2MP LSP optimize bandwidth is set to 50.			
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
sender_target	VRF target for importing MVPN routes		
receiver_target	Route target of the receiver		

```
routing-instances {
    l3vpn-${cust_name} {
        /* Enable auto-export to support receiver / sender VRFs on the same PE */
        routing-options {
            auto-export;
        }
        protocols {
            mvpn {
                route-target {
                    import-target {
                        target ${sender_target};
                    }
                    export-target {
                        target ${receiver_target};
                    }
                }
            }
        }
    }
}
```

NOTE: All routes with their VPN route target applied to the sender VPN routes are imported.

The route target for MVPN C-multicast routes are announced with a special receiver target community. This community is not equal to the regular target community of this VPN. It is assumed that this community is used for all receiver sites of all receiver VPNs.

The sender target community matches the regular VRF target community of the sender VPN; there is no special community allocated for this purpose.

The auto-export routing option enables automatic route distribution between sender and receiver routing instances that are configured on the same router.

Extranet Sender Configuration

The template in Table 86 provides configuration at the sender site.

Table 86: Multicast Layer 3 VPN Service—Extranet Sender Configuration

Template ID	mvpn-sender-extranet	**Device Role**	PE router, DCR
Service	Layer 3 VPN	**Guidelines**	Use to configure VPN with extranet multicast receivers
Parameters			
sender_cust_name	Customer name of the multicast sender		
receiver_target	Route target of the receiver		

```
routing-instances {
    l3vpn-${sender_cust_name} {
        /* Enable auto-export to support receiver / sender VRFs on the same PE */
        routing-options {
            auto-export;
        }
        protocols {
            mvpn {
                route-target {
                    import-target {
                        target ${receiver_target};
                    }
                }
            }
        }
    }
}
```

The sender site imports C-multicast routes marked with the receiver target community. The sender uses regular unicast VRF target for its MVPN routes.

Firewall and Network Address Translation Service-Specific Configuration

Two different firewall/NAT services are described this solution:

- Firewalling between Layer 3 VPN sites
- Firewalling and NAT between Layer 3 VPN and the Internet

Provisioning the firewalling service between spoke sites requires configuring the following network elements:

- The virtual router instance on the firewall/NAT device
- The Layer 3 VPN attachment circuit used to communicate with spoke sites on both the DCR and firewall/NAT sides
- The Layer 3 VPN attachment circuit used to communicate with hub sites on both the DCR and firewall/NAT sides

  NOTE: There might be no hub sites, resulting in firewall and NAT required only between spoke sites.
- A security policy for traffic between spoke sites on the firewall/NAT device
- A security policy for traffic between spoke sites and hub sites on the firewall/NAT device

The following components are configured when provisioning firewall and NAT between Layer 3 VPN and the Internet:

- A virtual router instance on the firewall/NAT device
- A Layer 3 VPN attachment circuit
- A DIA attachment circuit on both the DCR and the firewall/NAT sides
- A security policy for traffic between the Internet and a Layer 3 VPN

The templates in the following sections provide exact definitions.

Firewall/NAT in a Layer 3 VPN

Two Layer 3 VPN attachment circuits are provisioned for this service; one is used to communicate with spoke sites and another is used to communicate with hub sites.

Configuration for both attachment circuits on the DCR router matches exactly the Layer 3 VPN PE router configuration with the following assumptions made:

- The routing protocol is BGP (see **l3vpn-bgp**).
- The AS override is enabled (see **l3vpn-bgp-as-override**).

The template in Table 87 describes the firewall/NAT device routing configuration.

Table 87: Firewall or NAT Configuration for Layer 3 VPN Routing

Template ID	fwnat-l3vpn-routing	**Device Role**	Firewall/NAT
Service	FW/NAT	**Guidelines**	
Dependencies	fwnat-geo-redundancy		
Parameters			
hub_ifd	Physical interface towards PE router with a hub VRF		
hub_unit	Logical interface towards PE router with a hub VRF		
spoke_ifd	Physical interface towards PE router with a spoke VRF		
spoke_unit	Logical interface towards PE router with a spoke VRF		
cust_name	Customer name, used to produce VRF name identifiers		
spoke_pe_v4	PE router IPv4 address		
spoke_pe_v6	PE router IPv6 address		
hub_pe_v4	PE router IPv4 address		
hub_pe_v6	PE router IPv6 address		
peer_as	Customer autonomous system		

```
routing-instances {
    /* Allocate a virtual router for each customer */
    vr-${cust_name} {
        instance-type virtual-router;
        /* List all interfaces towards Layer 3 VPN PE in a VR */
        interface ${hub_ifd}.${hub_ifl};
        interface ${spoke_ifd}.${spoke_ifl};
        protocols {
            bgp {
                group v4-l3vpn-pe {
                    type external;
                    /* Export routes to PE only if this device is active */
                    export [ pol-fwnat-conditional-export ];
                    peer-as ${peer_as};
                    neighbor ${spoke_pe_v4} {
                        family inet {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                    neighbor ${hub_pe_v4} {
                        family inet {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
                group v6-l3vpn-pe {
                    type external;
                    /* Export routes to PE only if this device is active */
                    export [ pol-fwnat-conditional-export ];
                    peer-as ${peer_as};
                    neighbor ${spoke_pe_v6} {
                        family inet6 {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
```

```
                        as-override;
                    }
                    neighbor ${hub_pe_v6} {
                        family inet {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
            }
        }
    }
}
```

The template in Table 88 provides firewall configuration for illustration purposes only. In this template, security policies allow all traffic. In the real deployments, security policies must reflect security requirements (details are not covered in this solution release).

Table 88: Firewall/NAT Configuration for Layer 3 VPN Security

Template ID	fwnat-l3vpn-security	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	
Parameters			
hub_ifd	Physical interface towards PE router with a hub VRF		
hub_unit	Logical interface towards PE router with a hub VRF		
spoke_ifd	Physical interface towards PE router with a spoke VRF		
spoke_unit	Logical interface towards PE router with a spoke VRF		
cust_name	Customer name, used to produce VRF name identifiers		
spoke_pe_v4	PE router IPv4 address		
spoke_pe_v6	PE router IPv6 address		
hub_pe_v4	PE router IPv4 address		
hub_pe_v6	PE router IPv6 address		
peer_as	Peer autonomous system		

```
security {
    policies {
        /* Two sample policies below allow all traffic between hub-and-spoke zones */
        from-zone z-hub-${cust_name} to-zone z-spoke-${cust_name} {
            policy permit-all {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
        from-zone z-spoke-${cust_name} to-zone z-hub-${cust_name} {
            policy permit-all {
                match {
                    source-address any;
                    destination-address any;
                    application any;
```

```
                }
                then {
                    permit;
                }
            }
        }
        /* This policy rejects all intra-zone traffic between spokes */
        from-zone z-spoke-${cust_name} to-zone z-spoke-${cust_name} {
            policy reject-all {
                then {
                    reject;
                }
            }
        }
    }
    zones {
        /* This security zone is for interfaces towards hub sites */
        security-zone z-hub-${cust_name} {
            /* Enable ping and traceroute */
            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                ${hub_ifd}.${hub_ifl};
            }
        }
        /* This security zone is for interfaces towards spoke sites */
        security-zone z-spoke-${cust_name} {
            /* Enable ping and traceroute */
            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                ${spoke_ifd}.${spoke_ifl};
            }
        }
    }
}
```

Firewall/NAT Between Layer 3 VPN and the Internet

The service that enables firewall/NAT between a Layer 3 VPN and the Internet requires the following:

- DIA configuration on the PE/DCR router
- Layer 3 VPN configuration on the PE/DCR router
- Firewall and NAT configuration on the firewall/NAT device (including connectivity and routing to the PE devices and security services configuration)

The DIA service configuration template with default route injection should be used for DIA service provisioning (**dia-customer-bgp-default-only**).

Layer 3 VPN configuration on the PE/DCR router is based on a regular Layer 3 VPN BGP routing template (**l3vpn-bgp**).

The template in Table 89 provides the firewall/NAT device routing configuration.

Table 89: Firewall/NAT Configuration for Layer 3 VPN Internet Access

Template ID	fwnat-dia-l3vpn-routing	**Device Role**	Firewall/NAT
Service	Internet access from Layer 3 VPN	**Guidelines**	
Dependencies	fwnat-geo-redundancy		
Parameters			
dia_ifd	Physical interface towards DIA PE router		
dia_unit	Logical interface towards DIA PE router		
l3vpn_ifd	Physical interface towards Layer 3 VPN PE router		
l3vpn_unit	Logical interface towards Layer 3 VPN PE router		
cust_name	Customer name, used to produce VRF name identifiers		
l3vpn_pe_v4	PE router IPv4 address		
l3vpn_pe_v6	PE router IPv6 address		
dia_pe_v4	PE router IPv4 address		
dia_pe_v6	PE router IPv6 address		
peer_as	Peer autonomous system		

```
routing-instances {
    /* Allocate a virtual router for each customer */
    vr-${cust_name} {
        instance-type virtual-router;
        /* List all interfaces in a VR */
        interface ${dia_ifd}.${dia_ifl};
        interface ${l3vpn_ifd}.${l3vpn_ifl};
        protocols {
            bgp {
                group v4-l3vpn-pe {
                    type external;
                    /* Export routes to PE only if this device is active */
                    export [ pol-fwnat-conditional-export ];
                    peer-as ${peer_as};
                    neighbor ${l3vpn_pe_v4} {
                        family inet {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
                group v6-l3vpn-pe {
                    type external;
                    /* Export routes to PE only if this device is active */
                    export [ pol-fwnat-conditional-export ];
                    peer-as ${peer_as};
                    neighbor ${l3vpn_pe_v6} {
                        family inet6 {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
```

```
                /* Note! Export policy towards DIA peer depends on NAT configuration
                   It is provided separately.
                 */
                group v4-dia-pe {
                    type external;
                   peer-as ${peer_as};                    neighbor ${dia_pe_v4} {
                        family inet {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
                group v6-dia-pe {
                    type external;
                    peer-as ${peer_as};
                    neighbor ${dia_pe_v6} {
                        family inet6 {
                            unicast;
                        }
                        /* Override AS number, enable route distribution back to the same
                        system */
                        as-override;
                    }
                }
            }
        }
    }
}
```

The template in Table 90 provides security configuration for this service. Two security zones are defined—one for interfaces towards the Internet and another for interfaces towards the Layer 3 VPN. The security policy enables all sessions from the Layer 3 VPN towards the Internet and blocks all session initiation attempts from the Internet (there is no policy configured, so default policy applies).

Table 90: Firewall/NAT Configuration for Layer 3 VPN Security Policy Outbound

Template ID	fwnat-dia-l3vpn-security	Device Role	Firewall/NAT
Service	Firewall/NAT	Guidelines	
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
dia_pe_v4	PE router IPv4 address		
dia_pe_v6	PE router IPv6 address		
peer_as	Peer autonomous system		

```
security {
    policies {
        /* Allow traffic from Layer 3 VPN to Internet, no restrictions */
        from-zone z-vpn-${cust_name} to-zone z-internet-${cust_name} {
            policy permit-all {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
            }
```

```
        }
    }
    zones {
        /* This security zone is for interfaces towards Internet */
        security-zone z-internet-${cust_name} {
           /* Enable ping and traceroute */            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                ${dia_ifd}.${dia_ifl};
            }
        }
        /* This security zone is for interfaces towards VPN */
        security-zone z-vpn-${cust_name} {
            /* Enable ping and traceroute */
            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                ${l3vpn_ifd}.${l3vpn_ifl};
            }
        }
    }
}
```

The template must be modified if Internet access service from the Layer 3 VPN is combined with firewalling between Layer 3 VPN spoke locations. The main difference is there could be two different security zones on the Layer 3 VPN side (one for spoke sites and another for hub sites). The template in Table 91 assumes that both security zones (for hub and for spoke sites) are preconfigured earlier (see the **fwnat-l3vpn-security** template).

Table 91: Firewall/NAT Configuration for Layer 3 VPN Intra-Spoke Security Policy

Template ID	fwnat-dia-l3vpn-security-combo	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	Internet access from Layer 3 VPN with firewall between spokes
Dependencies	fwnat-l3vpn-security		
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
dia_pe_v4	PE router IPv4 address		
dia_pe_v6	PE router IPv6 address		
peer_as	Peer autonomous system		

```
security {
    policies {
        /* Allow traffic from hub sites to Internet, no restrictions */
        from-zone z-hub-${cust_name} to-zone z-internet-${cust_name} {
            policy permit-all {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
```

```
            }
        }
        /* Allow traffic from spoke sites to Internet, no restrictions */
       from-zone z-spoke-${cust_name} to-zone z-internet-${cust_name} {
           policy permit-all {                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
    }
    zones {
        /* This security zone is for interfaces towards Internet */
        security-zone z-internet-${cust_name} {
            /* Enable ping and traceroute */
            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                ${dia_ifd}.${dia_ifl};
            }
        }
    }
}
```

Internet access from Layer 3 VPN is typically complemented with some form of Network Address Translation. If network translation is enabled, NAT pool prefixes allocated to the customer must be announced to the Internet and every other prefix must be blocked. The template in Table 92 is used to configure NAT service and prefix export policy towards DIA router.

Table 92: Firewall/NAT Configuration for Layer 3 VPN Routing—NAT

Template ID	fwnat-dia-l3vpn-routing-nat	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	NAT configuration
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		
v4_pool_start	V4 NAT pool start address		
v4_pool_end	V4 NAT pool end address		
v4_pool_route	An array of V4 pool routes		
v6_pool_start	V6 NAT pool start address		
v6_pool_end	V6 NAT pool end address		
v6_pool_route	An array of V6 pool routes		

```
security {
    nat {
        source {
            /* NAT pool (v4) */
            pool v4-pool-${cust_name} {
                routing-instance {
                    vr-${cust_name};
                }
                address {
                    ${v4_pool_start} to ${v4_pool_end};
                }
```

```
            }
            /* NAT pool (v6) */
            pool v6-pool-${cust_name} {
           routing-instance {
                   vr-${cust_name};
               }
               address {
                   ${v6_pool_start} to ${v6_pool_end};
               }
            }
        }
    }
}
policy-options {
    /* This policy is used to export NAT pool static routes to DIA PE */
    policy-statement p-natpool-${cust_name} {
        /* Allow NAT Pool Address*/
        term accept-v4-nat-pool {
            from {
                protocol static;
                /* List all NAT pool routes here */
                route-filter ${v4_pool_route[i]} exact;
                /* Export the NAT pool route only if this device is active */
                condition fwnat-active-device;
            }
            then accept;
        }
        term accept-v6-nat-pool {
            from {
                protocol static;
                /* List all NAT pool routes here */
                route-filter ${v6_pool_route[i]} exact;
                /* Export the NAT pool route only if this device is active */
                condition fwnat-active-device;
            }
            then accept;
        }
        term reject-the-rest {
            then reject;
        }
    }
}
routing-instances {
    vr-${cust_name} {
        routing-options {
            rib vr-${cust_name}.inet6.0 {
                static {
                    /* NAT pool (v6) */
                    route ${v6_pool_route[i]} discard;
                    no-install;
                }
            }
            static {
                /* NAT pool (v4) */
                route ${v4_pool_route[i]} discard;
                no-install;
            }
        }
        protocols {
            bgp {
               /* Export NAT pool prefixes only */
               group v4-dia-pe {
                    export [ p-natpool-${cust_name} ];
                }
               group v6-dia-pe {
```

```
                    export [ p-natpool-${cust_name} ];
                }
            }
        }
    }
}
```

In the event Internet access service is deployed without NAT and with only firewall configuration, you should apply a regular export policy toward the DIA PE device. The template in Table 93 provides this configuration.

Table 93: Firewall/NAT Configuration for Layer 3 VPN Routing Without NAT

Template ID	fwnat-dia-l3vpn-routing-fw-only	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	Use in firewall only configuration
Dependencies	fwnat-geo-redundancy		
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		

```
routing-instances {
    vr-${cust_name} {
        protocols {
            bgp {
                /* Export all prefixes, if this device is active */
                group v4-dia-pe {
                     export [ pol-fwnat-conditional-export ];
                 }
                group v6-dia-pe {
                     export [ pol-fwnat-conditional-export ];
                 }
            }
        }
    }
}
```

The template in Table 94 provides NAT security configuration.

Table 94: Firewall/NAT Configuration for Layer 3 VPN Routing—NAT Security

Template ID	fwnat-dia-l3vpn-security-nat	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	NAT configuration for a regular Layer 3 VPN
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		

```
security {
    nat {
        source {
            rule-set rs-internet-access-${cust_name} {
                from zone z-vpn-${cust_name};
                to zone z-internet-${cust_name};
                rule v4 {
                    match {
                        source-address 0.0.0.0/0;
                    }
                    then {
                        source-nat {
                            pool {
```

```
                                v4-pool-${cust_name};
                            }
                        }
                    }
                }
                rule v6 {
                    match {
                        source-address 0::0/0;
                    }
                    then {
                        source-nat {
                            pool {
                                v6-pool-${cust_name};
                            }
                        }
                    }
                }
            }
        }
    }
}
```

The template in Table 95 provides a minor difference in security zone names in the event NAT service is deployed in combination with the firewall/NAT service between Layer 3 VPN locations.

Table 95: Firewall/NAT Configuration for Layer 3 VPN Routing—Intra-Spoke Firewall/NAT

Template ID	fwnat-dia-l3vpn- security-nat-combo	**Device Role**	Firewall/NAT
Service	Firewall/NAT	**Guidelines**	NAT configuration deployed in combination with firewalling between Layer 3 VPN locations
Parameters			
cust_name	Customer name, used to produce VRF name identifiers		

```
security {
    nat {
        source {
            rule-set rs-internet-access-${cust_name} {
                /* Enable source NAT for traffic from two Layer 3 VPN zones */
                from zone [ z-hub-${cust_name} z-spoke-${cust_name}];
                to zone z-internet-${cust_name};
                rule v4 {
                    match {
                        source-address 0.0.0.0/0;
                    }
                    then {
                        source-nat {
                            pool {
                                v4-pool-${cust_name};
                            }
                        }
                    }
                }
                rule v6 {
                    match {
                        source-address 0::0/0;
                    }
                    then {
                        source-nat {
                            pool {
                                v6-pool-${cust_name};
```

```
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
}
```

CHAPTER 6 Basic Connectivity Services

- Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6
- Configuration Scenario: Layer 2 Circuits for Termination into Layer 3 VPNs
- Configuration Scenario: Direct Internet Access Service

Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for this tested Layer 3 VPN service configuration:

- Topology
- Router Configurations
- Interface Guidelines
- Core Device Protocol Guidelines
- Routing Instance Guidelines
- Class-of-Service Guidelines
- Verification

Topology

Figure 26 shows the topology tested for the Layer 3 VPN service.

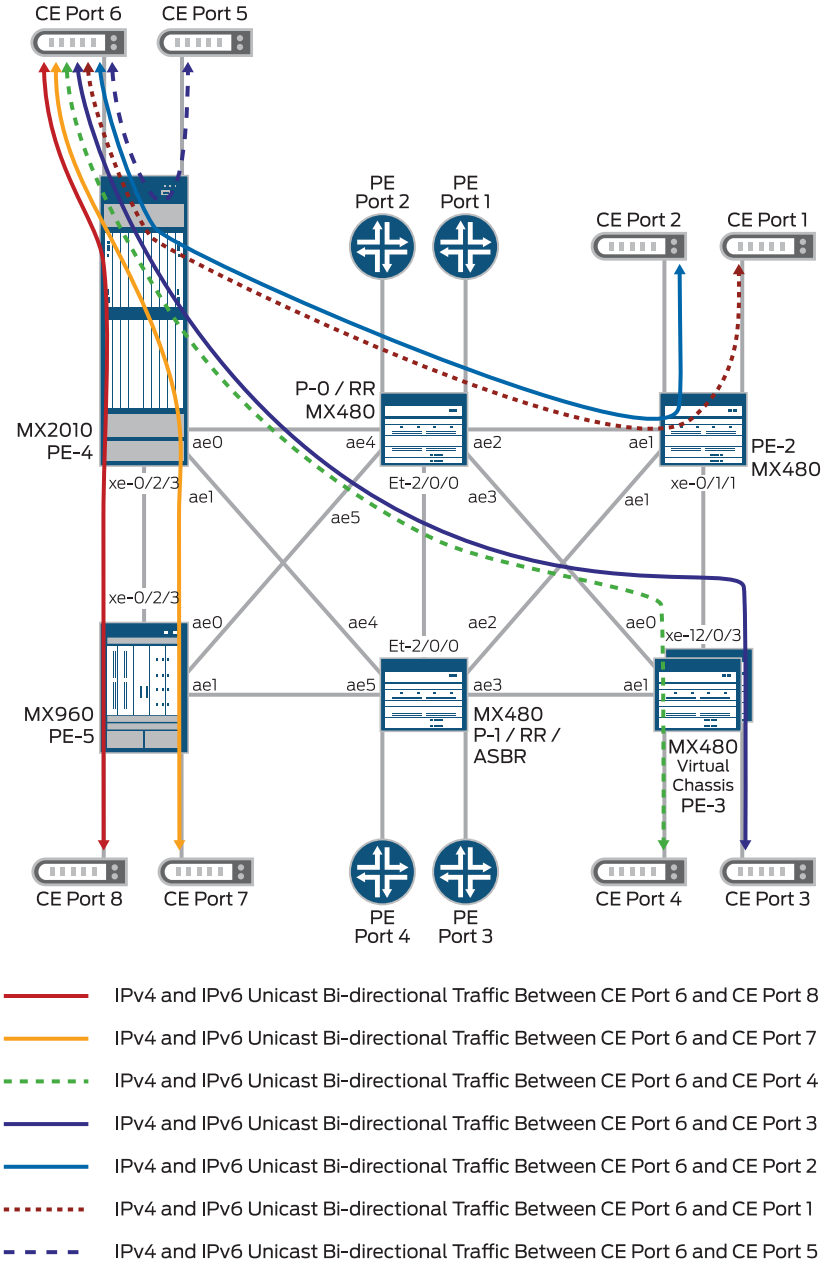

Figure 26: Layer 3 VPN service configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Interface Guidelines

Interface configuration for the Layer 3 VPN service consists of two general components: core interface configuration and edge (PE-CE) interface configuration.

In this scenario, the following guidelines were followed when configuring the device interfaces:

- Core interfaces are configured with MTU 9192, and edge interfaces are configured with MTU 9122.
- Edge interfaces are configured with single VLAN tagging, whereas core interfaces are untagged.
- Core interfaces are part of AE bundles and are configured with LACP in active mode and minimum-links.
- Loopback interfaces are configured on provider core and edge routers. The Unit 0 primary addresses of the loopback interfaces are used for Layer 3 VPN.

Core Device Protocol Guidelines

In this scenario, the following guidelines were followed when configuring protocols for core devices:

- IS-IS level 2 point-to-point is configured as the core IGP protocol.
- RSVP is configured with 95 percent subscription, aggregate, and link protection as the transport protocol for MPLS LSPs.
- Full mesh MPLS LSPs are configured with node-link protection, adaptive, and auto-bandwidth.
- To facilitate load balancing, 16 LSPs are configured between each PE device. In addition, BGP multipath, per packet load balancing, and aggregated Ethernet interfaces are also configured.
- Devices with PE–to-CE router connections are configured with static, BGPv4, BGPv6, OSPFv2, and OSPFv3 protocols.
- To facilitate link-level fault detection, BFD is configured for IS-IS, and both BGP and LFM are configured on all core-facing interfaces.
- For security, IS-IS and BGP protocol exchanges are authenticated using Message Digest 5 (MD5) authentication.
- To ensure consistent path selection before and after Routing Engine switchover events, the BGP path selection algorithm is configured to compare external router identifiers.

Routing Instance Guidelines

A routing instance is a collection of routing tables, interfaces, and routing protocol parameters. The set of interfaces specified in the routing instance belongs to the routing tables, and the parameters defined within the routing instance control the protocol information in the routing tables.

You can create multiple instances of BGP, OSPF, RIP, and static routes for Layer 3 VPN implementation. The multiple instances of BGP, OSPF, and RIP keep routing information for different VPNs separate. The VRF instance advertises routes from the CE router to the PE router, and advertises routes from the PE router to the CE router. Each VPN receives only routing information belonging to that VPN.

In this scenario, the following guidelines were followed when configuring routing instances:

- Each PE router was configured and tested with 4,000 VRFs.
- The **vrf-table-label** option is configured in the base VRF for better Layer 3 VPN scaling and performance.
- The VPN routing and forwarding routing (VRF) instance type is used for Layer 3 VPN implementations. This routing instance type has a VPN routing table as well as a corresponding VPN forwarding table. For this instance type, there is a one-to-one mapping between an interface and a routing instance. Each VRF instance corresponds with a forwarding table. Routes on an interface go into the corresponding forwarding table.

- Static routes are implemented in all VRF routing instances for this solution.
- Multiple instances of BGP are primarily used for Layer 3 VPN support. BGP is configured on the PE router to receive routes from the CE router and to send the instance routes to the CE router, if necessary. Multiple instances of BGP are used to maintain separate per-site forwarding tables, keeping VPN traffic separate on the PE router.
- Multiple instances of OSPF are used for Layer 3 VPN implementations. The multiple instances of OSPF keep routing information for different VPNs separate. The VRF instance advertises routes from the CE router to the PE router and advertises routes from the PE router to the CE router. Each VPN receives only routing information belonging to that VPN.

Class-of-Service Guidelines

Four different CoS profiles were configured for Layer 3 VPN customers in this scenario.

VPN CoS Profile #1 Configuration

This profile is assigned to advanced VPN customers that transmit eight classes of traffic.

The actions taken on PE-to-CE ingress traffic are as follows:

- Traffic is classified by DSCP in a BA classifier.
- Per-class color-aware policers are enabled on ingress.
- The expedited forwarding class is policed to a certain rate. For each assured forwarding class, there is a committed limit within which traffic is accepted. Traffic that falls outside of this committed limit is allowed, but it is marked.
- All customer traffic is policed by a hierarchical policer to a certain maximum rate. All traffic that falls within this maximum rate is treated as premium.
- Best-effort traffic and fc-af1 out-of-spec traffic is treated as non-premium.
- Ingress PE-to-CE traffic is classified into aggregated (core) forwarding classes. Full DSCP transparency is provided and DSCP fields remain intact.

The actions taken on PE-to-CE egress traffic are as follows:

- Traffic is classified based on DSCP or traffic class into edge forwarding classes.
- Each traffic class is serviced in a dedicated queue.
- Each queue is allocated with certain bandwidth.
- Expedited forwarding class queue is rate-limited to avoid buffering and reduce jitter.
- Assured forwarding and network control classes have bandwidth allocation preference over best-effort class.

NOTE: This rule also applies to excess traffic.

VPN CoS Profile #2 Configuration

This profile is identical to VPN CoS profile #1, with the following differences:

- Hierarchical policing is not configured on ingress PE-CE traffic.
- The guaranteed minimum transmission rate is ***not*** set for egress PE-CE traffic.

VPN CoS Profile #3 Configuration

This profile is assigned to advanced VPN customers that transmit eight classes of traffic in a number of VPN connections (logical interface sets). The difference between this profile and profile #1 and profile #2 are as follows:

- Traffic received from a group of interfaces (interface set) is policed to an aggregate rate.
- Traffic that is in-spec is assigned a priority over other traffic, as long as the in-spec traffic remains within a priority traffic limit.

Each VPN connection has a certain guaranteed traffic limit. The egress traffic rate can exceed the guaranteed limit up to a certain maximum limit as specified for each connection group (logical interface set).

VPN CoS Profile #4 Configuration

This profile is assigned to advanced VPN customers that transmit 8 classes of traffic in a number of VPN connections (logical interface sets). This profile has hierarchical ingress queuing configured in both the ingress and egress directions.

Verification

The following sections provide methods for verifying this Layer 3 VPN configuration scenario along with sample command outputs:

- Verify IS-IS Layer 2 Adjacency on All PE and P Routers
- Verify MP-IBGP Peering on All PE Devices
- Verify EBGP Peering with the CE Device on All PE Devices
- Verify EBGP6 Peering with the CE Device on All PE Devices
- Verify OSPF Adjacency with the CE Device on All PE Devices
- Verify OSPFv3 Adjacency with the CE Device on All PE Devices
- Verify LFM Session in Core-Facing Interfaces on All PE Devices
- Verify RSVP Neighbors in the Core on All PE and P Devices
- Verify Local CE Routes in the VRF Table of All PE Devices
- Verify Remote CE Device IPv4 Routes in the VRF Table of All PE Devices
- Verify Remote CE Device IPv4 Routes
- Verify Label-Switched Path Details
- Verify Forwarding Table Details
- Verify MPLS Table Transport Label Entries
- Verify MPLS Table VPN Label Entries
- Verify VRF Table IPv4 Local Route Entries
- Verify Remote CE Device IPv6 Routes in the VRF Table of All PE Devices
- Verify Remote CE Device IPv6 Routes
- Verify Label-Switched Path Details
- Verify VRF Table IPv6 Local Route Entries
- Verify Local PE-to-CE Interface Statistics
- Verify Local PE Core Device Interface Statistics
- Verify LSP Statistics on the PE Device
- Verify Remote PE-to-CE Device Interface Statistics

Verify IS-IS Layer 2 Adjacency on All PE and P Routers

Purpose View the state of IS-IS Layer 2 adjacency for all core-facing interfaces configured on PE and P devices.

Action Issue the following command:

regress@host# **run show isis adjacency**

```
Interface             System           L State       Hold (secs) SNPA
ae0.0                 sachem           2  Up         22
ae1.0                 hawkeye          2  Up         26
ae6.0                 haris            2  Up         19
xe-0/2/3.0            vinson           2  Up         21
```

Meaning The output indicates that IS-IS Layer 2 adjacency on all of the core-facing interfaces of device PE4 are in an Up state.

Verify MP-IBGP Peering on All PE Devices

Purpose View the state of MP-IBGP neighborship with the router reflectors on the PE devices.

Action Issue the following command:

regress@host# **run show bgp neighbor 10.255.50.71**

```
Peer: 10.255.50.71+179 AS 70  Local: 10.255.50.49+49473 AS 70
  Type: Internal    State: Established    Flags: <ImportEval Sync
RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ next-hop-self pol-export-directs pol-export-rr pol-final ]
  Options: <Preference LocalAddress AddressFamily Multipath Rib-
group Refresh>
  Options: <MultipathAs AutheKeyChain BfdEnabled>
  Authentication key chain: chain1
  Authentication algorithm: hmac-sha-1-96
  Address families configured: inet-unicast inet-multicast inet-vpn-
unicast inet6-unicast inet6-multicast inet6-vpn-unicast route-target
  Local Address: 10.255.50.49 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 10.255.50.71   Local ID: 10.255.50.49   Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 0
  BFD: enabled, up
  NLRI for restart configured on peer: inet-unicast inet-multicast inet-
vpn-unicast inet6-unicast inet6-multicast inet6-vpn-unicast route-
target
  NLRI advertised by peer: inet-unicast inet-multicast inet-vpn-unicast
inet6-unicast inet6-multicast inet6-vpn-unicast route-target


  NLRI for this session: inet-unicast inet-multicast inet-vpn-unicast inet6-
unicast inet6-multicast inet6-vpn-unicast route-target
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality  NLRI that restart is negotiated
for: inet-unicast inet-multicast inet-vpn-unicast inet6-unicast inet6-multicast
inet6-vpn-unicast route-target
  NLRI of received end-of-rib markers: inet-unicast inet-multicast inet-vpn-
unicast inet6-unicast inet6-multicast inet6-vpn-unicast route-target
  NLRI of all end-of-rib markers sent: inet-unicast inet-multicast inet-vpn-
unicast inet6-unicast inet6-multicast inet6-vpn-unicast route-target
  Peer supports 4 byte AS extension (peer-as 70)
  Peer does not support Addpath
```

Meaning The output indicates that MP-IBGP neighborship between the PE4 device and route reflector P0 is established. All BGP packet exchanges are authenticated. The following address families are exchanged—inet-unicast, inet-multicast, inet-vpn-unicast, inet6-unicast, inet6-multicast, inet6-vpn-unicast, and route-target. The BFD session over BGP is Up.

Verify EBGP Peering with the CE Device on All PE Devices

Purpose View the state of EBGP IPv4 neighborship with the CE device inside the VPN routing instance on the PE device.

Action Issue the following command:

regress@host# **run show bgp neighbor 2.0.9.38 instance vrf1**

```
Peer: 2.0.9.38+43601 AS 1001  Local: 2.0.9.37+179 AS 70
  Type: External    State: Established    Flags: <ImportEval Sync
RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Options: <Preference AddressFamily PeerAS Refresh>
  Address families configured: inet-unicast
  Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 2.0.9.38        Local ID: 2.0.9.37       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 4    Peer index: 3
  BFD: disabled, down
  Local Interface: xe-2/0/1.1
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table vrf1.inet.0 Bit: a0001
    RIB State: BGP restart is complete
    RIB State: VPN restart is complete
    Send state: in sync
    Active prefixes:              400
    Received prefixes:            400
    Accepted prefixes:            400
    Suppressed due to damping:    0
    Advertised prefixes:          133
```

Meaning The output indicates that the EBGP IPv4 neighborship is established between the PE4 device and CE device 2.0.9.38 inside VPN routing instance vrf1. The PE device has received and accepted 400 prefixes from the CE device and advertises 133 prefixes to the CE device.

Verify EBGP6 Peering with the CE Device on All PE Devices

Purpose View the state of EBGP IPv6 neighborship with the CE device inside the VPN routing instance on the PE device.

Action Issue the following command:

regress@host# **run show bgp neighbor 2002:0000:0000:0000:0000:0000:0200:0926 instance vrf1**

```
Peer: 2002::200:926+43551 AS 1001  Local: 2002::200:925+179 AS 70
 Type: External    State: Established    Flags: <ImportEval Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Options: <Preference AddressFamily PeerAS Refresh>
  Address families configured: inet6-unicast
  Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 2.0.9.38     Local ID: 2.0.9.37     Active Holdtime: 90
  Keepalive Interval: 30     Group index: 4     Peer index: 2
  BFD: disabled, down
  Local Interface: xe-2/0/1.1
  NLRI for restart configured on peer: inet6-unicast
  NLRI advertised by peer: inet6-unicast
  NLRI for this session: inet6-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table vrf1.inet6.0 Bit: b0001
    RIB State: BGP restart is complete
    RIB State: VPN restart is complete
    Send state: in sync
    Active prefixes:              400
    Received prefixes:            400
    Accepted prefixes:            400
    Suppressed due to damping:    0
    Advertised prefixes:          133
```

Meaning The output indicates that the EBGP IPv6 neighborship is established between the PE4 device and CE device 2002:0000:0000:0000:0000:0000:0200:0926 inside the VPN routing instance vrf1. The PE device has received and accepted 400 prefixes from the CE device and advertises 133 prefixes to the CE device.

Verify OSPF Adjacency with the CE Device on All PE Devices

Purpose View the state of OSPF adjacency with the CE device inside the VPN routing instance on the PE device.

Action Issue the following command:

regress@host# **run show ospf neighbor instance vrf32**

```
Address         Interface         State     ID            Pri     Dead
2.0.11.194      xe-2/0/0.32       Full      2.0.11.194     0       31
2.0.9.162       xe-2/0/1.32       Full      2.0.9.162      0       31
```

Meaning The output indicates that the OSPF adjacency with the CE devices inside the VPN routing instance is in a Full state.

Verify OSPFv3 Adjacency with the CE Device on All PE Devices

Purpose View the state of OSPFv3 adjacency with the CE device inside the VPN routing instance on the PE devices.

Action Issue the following command:

regress@host# **run show ospf3 neighbor instance vrf32**

```
ID                  Interface              State       Pri      Dead
2.0.11.194          xe-2/0/0.32            Full         0       35
  Neighbor-address fe80::200:ff:fe00:5dc
2.0.9.162           xe-2/0/1.32            Full         0       35
  Neighbor-address fe80::200:ff:fe00:4ca
```

Meaning The output indicates that the OSPFv3 adjacency with the CE devices inside the VPN routing instance is in a Full state.

Verify LFM Session in Core-Facing Interfaces on All PE Devices

Purpose View the status of LFM sessions on the core-facing interfaces in PE and P routers.

Action Issue the following command:

regress@host# **run show oam ethernet link-fault-management xe-0/1/7**

```
  Interface: xe-0/1/7
    Status: Running, Discovery state: Send Any
    Peer address: b0:a8:6e:2a:a8:54
    Flags:Remote-Stable Remote-State-Valid Local-Stable 0x50
    Remote entity information:
      Remote MUX action: forwarding, Remote parser action: forwarding
       Discovery mode: active, Unidirectional mode: unsupported
       Remote loopback mode: unsupported, Link events: supported
       Variable requests: unsupported
```

Meaning The output indicates that LFM is running in active mode on the interface, the local device has discovered the remove device, and the link is fully operational.

Verify RSVP Neighbors in the Core on All PE and P Devices

Purpose View the state of RSVP neighbors in PE and P devices.

Action Issue the following command:

regress@host# **run show rsvp neighbor**

```
RSVP neighbor: 4 learned
Address     Idle   Up/Dn   LastChange  HelloInt  HelloTx/Rx  MsgRcvd
1.0.0.29       0     1/0     1:03:25         9       422/422     68524
1.0.0.13       0     1/0     1:07:26         9       448/448     104194
1.0.0.42       0     1/0     1:08:08         9       452/451     16543
1.0.0.46       0     1/0     1:07:44         9       450/450     7247
```

Meaning This PE device has learned four RSVP neighbors and all the neighbors are in the Up state.

Verify Local CE Routes in the VRF Table of All PE Devices

Purpose View the directly connected CE device routes in the PE device VPN routing table.

Action Issue the following command:

regress@host# **run show route 7.0.62.128**

```
//routes learned through BGP/BGP+//
vrf1.inet.0: 3348 destinations, 5883 routes
(3348 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

7.0.62.128/30   *[BGP/170] 12:39:26, localpref 100
                   AS path: 1001 {1001} I, validation-state: unverified
                 > to 2.0.11.70 via xe-2/0/0.1
```

Issue the following command:

regress@host# **run show route 2002::700:3e80**

```
//routes learned through BGP/BGP+//
vrf1.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::700:3e80/126 *[BGP/170] 12:41:23, localpref 100
                  AS path: 1001 {1001} I, validation-state: unverified
                > to 2002::200:b46 via xe-2/0/0.1
```

Issue the following command:

regress@host# **run show route 7.0.68.192**

```
//routes learned through OSPF/OSPFv3//
vrf32.inet.0: 3349 destinations, 5885 routes (3349 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

7.0.68.192/30      *[OSPF/10] 12:42:16, metric 1
                    > to 2.0.11.194 via xe-2/0/0.32
```

Issue the following command:

regress@host# **run show route 2002::700:44c0**

```
//routes learned through OSPF/OSPFv3//
vrf32.inet6.0: 3355 destinations, 5894 routes (3355 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::700:44c0/126 *[OSPF3/10] 12:42:50, metric 1
                    > to fe80::200:ff:fe00:5dc via xe-2/0/0.32
```

Meaning This PE device has learned the routes 7.0.62.128 and 2002::700:3e80 from its directly connected CE device neighbor through BGP and installed the routes in the vrf1 routing table. The PE device has learned the routes 7.0.68.192 and 2002::700:44c0 from its directly connected CE device neighbor through OSPF/OSPFv3 and installed the routes in the vrf32 routing table.

Verify Remote CE Device IPv4 Routes in the VRF Table of All PE Devices

Purpose View the remote CE device routes in the PE device VPN routing table.

Action Issue the following command:

regress@host# **run show route 7.0.25.0**

```
//Remote CE IPv4 route//
vrf1.inet.0: 3348 destinations, 5883 routes (3348 active,
0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
7.0.25.0/30        *[BGP/170] 12:35:57, localpref 100, from
10.255.50.71 -> route received from RR0
                    AS path: 1001 {1001} I, validation-state: unverified
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-1
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-2
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-3
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-4
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-5
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-6
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-7
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-8
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-9
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-10
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-11
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-12
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-13
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-14
                  > to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-15
-> LSP to PE3 since remote CE is connected to PE3
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-16
                  [BGP/170] 12:35:56, localpref 100, from 10.255.50.73 ->
route received from RR1
                    AS path: 1001 {1001} I, validation-state: unverified
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-1
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-2
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-3
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-4
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-5
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-6
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-7
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-8
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-9
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-10
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-11
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-12
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-13
                    to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-14
                  > to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-15
                    to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-16
```

Meaning The output indicates that the PE device has received and accepted remote CE device route 7.0.25.0 from the PE3 device through route reflectors 10.255.50.71 and 10.55.50.73 using MP-IBGP. The PE device has installed the route in its vrf1 routing table and has selected the route received from address 10.255.50.71 as the best route. It is using LSP r4-to-r3-15 to reach the PE3 device.

Verify Remote CE Device IPv4 Routes

Purpose View the MPLS label values, BGP next-hop address, and LSP next-hop address associated with the VPN route installed in the PE device VPN routing table.

Action Issue the following command:

regress@host# **run show route 7.0.25.0 detail**

```
vrf1.inet.0: 3348 destinations, 5883 routes (3348 active, 0 holddown,
0 hidden)

7.0.25.0/30 (2 entries, 1 announced)
   *BGP    Preference: 170/-101
           Route Distinguisher: 10.255.50.79:1
           Next hop type: Indirect
           Address: 0x2300a824
           Next-hop reference count: 4039
           Source: 10.255.50.71 -> RR0
           Next hop type: Router, Next hop index: 1048578

           Next hop: 1.0.0.29 via ae1.0 weight 0x1, selected -> LSP
next-hop is P1
           Label-switched-path r4-to-r3-15
           Label operation: Push 564376 -> Transport label
           Label TTL action: no-prop-ttl -> No TTL propagation
           Session Id: 0x752d

           Protocol next hop: 10.255.50.79 -> Remote PE3
           Push 16
           Composite next hop: 2300b018 1115407 INH Session ID: 0x7507
           Indirect next hop: f6dc0ec 1110665 INH Session ID: 0x7507
           State: <Secondary Active Int Ext ProtectionCand>
           Local AS:    70 Peer AS:    70
           Age: 13:06:25   Metric2: 33554428
           Validation State: unverified
           Task: BGP_70.10.255.50.71+64165
           Announcement bits (1): 1-KRT
           AS path: 1001 {1001} I (Originator)
           Cluster list:  10.255.50.71
           Originator ID: 10.255.50.79
           Communities: target:70:1 src-as:70:0 rt-import:10.255.50.79:10
           Import Accepted Multipath -> Multipath is enabled
           VPN Label: 16 -> VPN label
           Localpref: 100
           Router ID: 10.255.50.71
           Primary Routing Table bgp.l3vpn.0
    BGP    Preference: 170/-101
           Route Distinguisher: 10.255.50.79:1
           Next hop type: Indirect
           Address: 0x2300a824
           Next-hop reference count: 4039
           Source: 10.255.50.73
           Next hop type: Router, Next hop index: 1048578

           Next hop: 1.0.0.29 via ae1.0 weight 0x1, selected
           Label-switched-path r4-to-r3-15
           Label operation: Push 564376
           Label TTL action: no-prop-ttl
           Session Id: 0x752d
```

```
                    Protocol next hop: 10.255.50.79
                    Push 16
                    Composite next hop: 2300b018 1115407 INH Session ID: 0x7507
                    Indirect next hop: f6dc0ec 1110665 INH Session ID: 0x7507
                    State: <Secondary NotBest Int Ext ProtectionCand>
                    Inactive reason: Not Best in its group - Update source
                    Local AS:    70 Peer AS:    70
                    Age: 13:06:24   Metric2: 33554428
                    Validation State: unverified
                    Task: BGP_70.10.255.50.73+53706
                    AS path: 1001 {1001} I (Originator)
                    Cluster list:  10.255.50.73
                    Originator ID: 10.255.50.79
                    Communities: target:70:1 src-as:70:0 rt-import:10.255.50.79:10
                    Import Accepted MultipathContrib
                    VPN Label: 16
                    Localpref: 100
                    Router ID: 10.255.50.73
                    Primary Routing Table bgp.l3vpn.0
```

Meaning VPN route 7.0.25.0/30 is installed in the vrf1 routing table in the PE device using a VPN label of 16 and an LSP label of 564376. The BGP next hop of the route is 10.255.50.79 (PE3) and the LSP next hop is 1.0.0.29.

Verify Label-Switched Path Details

Purpose View the status of MPLS label-switched paths (LSPs).

Action Issue the following command:

regress@host# **run show mpls lsp name r4-to-r3-15 detail**

```
IIngress LSP: 1066 sessions
10.255.50.79
  From: 10.255.50.49, State: Up, ActiveRoute: 0, LSPname: r4-to-r3-15
  ActivePath:  (primary) -> Primary path
  Node/Link protection desired -> Node-link protection is configured
  LSPtype: Static Configured, Penultimate hop popping
  LoadBalance: Random
  Autobandwidth -> Autobandwidth is configured
  MaxBW: 8Gbps
  AdjustTimer: 10800 secs AdjustThreshold: 5%
  Max AvgBW util: 141.771Mbps, Bandwidth Adjustment in 337 second(s).
  Overflow limit: 3, Overflow sample count: 0
  Underflow limit: 0, Underflow sample count: 8, Underflow Max AvgBW:
131.971Mbps
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    Bandwidth: 141.799Mbps
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.29 S 1.0.0.26 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
           10.255.50.73(flag=0x21) 1.0.0.29(flag=1 Label=564376)
10.255.50.79(flag=0x20) 1.0.0.26(Label=3)
Total 1 displayed, Up 1, Down 0
```

Meaning The output indicates that the ingress LSP from 10.255.50.4 (PE4) to 10.255.50.79 (PE3) is Up and using the primary path. The LSP is node-link protected and configured with autobandwidth. The primary path is going through 1.0.0.29.

Verify Forwarding Table Details

Purpose View the forwarding table entry in the PE device for the VPN route 7.0.25.0.

Action Issue the following command:

regress@host# **run show route forwarding-table table vrf1 destination 7.0.25.0 extensive**

```
Routing table: vrf1.inet [Index 4008]
Internet:
Destination:  7.0.25.0/30
  Route type: user
  Route reference: 0                    Route interface-index: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite              Index: 1115407  Reference: 808
-> composite next-hop
  Next-hop type: indirect               Index: 1110665  Reference: 63
  Next-hop type: unilist                Index: 1048578  Reference: 5
  Nexthop: 1.0.0.29
  Next-hop type: Push 475128            Index: 111997   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 584024            Index: 106477   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 528027            Index: 104751   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 499611            Index: 112015   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 529608            Index: 112052   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 450395            Index: 109288   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 588216            Index: 113546   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 534824            Index: 232841   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 564888            Index: 104753   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 534792            Index: 232844   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 499595            Index: 112072   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 534808            Index: 236671   Reference: 1
  Next-hop interface: ae1.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 498107            Index: 110162   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.13
  Next-hop type: Push 472203            Index: 105125   Reference: 1
  Next-hop interface: ae0.0          Weight: 0x1
  Nexthop: 1.0.0.29
```

```
  Next-hop type: Push 564376           Index: 113015   Reference: 1
  Next-hop interface: ae1.0         Weight: 0x1
  Nexthop: 1.0.0.29
  Next-hop type: Push 564392           Index: 113064   Reference: 1
  Next-hop interface: ae1.0         Weight: 0x1
```

Meaning The output indicates that the PE device has installed VPN route 7.0.25.0 in its forwarding table with a composite next hop.

Verify MPLS Table Transport Label Entries

Purpose View the MPLS label table entry for the transport label 564376 in the P router.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 564376**

```
mpls.0: 5019 destinations, 5019 routes (5015 active, 4 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

564376        *[RSVP/7/1] 07:16:35, metric 1
               > to 1.0.0.26 via ae3.0, label-switched-path r4-to-r3-15
                 to 1.0.0.22 via ae2.0, label-switched-path
Bypass->1.0.0.26
564376(S=0)   *[RSVP/7/1] 07:16:35, metric 1
               > to 1.0.0.26 via ae3.0, label-switched-path r4-to-r3-15
                 to 1.0.0.22 via ae2.0, label-switched-path
Bypass->1.0.0.26
```

Meaning The P router pops the outer label and sends the packet with only the VPN label to the PE3 device over aggregated Ethernet interface ae3.0.

Verify MPLS Table VPN Label Entries

Purpose View the MPLS label table entry in the PE router for VPN label of 16.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 16**

```
mpls.0: 4012 destinations, 4012 routes (4012 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

16                 *[VPN/0] 20:27:51
                      to table vrf1.inet.0, Pop
```

Meaning The output indicates that the PE router will pop the VPN label from the packet and perform a second lookup of the destination address in the vrf1 routing table.

Verify VRF Table IPv4 Local Route Entries

Purpose View the VPN routing table entry in the PE device for route 7.0.25.0.

Action Issue the following command:

regress@host# **run show route table vrf1.inet.0 7.0.25.0**

```
vrf1.inet.0: 3347 destinations, 5881 routes (3347 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

7.0.25.0/30     *[BGP/170] 15:14:58, localpref 100
                   AS path: 1001 {1001} I, validation-state: unverified
                 > to 2.0.4.230 via xe-3/0/1.1
```

Meaning The output indicates that the PE device has learned route 7.0.25.0 from its directly connected CE device using EBGP and installed the route in its vrf1 routing table.

Verify Remote CE Device IPv6 Routes in the VRF Table of All PE Devices

Purpose View the remote CE device routes in the PE device VPN routing table.

Action Issue the following command:

regress@host# **run show route 2002::700:1900**

```
vrf1.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::700:1900/126 *[BGP/170] 12:55:31, localpref 100, from
10.255.50.71
                      AS path: 1001 {1001} I, validation-state: unverified
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-1
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-2
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-3
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-4
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-5
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-6
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-7
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-8
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-9
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-10
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-11
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-12
                    > to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-13
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-14
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-15
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-16
                    [BGP/170] 12:49:53, localpref 100, from 10.255.50.73
                      AS path: 1001 {1001} I, validation-state: unverified
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-1
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-2
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-3
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-4
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-5
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-6
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-7
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-8
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-9
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-10
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-11
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-12
                    > to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-13
                      to 1.0.0.13 via ae0.0, label-switched-path r4-to-r3-14
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-15
                      to 1.0.0.29 via ae1.0, label-switched-path r4-to-r3-16
```

Meaning The output indicates that the PE device has received and accepted remote CE device route 2002::700:1900 from the PE3 device through route reflectors 10.255.50.71 and 10.55.50.73 using MP-IBGP. It has installed the route in its vrf1 routing table and has selected the route received from address 10.255.50.71 as the best route. It is using LSP r4-to-r3-12 to reach the PE3 device.

Verify Remote CE Device IPv6 Routes

Purpose View the MPLS label values, BGP next-hop address, and LSP next-hop address associated with the VPN route installed in the PE device VPN routing table.

Action Issue the following command:

regress@host# **run show route 2002::700:1900 detail**

```
vrf1.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)

2002::700:1900/126 (2 entries, 1 announced)
      *BGP  Preference: 170/-101
            Route Distinguisher: 10.255.50.79:1
            Next hop type: Indirect
            Address: 0x1ffeaf90
            Next-hop reference count: 4039
            Source: 10.255.50.71 ->RR0
            Next hop type: Router, Next hop index: 1049278

            Next hop: 1.0.0.13 via ae0.0 weight 0x1, selected -> LSP
next-hop is P0
            Label-switched-path r4-to-r3-13 -> LSP
            Label operation: Push 498107 -> transport label
            Label TTL action: no-prop-ttl
            Session Id: 0x7332

            Protocol next hop: ::ffff:10.255.50.79 -> R3
            Push 16
            Composite next hop: 1ffeaf34 1066401 INH Session ID: 0x752e
            Indirect next hop: 1eff6100 1049441 INH Session ID: 0x752e
            State: <Secondary Active Int Ext ProtectionCand>
            Local AS:    70 Peer AS:    70
            Age: 15:46:19   Metric2: 33554428
            Validation State: unverified
            Task: BGP_70.10.255.50.71+64165
            Announcement bits (1): 1-KRT
            AS path: 1001 {1001} I (Originator)
            Cluster list:  10.255.50.71
            Originator ID: 10.255.50.79
            Communities: target:70:1 src-as:70:0 rt-
import:10.255.50.79:10
            Import Accepted Multipath
            VPN Label: 16 -> VPN label
            Localpref: 100
            Router ID: 10.255.50.71
            Primary Routing Table bgp.l3vpn-inet6.0
     BGP    Preference: 170/-101
            Route Distinguisher: 10.255.50.79:1
            Next hop type: Indirect
            Address: 0x1ffeaf90
            Next-hop reference count: 4039
            Source: 10.255.50.73 -> RR1
            Next hop type: Router, Next hop index: 1049278

            Next hop: 1.0.0.13 via ae0.0 weight 0x1, selected
            Label-switched-path r4-to-r3-13
            Label operation: Push 498107
            Label TTL action: no-prop-ttl
            Session Id: 0x7332
```

```
Protocol next hop: ::ffff:10.255.50.79
Push 16
Composite next hop: 1ffeaf34 1066401 INH Session ID: 0x752e
Indirect next hop: 1eff6100 1049441 INH Session ID: 0x752e
State: <Secondary NotBest Int Ext ProtectionCand>
Inactive reason: Not Best in its group - Update source
Local AS:    70 Peer AS:    70
Age: 15:40:41   Metric2: 33554428
Validation State: unverified
Task: BGP_70.10.255.50.73+53706
AS path: 1001 {1001} I (Originator)
Cluster list:  10.255.50.73
Originator ID: 10.255.50.79
Communities: target:70:1 src-as:70:0 rt-import:10.255.50.79:10
Import Accepted MultipathContrib
VPN Label: 16
Localpref: 100
Router ID: 10.255.50.73
Primary Routing Table bgp.l3vpn-inet6.0
```

Meaning VPN route 2002::700:1900 is installed in the vrf1 routing table in the PE device using a VPN label of 16 and an LSP label of 498107. The BGP next hop of the route is ::ffff:10.255.50.79 (PE3) and the LSP next hop is 1.0.0.13.

Verify Label-Switched Path Details

Purpose View the status of MPLS label-switched paths (LSPs).

Action Issue the following command:

regress@host# **run show mpls lsp name r4-to-r3-13 detail**

```
Ingress LSP: 1066 sessions
10.255.50.79
  From: 10.255.50.49, State: Up, ActiveRoute: 0, LSPname: r4-to-r3-13
  ActivePath:  (primary)
  Node/Link protection desired
  LSPtype: Static Configured, Penultimate hop popping
  LoadBalance: Random
  Autobandwidth
  MaxBW: 8Gbps
  AdjustTimer: 10800 secs AdjustThreshold: 5%
  Max AvgBW util: 57.6757Mbps, Bandwidth Adjustment in 9801 second(s).
  Overflow limit: 3, Overflow sample count: 0
  Underflow limit: 0, Underflow sample count: 3, Underflow Max AvgBW:
57.6757Mbps
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    Bandwidth: 122.652Mbps
    SmartOptimizeTimer: 180
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.13 S 1.0.0.10 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.71(flag=0x21) 1.0.0.13(flag=1 Label=549627)
10.255.50.79(flag=0x20) 1.0.0.10(Label=3)
Total 1 displayed, Up 1, Down 0
```

Meaning The output indicates that the ingress LSP from 10.255.50.49 (PE4) to 10.255.50.79 (PE3) is Up and using the primary path. The LSP is node-link protected and configured with autobandwidth. The primary path is going through 1.0.0.13.

Verify VRF Table IPv6 Local Route Entries

Purpose View the ingress interface (customer-facing interface) statistics in the local PE device.

Action Issue the following command:

regress@host# **run show interfaces xe-2/0/0 detail**

```
vrf1.inet6.0: 3353 destinations, 5890 routes (3353 active, 0 holddown,0
hidden)
+ = Active Route, - = Last Active, * = Both

2002::700:1900/126 *[BGP/170] 16:21:45, localpref 100
                     AS path: 1001 {1001} I, validation-state: unverified
                   > to 2002::200:4e6 via xe-3/0/1.1
```

Meaning The output indicates that the PE3 device has received and accepted route 2002::700:1900 from its directly connected CE device using BGP.

Verify Local PE-to-CE Device Interface Statistics

Purpose Verify that the VRF table contains entries for specific IPv6 local routes.

Action Use the following command to show that the VRF table contains an entry in router PE3 for IPv6 local route 2002::700:1900.

regress@host# **run show route 2002::700:1900**

```
Physical interface: xe-2/0/0, Enabled, Physical link is Up
  Interface index: 336, SNMP ifIndex: 9778, Generation: 339
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 64:87:88:04:51:8c, Hardware address:
64:87:88:04:51:8c
  Last flapped   : 2013-07-25 00:28:35 PDT (01:15:30 ago)
  Statistics last cleared: 2013-07-25 01:42:53 PDT (00:01:12 ago)
  Traffic statistics:
   Input  bytes  :          25457851727           2846029776 bps
   Output bytes  :          25454448480           2844155872 bps
   Input  packets:             33071090               461842 pps
   Output packets:             33062352               461335 pps
   IPv6 transit statistics:
    Input  bytes  :         12734295367
    Output bytes  :         12732263734
    Input  packets:            16428173
    Output packets:            16428592
  Ingress traffic statistics at Packet Forwarding Engine:
    Input  bytes  :          7307870278                   0 bps
    Input  packets:             9001987                   0 pps
    Drop   bytes  :                   0                   0 bps
    Drop   packets:                   0                   0 pps
  Ingress queues: 8 supported, 8 in use -> traffic flowing in all 8 queues
  Queue counters:  Queued packets  Transmitted packets   Dropped packets
    0 fc-be              8938214              8938214                 0
    1 fc-ef                 8435                 8435                 0
    2 fc-af1                8435                 8435                 0
```

```
    3 fc-nc                13159                13159                    0
    4 fc-af2                8436                 8436                    0
    5 fc-af3               8 436                 8436                    0
    6 fc-af4               84 36                 8436                    0
    7 fc-af-exta           843 6                 8436                    0
```

Meaning The output indicates that the PE device is receiving traffic from its directly connected CE device with different CoS markings. It is classifying the packets into eight forwarding classes and putting them into eight ingress queues.

Verify Local PE Core Interface Statistics

Purpose View the core-facing interface statistics in the PE device.

Action Issue the following command:

regress@host# **run show interfaces ae1 detail**

```
Physical interface: ae1, Enabled, Physical link is Up
  Interface index: 1163, SNMP ifIndex: 706, Generation: 2990
  Link-level type: Ethernet, MTU: 9192, Speed: 20Gbps, BPDU Error:
None, MAC-REWRITE Error: None, Loopback: Disabled, Source filtering:
Disabled, Flow control: Disabled, Minimum links needed: 1,
  Minimum bandwidth needed: 0
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Current address: 66:87:88:04:50:1e, Hardware address:
66:87:88:04:50:1e
  Last flapped   : 2013-07-10 04:55:34 PDT (22:04:06 ago)
  Statistics last cleared: 2013-07-11 01:48:34 PDT (01:11:06 ago)
  Traffic statistics:
   Input  bytes  :         538734405086           1086063696 bps
   Output bytes  :         594730594185           1105895168 bps
   Input  packets:            700585045               176520 pps
   Output packets:            773393759               179612 pps
   IPv6 transit statistics:
    Input  bytes  :                   0
    Output bytes  :                   0
    Input  packets:                   0
    Output packets:                   0
  Ingress queues: 8 supported, 8 in use
  Queue counters:  Queued packets  Transmitted packets   Dropped packets
    0 fc-be                     0                    0                  0
    1 fc-ef                     0                    0                  0
    2 fc-af1                    0                    0                  0
    3 fc-nc                     0                    0                  0
    4 fc-af2                    0                    0                  0
    5 fc-af3                    0                    0                  0
    6 fc-af4                    0                    0                  0
    7 fc-af-extra               0                    0                  0
  Egress queues: 8 supported, 8 in use -> Traffic flowing through 4 core
queues
  Queue counters: Queued packets  Transmitted packets   Dropped packets
    0 fc-be            770696052            770696052                  0
    1 fc-ef               616770               616770                  0
    2 fc-af1             2158695              2158695                  0
    3 fc-nc                43472                43472                  0
    4 fc-af2                   0                    0                  0
    5 fc-af3                   0                    0                  0
    6 fc-af4                   0                    0                  0
    7 fc-af-extra              0                    0                  0
```

```
  Queue number:         Mapped forwarding classes
    0                   fc-be
    1                   fc-ef
    2                   fc-af1
    3                   fc-nc
    4                   fc-af2
    5                   fc-af3
    6                   fc-af4
    7                   fc-af-extra
  Logical interface ae1.0 (Index 1388) (SNMP ifIndex 1136) (Generation
159860)
    Flags: SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :     700585045     176520  538734405086   1086063696
        Output:     773393759     179608  594730531341   1105891816
    Link: -> traffic flowing through all the member links
      xe-0/0/7.0
        Input :     345726131      89411  265867712252    550300344
        Output:     384123180      89364  295380510987    549979464
      xe-0/2/2.0
        Input :     354858914      87109  272866692834    535763352
        Output:     389270579      90244  299350020354    555912352
    LACP info:        Role     System    System      Port    Port  Port
                        priority      identifier  priority  number   key
      xe-0/0/7.0   Actor     127  64:87:88:04:57:c0   127       1     2
      xe-0/0/7.0   Partner   127  b0:a8:6e:2a:c7:c0   127       8     5
      xe-0/2/2.0   Actor     127  64:87:88:04:57:c0   127       4     2
      xe-0/2/2.0   Partner   127  b0:a8:6e:2a:c7:c0   127       5     5
    LACP Statistics:      LACP Rx     LACP Tx   Unknown Rx   Illegal Rx
      xe-0/0/7.0             4260        4260            0            0
      xe-0/2/2.0             4260        4260            0            0
    Marker Statistics:  Marker Rx     Resp Tx   Unknown Rx    Illegal R
      xe-0/0/7.0                0           0            0            0
      xe-0/2/2.0                0           0            0            0
    Protocol inet, MTU: 9178, Generation: 430580, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 1.0.0.28/30, Local: 1.0.0.30, Broadcast: 1.0.0.31,
Generation: 130527
    Protocol iso, MTU: 9175, Generation: 430581, Route table: 0
    Protocol inet6, MTU: 9178, Generation: 430582, Route table: 0
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 2002::100:1c/126, Local: 2002::100:1e
    Generation: 130529
      Addresses, Flags: Is-Preferred
        Destination: fe80::/64, Local: fe80::6687:88ff:fe04:501e
    Protocol mpls, MTU: 9166, Maximum labels: 3, Generation: 130531
    Protocol multiservice, MTU: Unlimited, Generation: 430583,
Route table: 0
    Generation: 430584, Route table: 0
      Policer: Input: __default_arp_policer__
```

Meaning The output indicates that the PE device is reclassifying the packets into four forwarding classes towards the core and putting them into four queues. The core-facing interface is an AE bundle with load balancing configured across the member links.

Verify LSP Statistics on the PE Device

Purpose View the MPLS LSP traffic statistics.

Action Issue the following command:

regress@host# **run show mpls lsp statistics name r4-to-r3-15**

```
Ingress LSP: 1066 sessions
To              From          State   Packets                Bytes LSPnam
10.255.50.79    10.255.50.49  Up    611090337   472633518910 r4-to-r3-15
Total 1 displayed, Up 1, Down 0
```

Issue the following command:

regress@host# **run show mpls lsp name r4-to-r3-13 statistics**

```
Ingress LSP: 1066 sessions
To              From          State   Packets              Bytes LSPname
10.255.50.79  10.255.50.49    Up     13462736    10355215918 r4-to-r3-13
Total 1 displayed, Up 1, Down 0
```

Meaning The output indicates that the PE device is using the LSPs r4-to-r3-15 and r4-to-r3-13 for traffic forwarding.

Verify Remote PE-to-CE Device Interface Statistics

Purpose View the egress interface (customer-facing interface) statistics in the remote PE device.

Action Issue the following command:

regress@host# **run show interfaces xe-3/0/1 detail**

```
Physical interface: xe-3/0/1, Enabled, Physical link is Up
  Interface index: 782, SNMP ifIndex: 10139, Generation: 2479
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: b0:a8:6e:fc:13:df, Hardware address:
b0:a8:6e:fc:13:df
  Last flapped   : 2013-07-22 15:11:16 PDT (2d 10:40 ago)
  Statistics last cleared: 2013-07-25 01:50:40 PDT (00:01:06 ago)
  Traffic statistics:
   Input  bytes  :          24040974389           2845190904 bps
   Output bytes  :          24043705592           2841548152 bps
   Input  packets:             31231049              461880 pps
   Output packets:             31224380              460925 pps
  IPv6 transit statistics:
    Input  bytes  :         12023416915
    Output bytes  :         12025732077
    Input  packets:            15515583
    Output packets:            15514887
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :           2599885596                    0 bps
   Input  packets:              3204899                    0 pps
   Drop   bytes  :                    0                    0 bps
   Drop   packets:                    0                    0 pps
```

```
 Egress queues: 8 supported, 8 in use
Queue counters:  Queued packets  Transmitted packets  Dropped packets
  0 fc-be              10696212             10696212                0
  1 fc-ef                 20317                20317                0
  2 fc-af1                28728                28728                0
  3 fc-nc                 21668                21668                0
  4 fc-af2                20317                20317                0
  5 fc-af3                20317                20317                0
  6 fc-af4                10224                10224                0
  7 fc-af-extra           20317                20317                0
```

Meaning The output indicates that the PE device is reclassifying the packets received from the core into eight forwarding classes and putting them into eight egress queues for delivery to its directly attached CE device.

Configuration Scenario: Layer 2 Circuits for Termination into Layer 3 VPNs

A Layer 2 circuit is a point-to-point Layer 2 connection transported using MPLS or other tunneling technology on a service provider network. A Layer 2 circuit is similar to a circuit cross-connect (CCC). However, where each CCC requires a separate dedicated LSP, Layer 2 circuits transport multiple virtual circuits (VCs) over a single shared label-switched path (LSP) tunnel between two provider edge (PE) routers.

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for this tested configuration:

- Before You Begin
- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Before You Begin

The configuration of Layer 2 circuits in this example requires the configuration of the “Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6” described earlier in this document.

Topology

The configuration of Layer 2 circuits in this scenario requires the same basic configuration topology found in the “Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6”, with the addition of an aggregation CE device. Figure 27 shows the topology tested for this scenario.

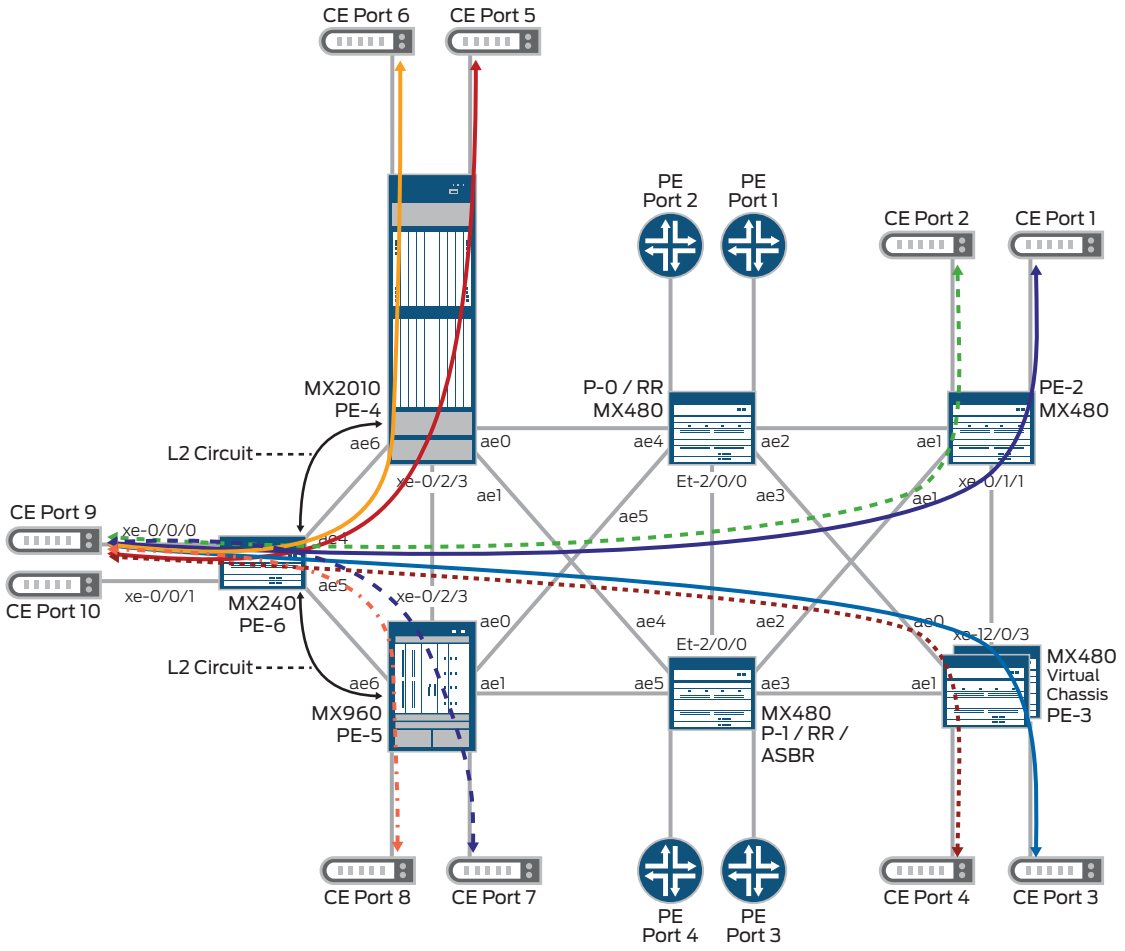

Figure 27: Layer 2 circuit configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*
- *Router PE-6 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this scenario, the following guidelines were followed when configuring the devices:

- Router PE6 functions as an access PE router that is connected to two external CE device ports.
- Router PE6 is configured with 500 primary and 500 backup l2circuit connections to routers PE4 and PE5.
- Routers PE4 and PE5 are each configured with 1,000 l2circuit connections to router PE6.
- In routers PE4 and PE5, the l2circuits are stitched to Layer 3 VPN using logical tunnel interfaces.
- CE device IP addresses are carried over through Layer 3 VPN to other remote PE devices.

Verification

The following sections provide methods for verifying this Layer 3 VPN configuration scenario along with sample command outputs:

- Verify Layer 2 Circuit Connections in the Access PE Device
- Verify Layer 2 Circuit Connections in Router PE4
- Verify Layer 2 Circuit Connections in Router PE5
- Verify CE Device Routes in Router PE4
- Verify CE Device Routes in Router PE5
- Verify CE Device Routes in Router PE2
- Verify Access Router PE6 Interface Statistics
- Verify Router PE4 Logical Tunnel Interface Statistics

Verify Layer 2 Circuit Connections in the Access PE Device

Purpose View the status of Layer 2 circuits on access PE device R6.

Action Issue the following command:

regress@host# **run show l2circuit connections**

```
Layer-2 Circuit Connections:
//500 primary l2circuits to PE4//
Neighbor: 10.255.50.49
    Interface             Type  St     Time last up          # Up trans
    xe-0/0/0.1(vc 1)      rmt   Up     Jul 22 13:41:49 2013           1
      Remote PE: 10.255.50.49, Negotiated control-word: Yes (Null)
      Incoming label: 546688, Outgoing label: 304736
      Negotiated PW status TLV: No
      Local interface: xe-0/0/0.1, Status: Up, Encapsulation: ETHERNET
    xe-0/0/0.3(vc 3)      rmt   Up     Jul 22 13:41:49 2013           1
      Remote PE: 10.255.50.49, Negotiated control-word: Yes (Null)
      Incoming label: 546608, Outgoing label: 304896
      Negotiated PW status TLV: No
      Local interface: xe-0/0/0.3, Status: Up, Encapsulation: ETHERNET
   xe-0/0/1.999(vc 999)   rmt   Up     Jul 22 13:41:49 2013           1
      Remote PE: 10.255.50.49, Negotiated control-word: Yes (Null)
      Incoming label: 546640, Outgoing label: 304800
      Negotiated PW status TLV: No
      Local interface: xe-0/0/1.999, Status: Up, Encapsulation: ETHERNET
//500 back-up l2circuits to PE4//
    xe-0/0/0.2(vc 2)          rmt   BK
    xe-0/0/0.4(vc 4)          rmt   BK
    xe-0/0/1.1000(vc 1000)    rmt   BK
```

```
//500 primary l2circuits to PE5//
Neighbor: 10.255.70.21
    Interface             Type  St     Time last up          # Up trans
    xe-0/0/0.2(vc 2)      rmt   Up     Jul 22 13:22:51 2013           1
      Remote PE: 10.255.70.21, Negotiated control-word: Yes (Null)
      Incoming label: 304704, Outgoing label: 303920
      Negotiated PW status TLV: No
      Local interface: xe-0/0/0.2, Status: Up, Encapsulation: ETHERNET
    xe-0/0/0.4(vc 4)      rmt   Up     Jul 22 13:22:51 2013           1
      Remote PE: 10.255.70.21, Negotiated control-word: Yes (Null)
      Incoming label: 304832, Outgoing label: 304048
      Negotiated PW status TLV: No
      Local interface: xe-0/0/0.4, Status: Up, Encapsulation: ETHERNET
   xe-0/0/1.1000(vc 1000) rmt   Up     Jul 22 13:22:48 2013           1
      Remote PE: 10.255.70.21, Negotiated control-word: Yes (Null)
      Incoming label: 303536, Outgoing label: 312720
      Negotiated PW status TLV: No
      Local interface: xe-0/0/1.1000, Status: Up, Encapsulation:
ETHERNET
//500 back-up l2circuits to PE5//
    xe-0/0/0.1(vc 1)          rmt   BK
    xe-0/0/0.3(vc 3)          rmt   BK
    xe-0/0/1.999(vc 999)      rmt   BK
```

Meaning The output indicates that this PE device has virtual circuits in the Up state with Layer 3 VPN PE devices 10.255.50.49 and 10.255.70.21.This PE device also has virtual circuits in the backup state with Layer 3 VPN PE devices 10.255.50.49 and 10.255.70.21.

Verify Layer 2 Circuit Connections in Router PE4

Purpose View the status of Layer 2 circuits in Layer 3 VPN PE device R4.

Action Issue the following command:

regress@host# **run show l2circuit connections**

```
//500 primary and back-up l2circuits to PE6//
Layer-2 Circuit Connections:
Neighbor: 10.255.51.168
    Interface             Type  St     Time last up         # Up trans
    lt-2/3/0.1(vc 1)      rmt   Up    Jul 22 19:04:41 2013  3 -> Primary
      Remote PE: 10.255.51.168, Negotiated control-word: Yes (Null)
      Incoming label: 304736, Outgoing label: 546688
      Negotiated PW status TLV: No
      Local interface: lt-2/3/0.1, Status: Up, Encapsulation: ETHERNET
    lt-2/3/0.2(vc 2)      rmt   OL -> back-up
    lt-2/3/0.999(vc 999) rmt   Up   Jul 22 13:41:50 2013   1 -> Primary
      Remote PE: 10.255.51.168, Negotiated control-word: Yes (Null)
      Incoming label: 304800, Outgoing label: 546640
      Negotiated PW status TLV: No
      Local interface: lt-2/3/0.999, Status: Up, Encapsulation: ETHERNET
    lt-2/3/0.1000(vc 1000)    rmt   OL -> back-up
```

Meaning The output indicates that device R4 has virtual circuits in the Up state and in the backup state with device R6. Virtual circuits from device R6 are terminating over logical tunnel (LT) interfaces in device R4.

Verify Layer 2 Circuit Connections in Router PE5

Purpose View the status of Layer 2 circuits in Layer 3 VPN PE device R5.

Action Issue the following command:

regress@host# **run show l2circuit connections**

```
//500 primary and back-up l2circuits to PE6//
Layer-2 Circuit Connections:
Neighbor: 10.255.51.168
    Interface             Type  St     Time last up          # Up trans
    lt-3/3/0.1(vc 1)      rmt   OL -> back-up
    lt-3/3/0.2(vc 2)      rmt   Up   Jul 22 13:22:48 2013  1 -> primary
      Remote PE: 10.255.51.168, Negotiated control-word: Yes (Null)
      Incoming label: 303920, Outgoing label: 304704
      Negotiated PW status TLV: No
      Local interface: lt-3/3/0.2, Status: Up, Encapsulation: ETHERNET
    lt-3/3/0.999(vc 999)  rmt   OL -> back-up
    lt-3/3/0.1000(vc 1000) rmt   Up   Jul 22 13:22:47 2013  1 -> primary
      Remote PE: 10.255.51.168, Negotiated control-word: Yes (Null)
      Incoming label: 312720, Outgoing label: 303536
      Negotiated PW status TLV: No
      Local interface: lt-3/3/0.1000, Status: Up, Encapsulation: ETHERNET
```

Meaning The output indicates that device R5 has virtual circuits in the Up state and in the backup state with R6. Virtual circuits from device R6 are terminating over logical tunnel (LT) interfaces in device R5.

Verify CE Device Routes in Router PE4

Purpose View the CE device routes received from the access PE device.

Action Issue the following command:

regress@host# **run show route 12.0.0.2/30**

```
vrf1.inet.0: 3348 destinations, 5883 routes (3348 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.0/30        *[Direct/0] 04:58:55
                    > via lt-2/3/0.2001
12.0.0.1/32        *[Local/0] 04:58:55
                       Local via lt-2/3/0.2001
```

Issue the following command:

regress@host# **run show route 12.0.0.6/30**

```
vrf2.inet.0: 3348 destinations, 5884 routes (3348 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.4/30   *[BGP/170] 10:19:32, localpref 100, from 10.255.50.71
               AS path: I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
```

```
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
                  [BGP/170] 10:19:31, localpref 100, from 10.255.50.73
                    AS path: I, validation-state: unverified
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                  > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                    to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0002/126**

```
vrf1.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:0/126    *[Direct/0] 06:16:55
                    > via lt-2/3/0.2001
2002::c00:1/128    *[Local/0] 06:16:55
                       Local via lt-2/3/0.2001
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0006/126**

```
vrf2.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:4/126 *[BGP/170] 11:33:53, localpref 100, from 10.255.50.71
                  AS path: I, validation-state: unverified
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

```
                                [BGP/170] 11:32:43, localpref 100, from 10.255.50.73
                                  AS path: I, validation-state: unverified
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                                > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                                  to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Meaning The output indicates that route 12.0.0.2 is learned directly over the l2circuit through the logical tunnel interface in the vrf1 routing table. Route 12.0.0.6 is learned from the other Layer 3 VPN PE device (R5) in the vrf2 routing table. This verifies that for the 12.0.0.2 route, the R4 device contains the primary l2circuit with access PE device R6 and, for the 12.0.0.6 route, the R4 device contains a backup l2circuit with access PE device R6. For this reason R4 is learning route 12.0.0.6 from Layer 3 VPN PE device R5, which has a primary l2circuit with the access PE device.

Verify CE Device Routes in Router PE5

Purpose View the CE device routes received from the access PE device.

Action Issue the following command:

```
regress@host# run show route 12.0.0.2/30
vrf1.inet.0: 3348 destinations, 5884 routes (3348 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.0/30  *[BGP/170] 06:11:05, localpref 100, from 10.255.50.71
                  AS path: I, validation-state: unverified
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-1
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-2
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-3
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-4
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-5
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-6
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-7
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-8
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-9
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-10
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-11
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-12
                > to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-13
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-14
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-15
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-16
                [BGP/170] 06:11:05, localpref 100, from 10.255.50.73
                  AS path: I, validation-state: unverified
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-1
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-2
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-3
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-4
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-5
```

```
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-6
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-7
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-8
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-9
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-10
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-11
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-12
                  > to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-13
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-14
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-15
                    to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-16
```

Issue the following command:

regress@host# **run show route 12.0.0.6/30**

```
vrf2.inet.0: 3348 destinations, 5883 routes (3348 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.4/30        *[Direct/0] 11:54:13
                    > via lt-3/3/0.2002
12.0.0.5/32        *[Local/0] 11:54:55
                      Local via lt-3/3/0.2002
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0002/126**

```
vrf1.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:0/126 *[BGP/170] 06:19:03, localpref 100, from 10.255.50.71
                  AS path: I, validation-state: unverified
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-1
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-2
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-3
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-4
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-5
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-6
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-7
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-8
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-9
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-10
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-11
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-12
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-13
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-14
                > to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-15
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-16
                [BGP/170] 06:19:03, localpref 100, from 10.255.50.73
                  AS path: I, validation-state: unverified
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-1
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-2
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-3
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-4
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-5
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-6
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-7
                  to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-8
```

```
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-9
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-10
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-11
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-12
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-13
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-14
                        > to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-15
                          to 1.0.0.41 via xe-0/2/3.0, label-switched-path r5-to-r4-16
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0006/126**

```
vrf2.inet6.0: 3354 destinations, 5893 routes (3354 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:4/126    *[Direct/0] 12:01:30
                     > via lt-3/3/0.2002
2002::c00:5/128    *[Local/0] 12:02:12
                        Local via lt-3/3/0.2002
```

Meaning The output indicates that the 12.0.0.6 route is learned directly over the l2circuit through the logical tunnel interface in the vrf2 routing table. The 12.0.0.2 route is learned from the other Layer 3 VPN PE device (R4) in the vrf1 routing table, verifying that for route 12.0.0.6, the R5 device contains the primary l2circuit with access PE device R6 and, for route 12.0.0.2, the R5 device contains a backup l2circuit with access PE device R6. For this reason, R5 is learning route 12.0.0.2 from Layer 3 VPN PE device R4, which has a primary l2circuit with the access PE device.

Verify CE Device Routes in Router PE2

Purpose View the CE routes in remote Layer 3 VPN PE devices R2 and R3.

Action Issue the following command:

regress@host# **run show route 12.0.0.2/30**

```
vrf1.inet.0: 3347 destinations, 5881 routes (3347 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.0/30  *[BGP/170] 06:14:14, localpref 100, from 10.255.50.71
                AS path: I, validation-state: unverified
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-1
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-2
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-3
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-4
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-5
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-6
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-7
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-8
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-9
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-10
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-11
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-12
              > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-13
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-14
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-15
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-16
               [BGP/170] 06:14:14, localpref 100, from 10.255.50.73
                AS path: I, validation-state: unverified
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-1
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-2
```

```
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-3
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-4
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-5
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-6
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-8
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-9
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-10
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-11
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-12
                  > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-13
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-14
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-15
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-16
```

Issue the following command:

regress@host# **run show route 12.0.0.6/30**

```
vrf2.inet.0: 3347 destinations, 5881 routes (3347 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.4/30  *[BGP/170] 11:41:17, localpref 100, from 10.255.50.71
                AS path: I, validation-state: unverified
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-1
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-2
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-3
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-4
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-5
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-6
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-7
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-8
              > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-9
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-10
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-11
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-12
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-13
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-14
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-15
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-16
               [BGP/170] 11:41:17, localpref 100, from 10.255.50.73
                AS path: I, validation-state: unverified
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-1
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-2
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-3
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-4
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-5
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-6
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-7
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-8
              > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-9
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-10
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-11
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-12
                to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-13
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-14
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-15
                to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-16
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0002/126**

```
vrf1.inet6.0: 3353 destinations, 5890 routes (3353 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:0/126 *[BGP/170] 06:20:23, localpref 100, from 10.255.50.71
                  AS path: I, validation-state: unverified
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-1
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-2
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-3
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-4
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-5
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-6
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-7
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-8
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-9
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-10
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-11
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-12
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-13
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-14
                > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-15
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-16
                [BGP/170] 06:20:23, localpref 100, from 10.255.50.73
                  AS path: I, validation-state: unverified
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-1
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-2
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-3
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-4
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-5
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-6
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-7
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-8
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-9
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-10
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-11
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-12
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-13
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-14
                > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r4-15
                  to 1.0.0.5 via ae0.0, label-switched-path r2-to-r4-16
```

Issue the following command:

regress@host# **run show route 2002:0000:0000:0000:0000:0000:0c00:0006/126**

```
vrf2.inet6.0: 3353 destinations, 5890 routes (3353 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c00:4/126 *[BGP/170] 11:47:14, localpref 100, from 10.255.50.71
                  AS path: I, validation-state: unverified
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-1
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-2
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-3
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-4
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-5
                  to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-6
```

```
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-8
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-9
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-10
                  > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-11
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-12
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-13
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-14
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-15
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-16
                  [BGP/170] 11:47:14, localpref 100, from 10.255.50.73
                    AS path: I, validation-state: unverified
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-1
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-2
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-3
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-4
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-5
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-6
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-8
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-9
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-10
                  > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-11
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-12
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r5-13
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-14
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-15
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r5-16
```

Meaning The output indicates that devices R2 and R3 are learning the CE routes through BGP from Layer 3 VPN PE devices R4 and R5.

Verify Access Router PE6 Interface Statistics

Purpose View the statistics for the access PE device ingress interface (connected to the access CE device).

Action Issue the following command:

regress@host# **run show interfaces xe-0/0/0 detail**

```
Physical interface: xe-0/0/0, Enabled, Physical link is Up
  Interface index: 1744, SNMP ifIndex: 5103, Generation: 1846
  Link-level type: Flexible-Ethernet, MTU: 9104, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 78:19:f7:52:c0:00, Hardware address:
78:19:f7:52:c0:00
  Last flapped   : 2013-07-22 13:18:55 PDT (14:03:02 ago)
  Statistics last cleared: 2013-07-23 03:20:08 PDT (00:01:49 ago)
  Traffic statistics:
   Input  bytes  :          39796860495           2911376064 bps
   Output bytes  :          39801135564           2912061928 bps
   Input  packets:             50508506               461845 pps
   Output packets:             50509781               461851 pps

   Logical interface xe-0/0/0.1 (Index 338) (SNMP ifIndex 5250)
(Generation 8432)
```

```
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.1 ] In(pop) Out(push
0x8100.1)
    Encapsulation: VLAN-CCC
    Traffic statistics:
     Input  bytes  :          39796849425
     Output bytes  :          39801007547
     Input  packets:             50508368
     Output packets:             50508372
    Local statistics:
     Input  bytes  :                    0
     Output bytes  :                    0
     Input  packets:                    0
     Output packets:                    0
    Transit statistics:
     Input  bytes  :          39796849425           2911373544 bps
     Output bytes  :          39801007547           2912050264 bps
     Input  packets:             50508368               461844 pps
     Output packets:             50508372               461842 pps
    Protocol ccc, MTU: 9104, Generation: 8690, Route table: 0
      Flags: Is-Primary
```

Meaning The output indicates that the access PE device is receiving and sending traffic from and to the access CE device on its ingress interfaces.

Verify Router PE4 Logical Tunnel Interface Statistics

Purpose View the logical tunnel (LT) interface statistics on the Layer 3 VPN PE device.

Action Issue the following command:

regress@host# **run show interfaces lt-2/3/0 detail**

```
Physical interface: lt-2/3/0, Enabled, Physical link is Up
  Interface index: 403, SNMP ifIndex: 7250, Generation: 2434
  Type: Logical-tunnel, Link-level type: Logical-tunnel, MTU: 9100,
Clocking: Unspecified, Speed: 10000mbps
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Link type      : Unspecified
  Link flags     : None
  Physical info  : 13
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 64:87:88:04:52:1f, Hardware address:
64:87:88:04:52:1f
  Alternate link address: Unspecified
  Last flapped   : 2013-07-22 13:38:12 PDT (13:46:55 ago)
  Statistics last cleared: 2013-07-22 15:47:29 PDT (11:37:38 ago)
  Traffic statistics:
   Input  bytes  :       17000673256454           5739484000 bps
   Output bytes  :       17002494373680           5739093304 bps
   Input  packets:          21881994705               923694 pps
   Output packets:          21882926146               923694 pps
   IPv6 transit statistics:
    Input  bytes  :       4228410186478
    Output bytes  :       4181885228921
    Input  packets:          5456000760
    Output packets:          5395990651

  Logical interface lt-2/3/0.1 (Index 335) (SNMP ifIndex 7255)
(Generation 35551)
    Flags: SNMP-Traps 0x4000 Encapsulation: Ethernet-CCC
    Traffic statistics:
```

```
    Input  bytes  :        8546662690691
    Output bytes  :        8608569708847
    Input  packets:          10902086100
    Output packets:          10980719235
   Local statistics:
    Input  bytes  :                    0
    Output bytes  :                    0
    Input  packets:                    0
    Output packets:                    0
   Transit statistics:
    Input  bytes  :        8546662690691           2896468456 bps
    Output bytes  :        8608569708847           2894228288 bps
    Input  packets:          10902086100              461834 pps
    Output packets:          10980719235              461834 pps
   Protocol ccc, MTU: 9100, Generation: 80231, Route table: 0
     Flags: Is-Primary
```

Issue the following command:

regress@host# **run show interfaces lt-2/3/0.2001 detail**

```
  Logical interface lt-2/3/0.2001 (Index 1617) (SNMP ifIndex 8729)
(Generation 38027)
    Flags: SNMP-Traps 0x4000 Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :        8390701749196
     Output bytes  :        8387786967285
     Input  packets:          10897424316
     Output packets:          10893698783
     IPv6 transit statistics:
      Input  bytes  :       4196758326942
      Output bytes  :       4195323909594
      Input  packets:          5415160679
      Output packets:          5413331456
    Local statistics:
     Input  bytes  :               43064
     Output bytes  :              246892
     Input  packets:                 609
     Output packets:                2734
    Transit statistics:
     Input  bytes  :        8390701706132           2845740296 bps
     Output bytes  :        8387786720393           2843335288 bps
     Input  packets:          10897423707              461844 pps
     Output packets:          10893696049              461843 pps
     IPv6 transit statistics:
      Input  bytes  :       4196758326942
      Output bytes  :       4195323909594
      Input  packets:          5415160679
      Output packets:          5413331456
```

Meaning The output indicates that the Layer 3 VPN PE device is receiving and sending traffic from the access PE device on its LT interfaces.

Configuration Scenario: Direct Internet Access Service

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for this tested Direct Internet Access (DIA) service configuration:

- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Topology

Figure 28 shows the topology tested for the DIA service.

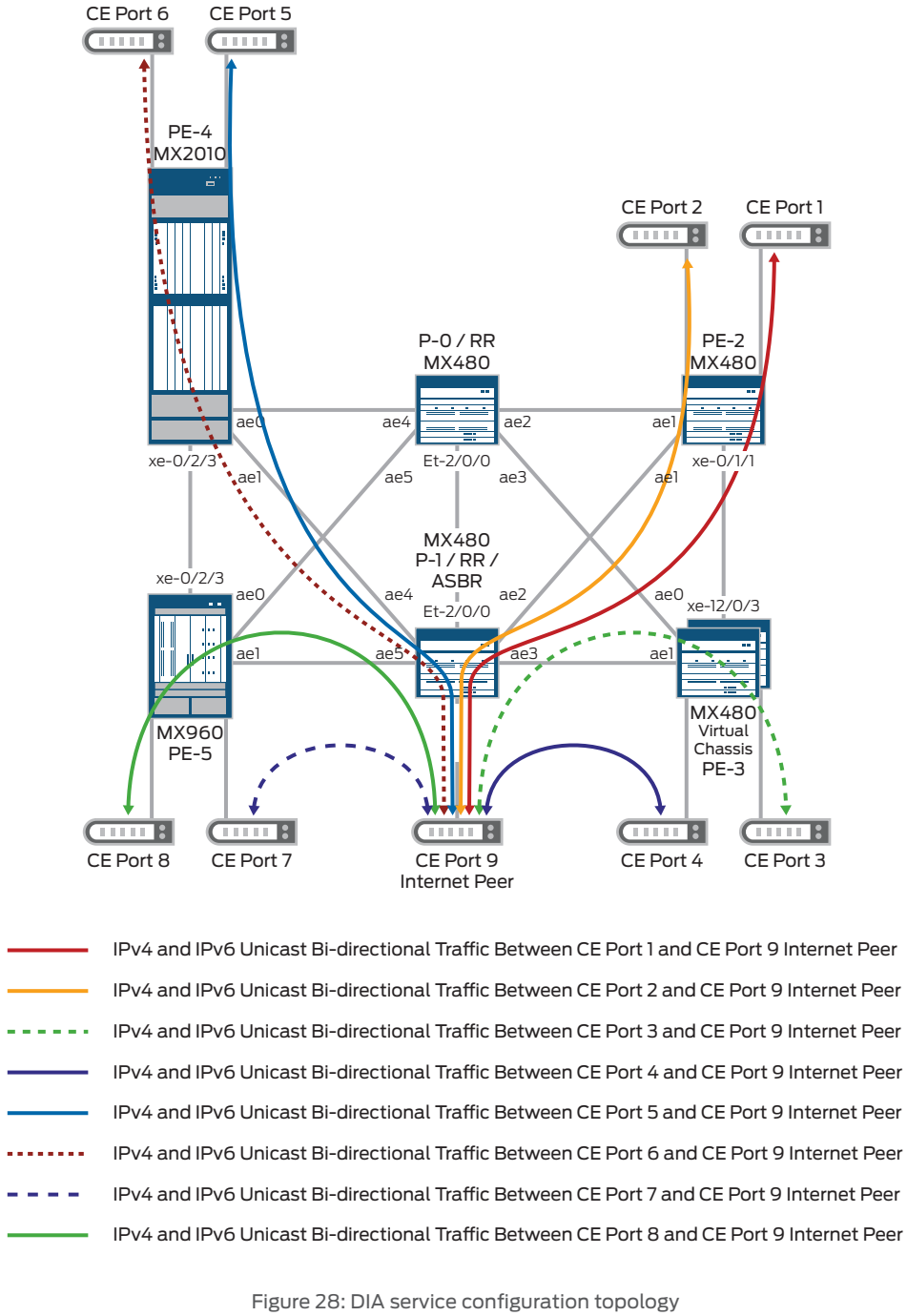

Figure 28: DIA service configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this scenario, the following guidelines were followed when configuring the devices:

- The DIA service described in this example provides connectivity between upstream providers (global Internet) and DIA customers accessing the service provider network.
- The AS boundary router peers with Internet customers and learns 250,000 IPv4 and IPv6 prefixes.
- Each PE device is configured with 24 EBGP IPv4 and IPv6 customer peers and learns 6,000 IPv4 and IPv6 prefixes from each customer port.
- Each PE device is also configured with static routes for each customer.
- Different import and export policies are created and applied for customers, peers, and upstream devices.
- Two different CoS profiles are configured for DIA customers; one with both ingress and egress policers, and one with an ingress policer and per-unit queuing/shaping performed upon egress.
- This example was tested with both IPv4 and IPv6 traffic sent from all customer ports to the Internet peering port and from the Internet peering port to all customer ports. All packets were a fixed 256 bytes and utilized a best-effort traffic class. Traffic was sent at 1 percent line rate.

Verification

The following sections provide methods for verifying this DIA configuration scenario along with sample command outputs:

- Verify BGP Peering with the Customer in PE Device
- Verify Local Customer Routes in PE Device
- Verify the Remote Customer Routes
- Verify Local Static Customer Routes in the PE Device
- Verify Remote Static Customer Routes in PE Devices from Other PE Devices
- Verify Peer Routes Are Received from the AS boundary router (P1)
- Verify Customers Receiving Only Default Routes from the PE Device
- Verify Customers Receiving Full Internet Routing Table from the PE Device (Customer Routes)
- Verify Customers Receiving Full Internet Routing Table from the PE Device (Peer Routes)
- Verify Customers Receiving Full Internet Routing Table from the PE Device (Direct Routes)
- Verify Customers Receiving Full Internet Routing Table from the PE Device (Static Routes)
- Verify the Existence of Customer Routes in the AS Boundary Router from Other PE Devices
- Verify the Existence of Direct Routes in the AS Boundary Router from Other PE Devices
- Verify the Existence of Static Customer Routes in the AS Boundary Router from Other PE Devices
- Verify BGP Peering with a Peer in the AS Boundary Router Device
- Verify Peer Routes in the AS Boundary Router Device
- Verify Aggregate Routes in the AS Boundary Router
- Verify Peers Receiving Customer Routes from the AS Boundary Router
- Verify Peers Receiving Aggregate Routes from the AS Boundary Router

Verify BGP Peering with the Customer in PE Device

Purpose View the status of EBGP neighborship with the customer CE device.

Action Issue the following command:

regress@host# **run show bgp neighbor 2.0.13.54**

```
Peer: 2.0.13.54+179 AS 10001   Local: 2.0.13.53+54802 AS 70
  Type: External    State: Established    Flags: <Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ pol-export-default-route ] Import: [ pol-reject-martians
pol-import-customer-default pol-final ]
  Options: <Preference RemovePrivateAS AddressFamily PeerAS PrefixLimit
Refresh>
  Options: <BfdEnabled>
  Address families configured: inet-unicast
  Holdtime: 90 Preference: 170
  Number of flaps: 1
  Last flap event: Closed
  Peer ID: 2.0.13.54     Local ID: 10.255.50.49     Active Holdtime: 90
  Keepalive Interval: 30         Group index: 3    Peer index: 1
  BFD: enabled, down
  Local Interface: xe-2/0/0.125
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table inet.0 Bit: 10001
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              500
    Received prefixes:            500
    Accepted prefixes:            500
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 29   Sent 26   Checked 29
  Input messages:   Total 7821  Updates 2  Refreshes 0  Octets 153665
  Output messages:  Total 8058  Updates 0  Refreshes 0  Octets 153266
```

regress@nanopdt# **run show bgp neighbor 2002::200:d36**

```
Peer: 2002::200:d36+44795 AS 10001 Local: 2002::200:d35+179 AS 70
  Type: External    State: Established    Flags: <Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ pol-export-default-route ] Import: [ pol-reject-martians
pol-import-customer-default pol-final ]
  Options: <Preference RemovePrivateAS AddressFamily PeerAS PrefixLimit
Refresh>
  Options: <BfdEnabled>
  Address families configured: inet6-unicast
  Holdtime: 90 Preference: 170
  Number of flaps: 1
  Last flap event: Closed
```

```
  Peer ID: 2.0.13.54    Local ID: 10.255.50.49   Active Holdtime: 90
  Keepalive Interval: 30    Group index: 3    Peer index: 23
  BFD: enabled, down
  Local Interface: xe-2/0/0.125
  NLRI for restart configured on peer: inet6-unicast
  NLRI advertised by peer: inet6-unicast
  NLRI for this session: inet6-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table inet6.0 Bit: 40001
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              500
    Received prefixes:            500
    Accepted prefixes:            500
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 6    Sent 29   Checked 6
  Input messages:  Total 1928   Updates 3   Refreshes 0  Octets 45328
  Output messages: Total 1982   Updates 0   Refreshes 0  Octets 37698
  Output Queue[3]: 0
```

Issue the following command:

regress@host# **run show bgp neighbor 2002::200:d36**

```
Peer: 2002::200:d36+44795 AS 10001 Local: 2002::200:d35+179 AS 70
  Type: External    State: Established    Flags: <Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ pol-export-default-route ] Import: [ pol-reject-martians
pol-import-customer-default pol-final ]
  Options: <Preference RemovePrivateAS AddressFamily PeerAS PrefixLimit
Refresh>
  Options: <BfdEnabled>
  Address families configured: inet6-unicast
  Holdtime: 90 Preference: 170
  Number of flaps: 1
  Last flap event: Closed
  Peer ID: 2.0.13.54     Local ID: 10.255.50.49    Active Holdtime: 90
  Keepalive Interval: 30         Group index: 3    Peer index: 23
  BFD: enabled, down
  Local Interface: xe-2/0/0.125
  NLRI for restart configured on peer: inet6-unicast
  NLRI advertised by peer: inet6-unicast
  NLRI for this session: inet6-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table inet6.0 Bit: 40001
    RIB State: BGP restart is complete
    Send state: in sync
```

```
    Active prefixes:              500
    Received prefixes:            500
    Accepted prefixes:            500
    Suppressed due to damping:    0
    Advertised prefixes:          0
  Last traffic (seconds): Received 6    Sent 29   Checked 6
  Input messages:  Total 1928   Updates 3   Refreshes 0   Octets 45328
  Output messages: Total 1982   Updates 0   Refreshes 0   Octets 37698
  Output Queue[3]: 0
```

Meaning The output indicates that the EBGPv4 and v6 neighborships with the customer peers are established, and this PE device has received and accepted 500 IPv4 and IPv6 prefixes from the customer.

Verify Local Customer Routes in PE Device

Purpose View the routes received from the directly attached customer devices.

Action Issue the following command:

```
regress@host# run show route 8.1.212.192/30 detail
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
8.1.212.192/30 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 26588
                Address: 0x1cfc7288
                Next-hop reference count: 2005
                Source: 2.0.13.54
                Next hop: 2.0.13.54 via xe-2/0/0.125, selected
                Session Id: 0x9b31
                State: <Active Ext>
                Local AS:    70 Peer AS: 10001
                Age: 16:03:33
                Validation State: unverified
                Task: BGP_10001.2.0.13.54+179
                Announcement bits (4): 0-KRT 7-Resolve tree 4 8-BGP_RT_
Background 9-Resolve tree 5
                AS path: 10001 {10001} I
                Communities: 70:3000 70:20000 -> customer community and
region-id community
                Accepted
                Localpref: 100
                Router ID: 2.0.13.54
```

Issue the following command:

```
regress@host# run show route 3002::801:d4c0 detail
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
3002::801:d4c0/126 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 34812
                Address: 0x1e7c8438
                Next-hop reference count: 2005
                Source: 2002::200:d36
                Next hop: 2002::200:d36 via xe-2/0/0.125, selected
                Session Id: 0x9b34
                State: <Active Ext>
                Local AS:    70 Peer AS: 10001
                Age: 16:08:17
                Validation State: unverified
```

```
                    Task: BGP_10001.2002::200:d36+44795
                    Announcement bits (4): 0-KRT 5-BGP_RT_Background
6-Resolve tree 6 7-Resolve tree 7
                    AS path: 10001 {10001} I
                    Communities: 70:3000 70:20000
                    Accepted
                    Localpref: 100
                    Router ID: 2.0.13.54
```

Meaning The output indicates that the PE device has received and accepted the IPv4 and IPv6 routes from the customer and has added the customer and region-id communities to the routes.

Verify the Remote Customer Routes

Purpose View the remote customer routes received from other PE devices in the network.

Action Issue the following command:

regress@host# **run show route 8.2.144.64/30**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
+ = Active Route, - = Last Active, * = Both

8.2.144.64/30 *[BGP/170] 08:13:08, localpref 100, from 10.255.50.71
                AS path: 10001 {10001} I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
               [BGP/170] 08:15:12, localpref 100, from 10.255.50.73
                AS path: 10001 {10001} I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Issue the following command:

regress@host# **run show route 3002::802:9040**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

3002::802:9040/126 *[BGP/170] 08:12:10, localpref 100, from 10.255.50.71
                     AS path: 10001 {10001} I, validation-state: unverified
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                   > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
                    [BGP/170] 08:10:51, localpref 100, from 10.255.50.73
                     AS path: 10001 {10001} I, validation-state: unverified
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                   > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                     to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Meaning The output indicates that the PE device is learning IPv4 and IPv6 remote customer routes from other PE devices in the network using BGP.

Verify Local Static Customer Routes in the PE Device

Purpose View the static customer routes for the directly attached customers.

Action Issue the following command:

regress@host# **run show route protocol static 16.0.0.0/24 detail**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
16.0.0.0/24 (1 entry, 1 announced)
        State: <FlashAll>
        *Static Preference: 5
                Next hop type: Router, Next hop index: 27675
                Address: 0xc89265c
                Next-hop reference count: 2005
                Next hop: 2.0.11.22 via xe-2/0/1.125, selected
                Session Id: 0x91e
                State: <Active Int Ext>
                Local AS:    70
                Age: 9:34:49
                Validation State: unverified
                Task: RT
                Announcement bits (5): 0-KRT 2-LDP 7-Resolve tree 4
8-BGP_RT_Background 9-Resolve tree 5
                AS path: I
                Communities: 70:3000
```

Issue the following command:

regress@host# **run show route 3002::1000:0/120 detail**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
3002::1000:0/120 (1 entry, 1 announced)
        *Static Preference: 5
                Next hop type: Router, Next hop index: 35217
                Address: 0xc892610
                Next-hop reference count: 2005
                Next hop: 2002::200:b16 via xe-2/0/1.125, selected
                Session Id: 0x936
                State: <Active Int Ext>
                Local AS:    70
                Age: 9:39:34
                Validation State: unverified
                Task: RT
                Announcement bits (4): 0-KRT 5-BGP_RT_Background
6-Resolve tree 6 7-Resolve tree 7
                AS path: I
                Communities: 70:3000
```

Meaning The output indicates that the PE device has installed the static customer routes in its routing table with the customer communities.

Verify Remote Static Customer Routes in PE Devices from Other PE Devices

Purpose View the static routes for the remote customers received from other PE devices in the network.

Action Issue the following command:

regress@host# **run show route 17.0.0.0/24**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
+ = Active Route, - = Last Active, * = Both

17.0.0.0/24  *[BGP/170] 09:37:35, localpref 100, from 10.255.50.71
                AS path: I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
               [BGP/170] 09:37:35, localpref 100, from 10.255.50.73
                AS path: I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Issue the following command:

regress@host# **run show route 3002::17.0.0.0/120**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both
3002::1100:0/120 *[BGP/170] 09:40:13, localpref 100, from 10.255.50.71
                AS path: I, validation-state: unverified
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
              > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
```

```
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
                        [BGP/170] 09:40:13, localpref 100, from 10.255.50.73
                          AS path: I, validation-state: unverified
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-1
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-2
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-3
                        > to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-4
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-5
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-6
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-7
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-8
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-9
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-10
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-11
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-12
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-13
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-14
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-15
                          to 1.0.0.42 via xe-0/2/3.0, label-switched-path r4-to-r5-16
```

Meaning The output indicates that the PE device has received the remote customer static routes from other PE devices in the network.

Verify Peer Routes Are Received from the AS Boundary Router (P1)

Purpose View the Internet peer routes received from the AS boundary router.

Action Issue the following command:

regress@host# **run show route 9.0.0.0/30 detail**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
9.0.0.0/30 (2 entries, 1 announced)
   BGP    Preference: 170/-101
          Next hop type: Indirect
          Address: 0x2dcef10c
          Next-hop reference count: 1250025
          Source: 10.255.50.73
          Next hop type: Router, Next hop index: 840
          Next hop: 1.0.0.29 via ae1.0, selected
          Session Id: 0x9b0e
          Protocol next hop: 2.0.0.122
          Indirect next hop: 5fabe4e0 1773394 INH Session ID: 0xa1e6
          State: <Active Int Ext>
          Local AS: 70  Peer AS: 70  Age: 19:25:42  Metric2: 16777224
          Validation State: unverified
          Task: BGP_70.10.255.50.73+61053
          Announcement bits (4): 0-KRT 7-Resolve tree 4 8-BGP_RT_
Background 9-Resolve tree 5
          AS path: 4000 {4000} I
```

```
            Communities: 70:2000 70:20000 71:666 -> Peer community and
Region-id community
            Accepted Multipath
            Localpref: 100
            Router ID: 10.255.50.73
    BGP     Preference: 170/-101
            Next hop type: Indirect
            Address: 0x2dcef10c
            Next-hop reference count: 1250025
            Source: 10.255.50.71
            Next hop type: Router, Next hop index: 840
            Next hop: 1.0.0.29 via ae1.0, selected
            Session Id: 0x9b0e
            Protocol next hop: 2.0.0.122
          Indirect next hop: 5fabe4e0 1773394 INH Session ID: 0xa1e6
            State: <NotBest Int Ext>
          Inactive reason: Not Best in its group - Cluster list length
            Local AS:    70 Peer AS:    70
            Age: 19:24:07   Metric2: 16777224
            Validation State: unverified
            Task: BGP_70.10.255.50.71+52876
            AS path: 4000 {4000} I (Originator)
            Cluster list:  10.255.50.71
            Originator ID: 10.255.50.73
            Communities: 70:2000 70:20000 71:666
            Accepted MultipathContrib
            Localpref: 100
            Router ID: 10.255.50.71
```

Meaning The output indicates that the PE device has received the peer routes from the AS boundary router with the peer and region-id communities added.

Verify Customers Receiving Only Default Routes from the PE Device

Purpose View the routes advertised by the PE device to the customer BGP neighbor 2.0.11.2.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.11.22 extensive**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
* 0.0.0.0/0 (1 entry, 1 announced)
 BGP group customers-default-only type External
     Nexthop: Self
     AS path: [70] I
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:0b16 detail**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
* ::/0 (1 entry, 1 announced)
 BGP group customers-default-only type External
     Nexthop: Self
    AS path: [70] I
```

Meaning The output indicates that the PE device is advertising only the default route to the customers.

Verify Customers Receiving Full Internet Routing Table from the PE Device (Customer Routes)

Purpose View the routes advertised by the PE device to customer BGP neighbor 2.0.11.38.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.11.38 8.1.119.0/30 detail**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
* 8.1.119.0/30 (1 entry, 1 announced)
 BGP group customers-full-view type External
     Nexthop: Self
     AS path: [70] 10001 {10001} I
     Communities: 70:20000
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:0b26 3002::801:7700 detail**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
* 3002::801:7700/126 (1 entry, 1 announced)
 BGP group customers-full-view type External
     Nexthop: Self
     AS path: [70] 10001 {10001} I
     Communities: 70:20000
```

Meaning The output indicates that the remote customer routes are advertised by the PE device to its directly attached customers.

Verify Customers Receiving Full Internet Routing Table from the PE Device (Peer Routes)

Purpose View the routes advertised by the PE device to customer BGP neighbor 2.0.11.38.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.11.38 9.0.0.0/30 detail**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
* 9.0.0.0/30 (2 entries, 1 announced)
 BGP group customers-full-view type External
     Nexthop: Self
     AS path: [70] 4000 {4000} I
     Communities: 70:20000 71:666
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:0b26 3002::900:0 detail**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
* 3002::900:0/126 (2 entries, 1 announced)
 BGP group customers-full-view type External
     Nexthop: Self
     AS path: [70] 4000 {4000} I
     Communities: 70:20000 71:666
```

Meaning The output indicates that the remote Internet peer routes are advertised by the PE device to its directly attached customers.

Verify Customers Receiving Full Internet Routing Table from the PE Device (Direct Routes)

Purpose View the routes advertised by the PE device to customer BGP neighbor 2.0.11.38.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.11.38 2.0.0.0/8**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 2.0.11.20/30              Self                                       I
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:0b26 2002::2.0.0.0/104**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 2002::200:b14/126         Self                                       I
```

Meaning The output indicates that the direct routes for the remote customers are advertised by the PE device to its directly attached customers.

Verify Customers Receiving Full Internet Routing Table from the PE Device (Static Routes)

Purpose View the static customer routes advertised by the PE device to its directly connected customers.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.11.38 16.0.0.0/8**

```
inet.0: 309116 destinations, 595194 routes (309106 active, 0 holddown,
10 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 16.0.0.0/24               Self                                       I
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:0b26 3002::16.0.0.0/104**

```
inet6.0: 308251 destinations, 594357 routes (308251 active, 0 holddown,
0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 3002::1000:0/120          Self                                       I
```

Meaning The output indicates that the PE device is advertising both the local and remote customer static routes to its directly connected customers.

Verify the Existence of Customer Routes in the AS Boundary Router from Other PE Devices

Purpose View that the remote customer routes are received by the AS boundary router from other PE devices in the network.

Action Issue the following command:

regress@host# **run show route 8.1.212.192/30**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown,
9 hidden)
+ = Active Route, - = Last Active, * = Both

8.1.212.192/30 *[BGP/170] 08:19:34, localpref 100, from 10.255.50.49
                AS path: 10001 {10001} I, validation-state: unverified
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
```

```
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
                  [BGP/170] 08:17:04, localpref 100, from 10.255.50.71
                    AS path: 10001 {10001} I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Issue the following command:

regress@host# **run show route 3002::801:7700/126**

```
inet6.0: 308250 destinations, 356397 routes (308250 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both
3002::801:7700/126 *[BGP/170] 09:16:13, localpref 100, from 10.255.50.49
                    AS path: 10001 {10001} I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
                  [BGP/170] 09:10:57, localpref 100, from 10.255.50.71
```

```
                    AS path: 10001 {10001} I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Meaning The output indicates that the AS boundary router device has received the remote customer routes from other PE devices in the network.

Verify the Existence of Direct Routes in the AS Boundary Router from Other PE Devices

Purpose View the remote customer direct routes in the AS boundary router.

Action Issue the following command:

regress@host# **run show route 2.0.11.20/30**

```
2.0.11.20/30 *[BGP/170] 00:48:01, localpref 100, from 10.255.50.49
                    AS path: I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
                  [BGP/170] 00:48:01, localpref 100, from 10.255.50.71
                    AS path: I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
```

```
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Issue the following command:

regress@host# **run show route 2002::200:b14/126**

```
2002::200:b14/126  *[BGP/170] 01:18:37, localpref 100, from 10.255.50.49
                      AS path: I, validation-state: unverified
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
                    [BGP/170] 01:18:37, localpref 100, from 10.255.50.71
                      AS path: I, validation-state: unverified
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                      to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Meaning The output indicates that the AS boundary router device has received and installed the direct routes from other PE devices in the network.

Verify the Existence of Static Customer Routes in the AS Boundary Router from Other PE Devices

Purpose View the remote customer static routes in the AS boundary router device.

Action Issue the following command:

regress@host# **run show route 16.0.0.0/8**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown, 9
hidden)
+ = Active Route, - = Last Active, * = Both
16.0.0.0/24  *[BGP/170] 09:20:19, localpref 100, from 10.255.50.49
                AS path: I, validation-state: unverified
              > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
               [BGP/170] 09:20:19, localpref 100, from 10.255.50.71
                AS path: I, validation-state: unverified
              > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Issue the following command:

regress@host# **run show route 3002::1000:0/120**

```
3002::1000:0/120  *[BGP/170] 09:22:11, localpref 100, from 10.255.50.49
                    AS path: I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
```

```
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
                  [BGP/170] 09:22:11, localpref 100, from 10.255.50.71
                    AS path: I, validation-state: unverified
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-1
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-2
                  > to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-3
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-4
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-5
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-6
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-7
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-8
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-9
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-10
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-11
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-12
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-13
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-14
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-15
                    to 1.0.0.30 via ae4.0, label-switched-path r1-to-r4-16
```

Meaning The output indicates that the AS boundary router device has received and installed the remote customer static routes from other PE devices in the network.

Verify BGP Peering with a Peer in the AS Boundary Router Device

Purpose View the status of BGP neighborship with the Internet peer in AS boundary router.

Action Issue the following command:

regress@host# **run show bgp neighbor 2.0.0.122**

```
Peer: 2.0.0.122+32853 AS 4000  Local: 2.0.0.121+179 AS 70
  Type: External    State: Established    Flags: <Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ pol-export-peer-or-upstream pol-final ] Import: [ pol-prefix-
allow-4000-bh pol-import-peer-default pol-final ]
  Options: <Preference RemovePrivateAS AddressFamily PeerAS PrefixLimit
LocalAS Refresh>
  Options: <BfdEnabled AcceptRemoteNextHop>
  Address families configured: inet-unicast
  Holdtime: 90 Preference: 170 Local AS: 70 Local System AS: 70
  Number of flaps: 1
  Last flap event: Closed
  Peer ID: 2.0.0.122     Local ID: 10.255.50.73    Active Holdtime: 90
  Keepalive Interval: 30         Group index: 5    Peer index: 0
  BFD: enabled, down
  Local Interface: xe-0/1/1.4000
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
 Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
```

```
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table inet.0 Bit: 10002
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              250005
    Received prefixes:            250005
    Accepted prefixes:            250005
    Suppressed due to damping:    0
    Advertised prefixes:          48004
  Last traffic (seconds): Received 19   Sent 29   Checked 49
  Input messages:  Total 387  Updates 311  Refreshes 0   Octets 1269840
  Output messages: Total 1555  Updates 1477  Refreshes 0  Octets 321820
  Output Queue[0]: 0
```

Issue the following command:

regress@host# **run show bgp neighbor 2002:0000:0000:0000:0000:0000:0200:007a**

```
Peer: 2002::200:7a+33126 AS 4000 Local: 2002::200:79+179 AS 70
  Type: External    State: Established    Flags: <Sync RSync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ pol-export-peer-or-upstream pol-final ] Import: [ pol-prefix-
allow-4000-bh pol-import-peer-default pol-final ]
  Options: <Preference RemovePrivateAS AddressFamily PeerAS PrefixLimit
LocalAS Refresh>
  Options: <BfdEnabled AcceptRemoteNextHop>
  Address families configured: inet6-unicast
  Holdtime: 90 Preference: 170 Local AS: 70 Local System AS: 70
  Number of flaps: 1
  Last flap event: Closed
  Peer ID: 2.0.0.122     Local ID: 10.255.50.73     Active Holdtime: 90
  Keepalive Interval: 30         Group index: 6    Peer index: 0
  BFD: enabled, down
  Local Interface: xe-0/1/1.4000
  NLRI for restart configured on peer: inet6-unicast
  NLRI advertised by peer: inet6-unicast
  NLRI for this session: inet6-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Peer does not support Receiver functionality
  Peer does not support 4 byte AS extension
  Peer does not support Addpath
  Table inet6.0 Bit: 40002
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              250005
    Received prefixes:            250005
    Accepted prefixes:            250005
    Suppressed due to damping:    0
    Advertised prefixes:          48004
  Last traffic (seconds): Received 15   Sent 28   Checked 15
  Input messages:  Total 1142 Updates 1065  Refreshes 0  Octets 4341029
  Output messages: Total 939  Updates 862   Refreshes 0  Octets 893763
  Output Queue[3]: 0
```

Meaning The output indicates that the AS boundary router device has established BGP neighborship with the Internet peer. The AS boundary router device has received and installed 250,000 prefixes from the peer and advertised 48,000 prefixes to the peer.

Verify Peer Routes in the AS Boundary Router Device

Purpose View the routes received from the Internet peer in the AS boundary router device.

Action Issue the following command:

regress@host# **run show route 9.0.0.0/30 detail**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown,
9 hidden)
9.0.0.0/30 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 1006
                Address: 0xa0f336c
                Next-hop reference count: 1000020
                Source: 2.0.0.122
                Next hop: 2.0.0.122 via xe-0/1/1.4000, selected
                Session Id: 0x1e8
                State: <Active Ext>
                Local AS:    70 Peer AS:  4000
                Age: 9:23:04
                Validation State: unverified
                Task: BGP_4000.2.0.0.122+35220
                Announcement bits (5): 0-KRT 2-RT 8-Resolve tree 4
9-BGP_RT_Background 10-Resolve tree 5
                AS path: 4000 {4000} I
                AS path: Recorded
                Communities: 70:2000 70:20000 71:666 -> peer community
                Accepted
                Localpref: 100
                Router ID: 2.0.0.122
```

Issue the following command:

regress@host# **run show route 3002::900:0/126 detail**

```
inet6.0: 308250 destinations, 356397 routes (308250 active, 0 holddown,
0 hidden)
3002::900:0/126 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 1042
                Address: 0xb741ef0
                Next-hop reference count: 1000020
                Source: 2002::200:7a
                Next hop: 2002::200:7a via xe-0/1/1.4000, selected
                Session Id: 0x1e9
                State: <Active Ext>
                Local AS:    70 Peer AS:  4000
                Age: 9:24:50
                Validation State: unverified
                Task: BGP_4000.2002::200:7a+35292
                Announcement bits (5): 0-KRT 2-RT 7-BGP_RT_Background
8-Resolve tree 6 9-Resolve tree 7
                AS path: 4000 {4000} I
                AS path: Recorded
                Communities: 70:2000 70:20000 71:666
                Accepted
                Localpref: 100
                Router ID: 2.0.0.122
```

Meaning The output indicates that the AS boundary router device has received and accepted the routes from the peer and added the peer and region-id communities.

Verify Aggregate Routes in the AS Boundary Router

Purpose View the aggregate routes configured in the AS boundary router for the customer static prefixes.

Action Issue the following command:

regress@host# **run show route protocol aggregate**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown,
9 hidden)
+ = Active Route, - = Last Active, * = Both

14.0.0.0/8          *[Aggregate/130] 10:16:23
                       Reject
15.0.0.0/8          *[Aggregate/130] 10:16:23
                       Reject
16.0.0.0/8          *[Aggregate/130] 10:16:23
                       Reject
17.0.0.0/8          *[Aggregate/130] 10:16:23
                       Reject

inet6.0: 308250 destinations, 356397 routes (308250 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both

3002::e00:0/104     *[Aggregate/130] 10:16:24
                       Reject
3002::f00:0/104     *[Aggregate/130] 10:16:24
                       Reject
3002::1000:0/104    *[Aggregate/130] 10:16:24
                       Reject
3002::1100:0/104    *[Aggregate/130] 10:16:24
                       Reject
```

Meaning The output indicates that the AS boundary router device has installed the aggregate routes.

Verify Peers Receiving Customer Routes from the AS Boundary Router

Purpose View the customer routes advertised by the AS boundary router device to the Internet peer.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.0.122**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown,
9 hidden)
  Prefix              Nexthop          MED     Lclpref    AS path
* 8.0.0.0/30          Self                                10001 {10001}
I
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:007a**

```
inet6.0: 308250 destinations, 356397 routes (308250 active, 0 holddown,
0 hidden)
  Prefix              Nexthop          MED     Lclpref    AS path
* 3002::800:0/126     Self                                10001 {10001}
I
```

Meaning The output indicates that the AS boundary router device is advertising the remote customer routes to the Internet peer.

Verify Peers Receiving Aggregate Routes from the AS Boundary Router

Purpose View the aggregate routes advertised by the AS boundary router device to the Internet peer.

Action Issue the following command:

regress@host# **run show route advertising-protocol bgp 2.0.0.122 16.0.0.0/8 detail**

```
inet.0: 309116 destinations, 357237 routes (309107 active, 0 holddown,
9 hidden)
* 16.0.0.0/8 (1 entry, 1 announced)
 BGP group v4-gtsm type External
     Nexthop: Self
     AS path: [70] I (LocalAgg)
```

Issue the following command:

regress@host# **run show route advertising-protocol bgp 2002:0000:0000:0000:0000:0000:0200:007a 3002::e00:0/104 detail**

```
inet6.0: 308250 destinations, 356397 routes (308250 active, 0 holddown,
0 hidden)
* 3002::e00:0/104 (1 entry, 1 announced)
 BGP group v6-gtsm type External
     Nexthop: Self
     AS path: [70] I (LocalAgg)
```

Meaning The output indicates that the AS boundary router device is advertising the aggregate routes to the Internet peer.

CHAPTER 7 Advanced Connectivity Services

- Configuration Scenario: Next-Generation Multicast VPN Services for IPv4 and IPv6 Traffic
- Configuration Scenario: Next-Generation Multicast VPN Services Over a Layer 3 VPN Network Where a PE Router Acts as a P Router
- Configuration Scenario: Next-Generation Multicast VPN Services Over a Layer 3 VPN Network With Extranet Multicast

Configuration Scenario: Next-Generation Multicast VPN Services for IPv4 and IPv6 Traffic

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for an added NG MVPN configuration that delivers both IPv4 and IPv6 traffic over an existing Layer 3 VPN business edge topology:

- Before You Begin
- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Before You Begin

This example is used to deploy multicast service to Layer 3 VPN customers. The configuration of next-generation MVPN in this example requires the configuration of the "Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6" described earlier in this document.

Topology

Figure 29 shows the topology tested for the added NG MVPN IPv4/v6 service.

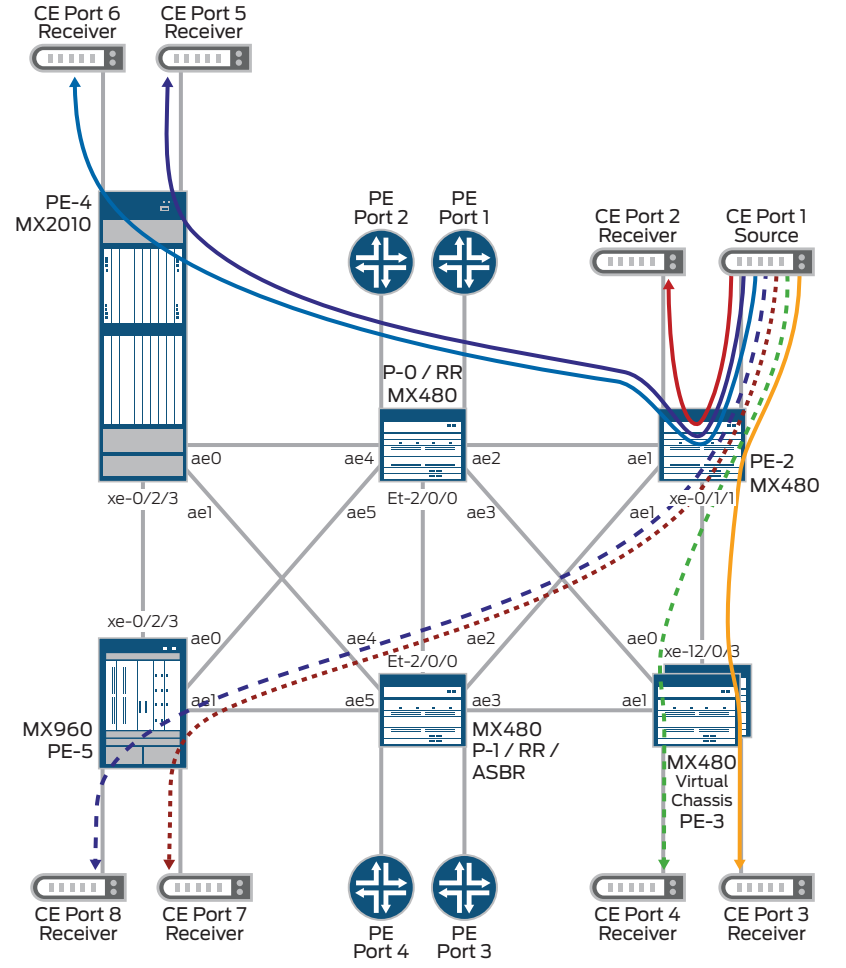

Figure 29: NG MVPN IPv4 and IPv6 configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this configuration scenario, the following guidelines were followed when configuring the devices:

- This multicast service uses next-generation MVPN in conjunction with RSVP point-to-multipoint (P2MP) provider tunnels.
- Multicast sources and receivers are both part of the same VPN. Each VPN is configured with two sources and two receivers. Each source is sending traffic to 100 multicast groups.
- IGMPv3 and MLDv2 are used as the receiver side protocols.
- PIM-SM is used in the customer instance.
- Anycast RP is configured to implement RP redundancy.
- The P2MP selective provider tunnel is configured for 10 groups. This configuration provides a separate multicast channel for each of these groups. A P2MP inclusive provider tunnel is configured for the remaining 90 groups.
- Bandwidth allocation and link protection is configured for each P2MP tunnel.
- IPv4 and IPv6 traffic from all sources to all receivers utilize random frame sizes ranging from 74 to 1,500 bytes and a line rate of 5 Mbps. All traffic is sent to the BE queue.

Verification

The following sections provide methods for verifying this Layer 3 VPN configuration scenario along with sample command outputs:

- Verify MP-IBGP Peering with the route reflector
- Verify the Existence of an RSVP P2MP Inclusive Provider Tunnel in the Sender PE Device
- Verify an RSVP P2MP Selective Provider Tunnel in the Sender PE Device
- Verify MVPN Routes in the VRF Table in the Sender PE Device
- Verify Multicast Route in Sender PE Device
- Verify RSVP P2MP Sessions in Sender PE Device
- Verify RSVP Session
- Verify MPLS Table in P1
- Verify MPLS Table in Receiver PE4
- Verify that a Second Route Lookup Occurs in vrf1.inet.1 in Receiver PE4
- Verify the Multicast Route Table in Receiver PE4
- Verify C-PIM Database in Receiver PE4
- Verify Sender PE2 Interface Traffic Statistics
- Verify Receiver PE5 Interface Traffic Statistics
- Verify MPLS P2MP LSP Statistics in Sender PE2

Verify MP-IBGP Peering with the Route Reflector

Purpose View the status of BGP neighborship with the route reflector in the PE device.

Action Issue the following command:

regress@host# **run show bgp neighbor 30.0.0.10**

```
Peer: 30.0.0.10+179 AS 70      Local: 30.0.0.12+64284 AS 70
  Type: Internal    State: Established    Flags: <ImportEval Sync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ next-hop-self ]
  Options: <Preference LocalAddress AddressFamily Multipath Rib-group
Refresh>
  Options: <MultipathAs AutheKeyChain BfdEnabled>
  Authentication key chain: chain1
  Authentication algorithm: hmac-sha-1-96
  Address families configured: inet-mvpn inet6-mvpn
  Local Address: 30.0.0.12 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 10.255.50.71   Local ID: 10.255.50.77   Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 1
  BFD: enabled, up
  NLRI for restart configured on peer: inet-mvpn inet6-mvpn
  NLRI advertised by peer: inet-mvpn inet6-mvpn
  NLRI for this session: inet-mvpn inet6-mvpn
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-mvpn inet6-mvpn
  NLRI of received end-of-rib markers: inet-mvpn inet6-mvpn
  NLRI of all end-of-rib markers sent: inet-mvpn inet6-mvpn
  Peer supports 4 byte AS extension (peer-as 70)
  Peer does not support Addpath
```

Meaning The output indicates that the BGP neighborship with the route reflector is established with **inet-mvpn inet6-mvpn** address families. All BGP message exchanges are authenticated. BFD over BGP is up.

Verify the Existence of an RSVP P2MP Inclusive Provider Tunnel in the Sender PE Device

Purpose View the status of an RSVP point-to-multipoint inclusive provider tunnel for vrf1 in the sender PE device.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp name 10.255.50.77:1:mvpn:vrf1 detail**

```
Ingress LSP: 64 sessions
P2MP name: 10.255.50.77:1:mvpn:vrf1, P2MP branch count: 3 -> 3 sub-LSP's

10.255.70.21 -> PE5
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.70.21:10.255.50.77:1:mvpn:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mvpn:vrf1
  Link protection desired -> Link protection configured
  LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps -> bandwidth configured
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
```

```
    Reoptimization in 48 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.21 S 1.0.0.34 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.73(flag=0x21) 1.0.0.21(flag=1 Label=445289)
10.255.70.21(flag=0x20) 1.0.0.34(Label=16)

10.255.50.49 -> PE4
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.50.49:10.255.50.77:1:mvpn:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mvpn:vrf1
  Link protection desired
  LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                   State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 22 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.21 S 1.0.0.30 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.73(flag=0x21) 1.0.0.21(flag=1 Label=445289)
10.255.50.49(flag=0x20) 1.0.0.30(Label=17)

10.255.50.79 -> PE3
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.50.79:10.255.50.77:1:mvpn:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mvpn:vrf1
  Link protection desired
LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                   State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 10 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
16777214)
 1.0.0.38 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.79(flag=0x20) 1.0.0.38(Label=16)
Total 3 displayed, Up 3, Down 0
```

Meaning The output indicates that the point-to-multipoint LSP is up. It has three sub-LSPs connected to three receiver PE devices. All sub-LSPs have link protection, are using the primary path, and have allocated a configured bandwidth of 1,000 Kbps.

Verify an RSVP P2MP Selective Provider Tunnel in the Sender PE Device

Purpose View the status of an RSVP point-to-multipoint selective provider tunnel for vrf1 in the sender PE device.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp name 10.255.50.77:1:mv1:vrf1 detail**

```
Ingress LSP: 84 sessions
P2MP name: 10.255.50.77:1:mv1:vrf1, P2MP branch count: 3

10.255.50.49
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.50.49:10.255.50.77:1:mv1:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mv1:vrf1
  Link protection desired
  LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 33 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.5 S 1.0.0.14 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.71(flag=0x21) 1.0.0.5(flag=1 Label=678332)
10.255.50.49(flag=0x20) 1.0.0.14(Label=17)

10.255.50.79
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.50.79:10.255.50.77:1:mv1:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mv1:vrf1
  Link protection desired
  LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                    State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 32 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
16777214)
 1.0.0.38 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.79(flag=0x20) 1.0.0.38(Label=16)

10.255.70.21
  From: 10.255.50.77, State: Up, ActiveRoute: 0, LSPname:
10.255.70.21:10.255.50.77:1:mv1:vrf1
  ActivePath:  (primary)
  P2MP name: 10.255.50.77:1:mv1:vrf1
```

```
  Link protection desired
  LSPtype: Dynamic Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
 *Primary                   State: Up
    Priorities: 7 0
    Bandwidth: 1000kbps
    OptimizeTimer: 50
    SmartOptimizeTimer: 180
    Reoptimization in 32 second(s).
    Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric:
33554428)
 1.0.0.5 S 1.0.0.18 S
    Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node
10=SoftPreempt 20=Node-ID):
          10.255.50.71(flag=0x21) 1.0.0.5(flag=1 Label=678332)
10.255.70.21(flag=0x20) 1.0.0.18(Label=16)
Total 3 displayed, Up 3, Down 0
```

Meaning The output indicates that the P2MP LSP is up. It has three sub-LSPs connecting to three receiver PE devices. All sub-LSPs have link protection, they are using the primary path, and have allocated a configured bandwidth of 1,000 Kbps.

Verify MVPN Routes in the VRF Table in the Sender PE Device

Purpose View the MVPN routes in the sender PE device's vrf1 MVPN routing table.

Action Issue the following command:

regress@host# **run show route table vrf1.mvpn.0**

```
vrf1.mvpn.0: 344 destinations, 677 routes (344 active, 72 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.50.49:1:10.255.50.49/240 -> Type 1 AD route received from PE4
                    *[BGP/170] 02:52:58, localpref 100, from 30.0.0.10
                       AS path: I, validation-state: unverified
                       to 1.0.0.5 via ae0.0
                     > to 1.0.0.21 via ae1.0
                     [BGP/170] 02:52:30, localpref 100, from 30.0.0.11
                       AS path: I, validation-state: unverified
                       to 1.0.0.5 via ae0.0
                     > to 1.0.0.21 via ae1.0
1:10.255.50.77:1:10.255.50.77/240 -> Local Type 1 AD route generated by
this PE
                    *[MVPN/70] 02:57:22, metric2 1
                       Indirect
1:10.255.50.79:1:10.255.50.79/240 -> Type 1 AD route received from PE3
                    *[BGP/170] 02:52:21, localpref 100, from 30.0.0.10
                       AS path: I, validation-state: unverified
                     > to 1.0.0.38 via xe-0/1/1.0
                     [BGP/170] 02:52:20, localpref 100, from 30.0.0.11
                       AS path: I, validation-state: unverified
                     > to 1.0.0.38 via xe-0/1/1.0
1:10.255.70.21:1:10.255.70.21/240 -> Type 1 AD route received from PE5
                    *[BGP/170] 02:52:58, localpref 100, from 30.0.0.10
                       AS path: I, validation-state: unverified
                       to 1.0.0.5 via ae0.0
                     > to 1.0.0.21 via ae1.0
                     [BGP/170] 02:52:30, localpref 100, from 30.0.0.11
```

```
                        AS path: I, validation-state: unverified
                        to 1.0.0.5 via ae0.0
                      > to 1.0.0.21 via ae1.0
3:10.255.50.77:1:32:2.0.1.158:32:230.0.0.10:10.255.50.77/240 -> Type 3
AD route generated by local PE for S-PMSI
                    *[MVPN/70] 00:07:44, metric2 1
                        Indirect
4:3:10.255.50.77:1:32:2.0.1.158:32:230.0.0.10:10.255.50.77:10.255.50.49
/240 -> Type 4 AD route received from receiver PE4
                    *[BGP/170] 00:07:43, localpref 100, from 30.0.0.10
                        AS path: I, validation-state: unverified
                        to 1.0.0.5 via ae0.0
                      > to 1.0.0.21 via ae1.0
                      [BGP/170] 00:07:43, localpref 100, from 30.0.0.11
                        AS path: I, validation-state: unverified
                        to 1.0.0.5 via ae0.0
                      > to 1.0.0.21 via ae1.0
5:10.255.50.77:1:32:2.0.1.158:32:230.0.0.1/240 -> Type 5 AD route
generated by local PE which is the source PE as well as the RP
                    *[PIM/105] 00:07:45
                        Multicast (IPv4) Composite
6:10.255.50.77:1:70:32:11.0.0.3:32:230.0.0.1/240 -> Type 6 C-multicast
route generated by the local PE since it is the receiver PE as well
                    *[PIM/105] 02:54:36
                        Multicast (IPv4) Composite
7:10.255.50.77:1:70:32:2.0.1.158:32:230.0.0.1/240 -> Type 7 C-multicast
route received from the receiver PE's
                    *[MVPN/70] 00:07:45, metric2 1
                        Multicast (IPv4) Composite
                      [PIM/105] 00:07:45
                        Multicast (IPv4) Composite
                      [BGP/170] 00:07:44, localpref 100, from 30.0.0.10
                        AS path: I, validation-state: unverified
                      > to 1.0.0.5 via ae0.0
                      [BGP/170] 00:07:44, localpref 100, from 30.0.0.11
                        AS path: I, validation-state: unverified
                      > to 1.0.0.21 via ae1.0
```

Meaning The output indicates that the PE device has generated a Type 1 route and it has received a Type 1 route from every other PE device. This PE device has generated a Type 3 route and received a Type 4 route from other PE devices for selective provider tunnels. Because this PE device is the RP, it has generated Type 5 routes. Because this PE device has local receivers attached, it has generated a Type 6 route and it has received Type 7 routes from other PE devices for the remote receivers.

Verify Multicast Route in Sender PE Device

Purpose View the vrf1 multicast routing table in the sender PE device.

Action Issue the following command:

regress@host# **run show multicast route instance vrf1 detail**

```
Instance: vrf1 Family: INET

Group: 230.0.0.1
    Source: 2.0.1.158/32
    Upstream interface: xe-1/2/1.63
    Downstream interface list:
        xe-0/1/1.0 ae1.0 xe-1/2/0.63
    Session description: Unknown
```

```
    Statistics: 6 kBps, 8 pps, 218982 packets
    Next-hop ID: 1949430
    Upstream protocol: MVPN

Group: 230.0.0.10
    Source: 2.0.1.158/32
    Upstream interface: xe-1/2/1.63
    Downstream interface list:
        xe-0/1/1.0 ae0.0 xe-1/2/0.63
    Session description: Unknown
    Statistics: 6 kBps, 8 pps, 221486 packets
    Next-hop ID: 1949516
    Upstream protocol: MVPN
```

Issue the following command:

regress@host# **run show multicast route inet6 extensive instance vrf1**

```
Instance: vrf1 Family: INET6

Group: ff15::1
    Source: 2002::200:19e/128
    Upstream interface: xe-1/2/1.63
    Downstream interface list:
        xe-0/1/1.0 ae1.0 xe-1/2/0.63
    Session description: Unknown
    Statistics: 6 kBps, 8 pps, 179880 packets
    Next-hop ID: 1949525
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 06:22:21

Group: ff15::a
    Source: 2002::200:19e/128
    Upstream interface: xe-1/2/1.63
    Downstream interface list:
        ae0.0 xe-1/2/0.63
    Session description: Unknown
    Statistics: 6 kBps, 8 pps, 181435 packets
    Next-hop ID: 1144572
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 06:25:40
```

Meaning The output indicates that the PE device has installed the routes for groups 230.0.0.1,230.0.0.10,ff15 ::1,ff15::a. Source 2.0.1.158/32 and 2002::200:19e/128 are the active sources sending traffic to these groups. The source is reachable through upstream interface xe-1/2/1.63, and receivers are reachable through the downstream interfaces xe-0/1/1.0, ae1.0, ae0.0, and xe-1/2/0.63. The upstream protocol is MVPN.

Verify RSVP P2MP Sessions in Sender PE Device

Purpose View the status of RSVP point-to-multipoint inclusive and selective provider tunnels.

Action Issue the following command for the inclusive tunnel:

regress@host# **run show rsvp session p2mp ingress name 10.255.50.77:1:mvpn:vrf1**

```
P2MP name: 10.255.50.77:1:mvpn:vrf1, P2MP branch count: 3
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.70.21    10.255.50.77    Up       0  1 SE       -   776283
10.255.70.21:10.255.50.77:1:mvpn:vrf1
10.255.50.49    10.255.50.77    Up       0  1 SE       -   776283
10.255.50.49:10.255.50.77:1:mvpn:vrf1
10.255.50.79    10.255.50.77    Up       0  1 SE       -       16
10.255.50.79:10.255.50.77:1:mvpn:vrf1
```

Issue the following command for the selective tunnel:

regress@host# **run show rsvp session p2mp ingress name 10.255.50.77:1:mv1:vrf1**

```
P2MP name: 10.255.50.77:1:mv1:vrf1, P2MP branch count: 3
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.50.49    10.255.50.77    Up       0  1 SE       -   616815
10.255.50.49:10.255.50.77:1:mv1:vrf1
10.255.70.21    10.255.50.77    Up       0  1 SE       -   616815
10.255.70.21:10.255.50.77:1:mv1:vrf1
10.255.50.79    10.255.50.77    Up       0  1 SE       -       16
10.255.50.79:10.255.50.77:1:mv1:vrf1
```

Meaning The output indicates that the tunnels are up. There are three sub-LSPs for three PE devices and this PE device has received the labels.

Verify RSVP Session

Purpose Verify the RSVP point-to-multipoint session in the P router and receiver PE device.

Action Issue the following command for the RSVP session in P1:

regress@host# **run show rsvp session p2mp name 10.255.50.77:1:mvpn:vrf1**

```
P2MP name: 10.255.50.77:1:mvpn:vrf1, P2MP branch count: 2
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.70.21    10.255.50.77    Up       0  1 SE  776283       16
10.255.70.21:10.255.50.77:1:mvpn:vrf1
10.255.50.49    10.255.50.77    Up       0  1 SE  776283       17
10.255.50.49:10.255.50.77:1:mvpn:vrf1
```

Issue the following command for the RSVP session in P0:

regress@host# **run show rsvp session p2mp name 10.255.50.77:1:mv1:vrf1**

```
P2MP name: 10.255.50.77:1:mv1:vrf1, P2MP branch count: 2
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.50.49    10.255.50.77    Up       0  1 SE  616815       17
10.255.50.49:10.255.50.77:1:mv1:vrf1
10.255.70.21    10.255.50.77    Up       0  1 SE  616815       16
10.255.70.21:10.255.50.77:1:mv1:vrf1
```

Issue the following command for the RSVP session in P4:

regress@host# **run show rsvp session p2mp name 10.255.50.77:1:mvpn:vrf1**

```
P2MP name: 10.255.50.77:1:mvpn:vrf1, P2MP branch count: 1
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.50.49    10.255.50.77    Up       0  1 SE      17        -
10.255.50.49:10.255.50.77:1:mvpn:vrf1
```

Meaning The output indicates that the P router has two sub-LSPs for each P2MP tunnel. Each sub-LSP has the same labelin values but with the different labelout values. The receiver PE device has one P2MP LSP with the labelin value matching the labelout value in the P router.

Verify MPLS Table in P1

Purpose View the MPLS label table entry for the label 776283.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 776283**

```
mpls.0: 5343 destinations, 5343 routes (5329 active, 14 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

776283             *[RSVP/7/2] 06:55:55, metric 1
                    > to 1.0.0.34 via ae5.0, Swap 16
                      to 1.0.0.30 via ae4.0, Swap 17
```

Meaning The output indicates that the P router has two entries using the swap operation. This device uses label 16 for PE5 and label 17 for PE4. The entries also have different outgoing interfaces.

Verify MPLS Table in Receiver PE4

Purpose View the MPLS label table entry in PE router for label 17.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 17**

```
mpls.0: 5045 destinations, 5045 routes (5045 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

17                 *[VPN/0] 21:38:39
                      to table vrf1.inet.0, Pop
```

Meaning The output indicates that the PE device pops the label and performs a second lookup in the vrf1.inet.1 table.

Verify that a Second Route Lookup Occurs in vrf1.inet.1 in Receiver PE4

Purpose View the route entry in the vrf1.inet.1 table.

Action Issue the following command:

regress@host# **run show route table vrf1.inet.1 detail**

```
vrf1.inet.1: 103 destinations, 103 routes (103 active, 0 holddown, 0
hidden)

230.0.0.1.2.0.1.158/64 (1 entry, 1 announced)
        *MVPN   Preference: 70
                Next hop type: Multicast (IPv4) Composite, Next hop
index: 1050161
                Address: 0xc5b457c
                Next-hop reference count: 400
                State: <Active Int>
                Age: 11:52:34
```

```
                                Validation State: unverified
                                Task: mvpn global task
                                Announcement bits (1): 0-KRT
                                AS path: I
```

Meaning The output indicates that the PE device has installed the route in the vrf1.inet.1 table with a composite next hop where the second lookup occurs.

Verify the Multicast Route Table in Receiver PE4

Purpose View the multicast routes in the vrf1 routing table.

Action Issue the following command:

regress@host# **run show multicast route instance vrf1 extensive**

```
Instance: vrf1 Family: INET

Group: 230.0.0.1
    Source: 2.0.1.158/32
    Upstream interface: lsi.69633
    Downstream interface list:
        xe-2/0/0.63 xe-2/0/1.63
    Session description: Unknown
    Statistics: 7 kBps, 8 pps, 12525 packets
    Next-hop ID: 1858048
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:26:38

Instance: vrf1 Family: INET6

Group: ff15::1
    Source: 2002::200:19e/128
    Upstream interface: lsi.73729
    Downstream interface list:
        xe-2/0/1.63 xe-2/0/0.63
    Session description: Unknown
    Statistics: 6 kBps, 8 pps, 181948 packets
    Next-hop ID: 1048779
    Upstream protocol: MVPN
```

Meaning The output indicates that the PE device has installed the multicast routes for groups 230.0.0.1 and ff15::1 and sources 2.0.1.158/32 and 2002::200:19e/128. Upstream interfaces are the label-switched interfaces (LSIs) facing the core. The upstream protocol is MVPN and there are two downstream receivers on interfaces xe-2/0/0.63 and xe-2/0/1.63.

Verify C-PIM Database in Receiver PE4

Purpose View the PIM join states in vrf1 table for the customer receiver IGMP joins.

Action Issue the following command:

regress@host# **run show pim join extensive instance vrf1**

```
Instance: PIM.vrf1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 230.0.0.1
    Source: *
    RP: 11.0.0.3 -> Anycast RP address in PE2 and PE3
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 15:54:41
    Downstream neighbors:
        Interface: xe-2/0/1.63
            2.0.10.29 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:54:44 Time since last Join: 15:54:44
    Downstream neighbors:
        Interface: xe-2/0/0.63
            2.0.12.61 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:54:44 Time since last Join: 15:54:44
    Number of downstream interfaces: 2
```

Issue the following command:

regress@host# **run show pim join inet6 extensive instance vrf1**

```
Instance: PIM.vrf1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff15::1
    Source: *
    RP: 2002:2::b00:1
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 16:04:06
    Downstream neighbors:
        Interface: xe-2/0/1.63
            fe80::6687:8800:3f04:518d State: Join Flags: SRW  Timeout:
Infinity
            Uptime: 16:04:07 Time since last Join: 16:04:07
    Downstream neighbors:
        Interface: xe-2/0/0.63
            fe80::6687:8800:3f04:518c State: Join Flags: SRW  Timeout:
Infinity
            Uptime: 16:04:04 Time since last Join: 16:04:04
    Number of downstream interfaces: 2
```

Meaning The output indicates that the PE device has installed PIM (*,G) join states in response to receiving the IGMP reports from the receivers. Upstream is the anycast RP address and downstream are the interfaces connecting the receivers. The upstream protocol is BGP.

Verify Sender PE2 Interface Traffic Statistics

Purpose View the ingress interface statistics in the sender PE device.

Action Issue the following command:

regress@host# **run show interfaces xe-1/2/1 detail**

```
Physical interface: xe-1/2/1, Enabled, Physical link is Up
  Interface index: 1808, SNMP ifIndex: 12053, Generation: 1872
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: b0:a8:6e:76:29:ef, Hardware address:
b0:a8:6e:76:29:ef
  Last flapped   : 2013-07-22 13:25:57 PDT (2d 14:17 ago)
  Statistics last cleared: 2013-07-25 03:42:19 PDT (00:01:20 ago)
  Traffic statistics:
   Input  bytes  :        48795921         4816048 bps
   Output bytes  :          394034           35256 bps
   Input  packets:           71713             838 pps
   Output packets:            4819              53 pps
   IPv6 transit statistics:
    Input  bytes  :       48280113
    Output bytes  :          78408
    Input  packets:          62369
    Output packets:           1089
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :        73469707               0 bps
   Input  packets:           96187               0 pps
   Drop   bytes  :               0               0 bps
   Drop   packets:               0               0 pps
  Ingress queues: 8 supported, 8 in use
  Queue counters:    ueued packets  Transmitted packets  Dropped
packets
    0 fc-be                  89399           89399                 0
    1 fc-ef                      0               0                 0
    2 fc-af1                     0               0                 0
    3 fc-nc                   6788            6788                 0
    4 fc-af2                     0               0                 0
    5 fc-af3                     0               0                 0
    6 fc-af4                     0               0                 0
    7 fc-af-extra                0               0                 0
```

Meaning The output indicates that the PE device is receiving multicast data from the source and classifying the data into the BE forwarding class.

Verify Receiver PE5 Interface Traffic Statistics

Purpose View the egress interface statistics in the receiver PE device.

Action Issue the following command:

regress@host# **run show interfaces xe-3/0/0 detail**

```
Physical interface: xe-3/0/0, Enabled, Physical link is Up
  Interface index: 138, SNMP ifIndex: 548, Generation: 141
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU
Device flags   : Present Running
```

```
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 00:1d:b5:26:51:ef, Hardware address:
00:1d:b5:26:51:ef
  Last flapped   : 2013-07-23 05:50:32 PDT (1d 21:53 ago)
  Statistics last cleared: 2013-07-25 03:42:21 PDT (00:01:25 ago)
  Traffic statistics:
   Input  bytes  :         525810            4952 bps
   Output bytes  :       52336448         4865968 bps
   Input  packets:           9474               2 pps
   Output packets:          72247             849 pps
   IPv6 transit statistics:
    Input  bytes  :          2503
    Output bytes  :      51993642
    Input  packets:            30
    Output packets:         68214
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :         514627               0 bps
   Input  packets:           5206               0 pps
   Drop   bytes  :              0               0 bps
   Drop   packets:              0               0 pps
  Egress queues: 8 supported, 8 in use
  Queue counters:  Queued packets  Transmitted packets  Dropped packets
    0 fc-be                52734               52734                0
    1 fc-ef                    0                   0                0
    2 fc-af1                   0                   0                0
    3 fc-nc                 2345                2345                0
    4 fc-af2                   0                   0                0
    5 fc-af3                   0                   0                0
    6 fc-af4                   0                   0                0
    7 fc-af-extra              0                   0                0
  Queue number:         Mapped forwarding classes
    0                   fc-be
    1                   fc-ef
    2                   fc-af1
    3                   fc-nc
```

Issue the following command:

regress@host# **run show interfaces xe-3/0/1 detail**

```
Physical interface: xe-3/0/1, Enabled, Physical link is Up
  Interface index: 139, SNMP ifIndex: 551, Generation: 142
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 00:1d:b5:26:51:f0, Hardware address:
00:1d:b5:26:51:f0
  Last flapped   : 2013-07-23 05:50:33 PDT (1d 21:53 ago)
  Statistics last cleared: 2013-07-25 03:42:30 PDT (00:01:21 ago)
  Traffic statistics:
   Input  bytes  :         462830            3456 bps
   Output bytes  :       48714436         4930744 bps
```

```
  Input  packets:           8546              1 pps
  Output packets:          67227            839 pps
  IPv6 transit statistics:
   Input  bytes  :             0
   Output bytes  :      48394780
   Input  packets:             0
   Output packets:         63458
 Ingress traffic statistics at Packet Forwarding Engine:
  Input  bytes  :         498936              0 bps
  Input  packets:           6521              0 pps
  Drop   bytes  :              0              0 bps
  Drop   packets:              0              0 pps
 Egress queues: 8 supported, 8 in use
 Queue counters:  Queued packets  Transmitted packets  Dropped packets
   0 fc-be              33804                33804                0
   1 fc-ef                  0                    0                0
   2 fc-af1                 0                    0                0
   3 fc-nc               1471                 1471                0
   4 fc-af2                 0                    0                0
   5 fc-af3                 0                    0                0
   6 fc-af4                 0                    0                0
   7 fc-af-extra            0                    0                0
```

Meaning The output indicates that the receiver PE device is receiving multicast data from the sender PE device and it is sending that multicast data to two receivers through downstream interfaces xe-3/0/0 and xe-3/0/1.

Verify MPLS P2MP LSP Statistics in Sender PE2

Purpose View the traffic statistics in the point-to-multipoint tunnels in the sender PE device.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp statistics**

```
P2MP name: 10.255.50.77:1:mvpn:vrf1, P2MP branch count: 3 -> inclusive
tunnel
To              From            State     Packets     Bytes LSPname
10.255.70.21    10.255.50.77    Up       18050833      13809074778
10.255.70.21:10.255.50.77:1:mvpn:vrf1
10.255.50.49    10.255.50.77    Up       18050833      13809074778
10.255.50.49:10.255.50.77:1:mvpn:vrf1
10.255.50.79    10.255.50.77    Up          37335         28423363
10.255.50.79:10.255.50.77:1:mvpn:vrf1

P2MP name: 10.255.50.77:1:mv1:vrf1, P2MP branch count: 3 -> selective
tunnel
To              From            State     Packets     Bytes LSPname
10.255.50.49    10.255.50.77    Up         200567        153504509
10.255.50.49:10.255.50.77:1:mv1:vrf1
10.255.70.21    10.255.50.77    Up         200567        153504509
10.255.70.21:10.255.50.77:1:mv1:vrf1
10.255.50.79    10.255.50.77    Up            404           315007
10.255.50.79:10.255.50.77:1:mv1:vrf1
```

Meaning The output indicates that the PE device is sending multicast data over MPLS LSPs to the receiver PE devices.

Configuration Scenario: Next-Generation Multicast VPN Services Over a Layer 3 VPN Network Where a PE Router Acts as a P Router

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for an NG MVPN configuration in which a PE router functions as a P router in a Layer 3 VPN network:

- Before You Begin
- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Before You Begin

This example is used to deploy multicast service to Layer 3 VPN customers. The configuration of NG MVPN in this example requires the configuration of the "Configuration Scenario: Layer 3 VPN Unicast IPv4 and IPv6" described earlier in this document.

Topology

Figure 30 shows the topology tested for the added NG MVPN service.

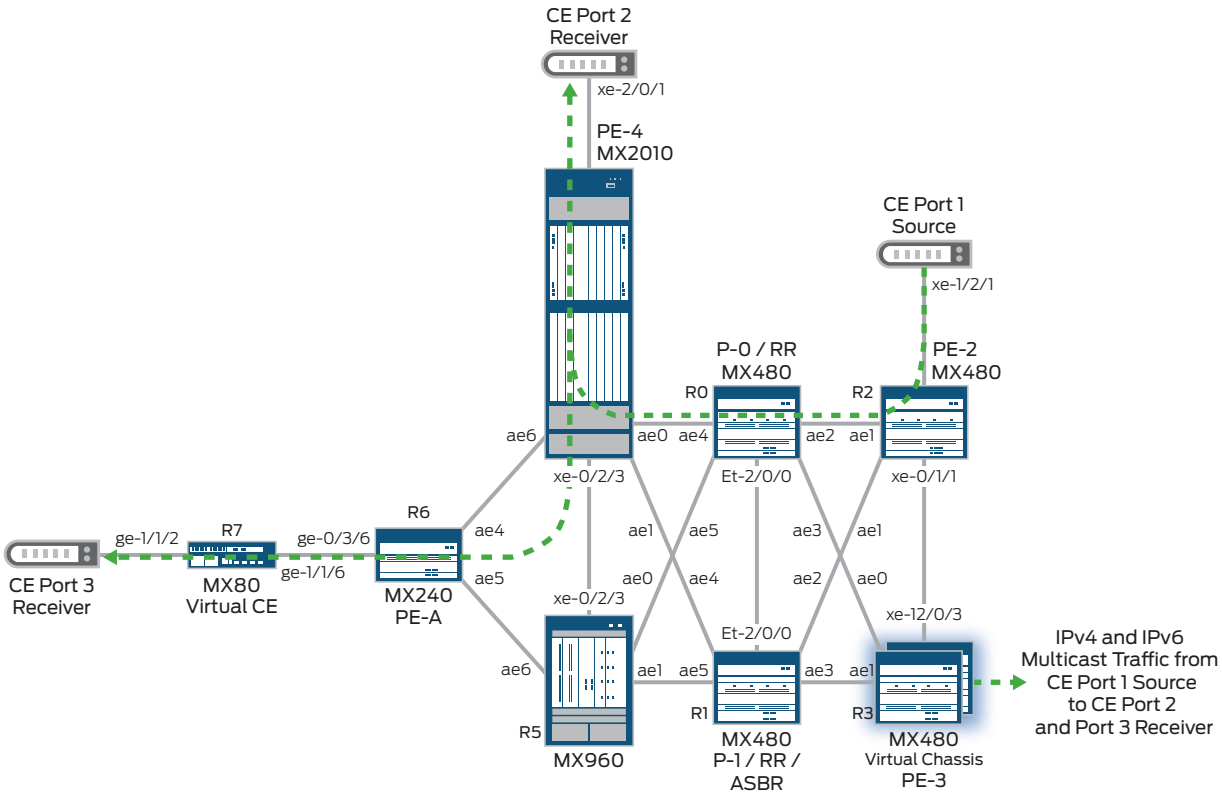

Figure 30: NG MVPN PE as P configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*
- *Router PE-6 Business Edge Solution Test Lab Configuration*
- *Router R7 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this scenario, the following guidelines were followed when configuring the devices:

- This multicast service uses next-generation MVPN in conjunction with RSVP P2MP provider tunnels.
- Multicast sources and receivers are both part of the same VPN. There are two multicast sources and receivers. Each source is sending traffic to 10 multicast groups.
- IGMPv3 and MLDv2 are used as the receiver-side protocols.
- PIM-SM is used in the customer instance.
- Two virtual tunnel interfaces are configured in PE4 for redundancy.

NOTE: Device PE4 is the PE device that is also acting as a P device for this MVPN service.

- A P2MP inclusive provider tunnel is configured for all groups.
- Bandwidth allocation and link protection is configured for each P2MP tunnel.
- IPv4 and IPv6 traffic from all sources to all receivers utilize random frame sizes ranging from 74 to 1,500 bytes and a line rate of 5 Mbps. All traffic is sent using the BE queue.

Verification

The following sections provide methods for verifying this next-generation MVPN configuration scenario along with sample command outputs:

- Verify MP-IBGP Peering with the Route Reflector
- Verify the Existence of the RSVP P2MP Inclusive Provider Tunnel in Sender PE2
- Verify MVPN Routes in VRF Table in Sender PE2
- Verify Multicast Route in Sender PE2
- Verify RSVP P2MP Sessions in Sender PE Device
- Verify RSVP Sessions
- Verify MPLS Table in Router P0
- Verify MPLS Table in the Device Acting as a P and PE Router for the Multicast VPN and Receiver PE4
- Verify Multicast Route Table in the Device Acting as a P and PE Router for the Multicast VPN and Receiver PE4 Device
- Verify C-PIM Database in Bud and Receiver PE4
- Verify MPLS Table in Receiver PE6
- Verify Multicast Route Table in Receiver Device PE6
- Verify C-PIM Database in Receiver PE6
- Verify Sender PE2 Interface Traffic Statistics
- Verify Receiver PE4 Interface Traffic Statistics
- Verify Receiver PE6 Interface Traffic Statistics
- Verify MPLS P2MP LSP Statistics in Sender PE Device

Verify MP-IBGP Peering with the Route Reflector

Purpose View the status of BGP neighborship with the route reflector in the PE device.

Action Issue the following command:

regress@host# **run show bgp neighbor 30.0.0.10**

```
Peer: 30.0.0.10+179 AS 70      Local: 30.0.0.12+64284 AS 70
  Type: Internal    State: Established    Flags: <ImportEval Sync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ next-hop-self ]
  Options: <Preference LocalAddress AddressFamily Multipath Rib-group
Refresh>
  Options: <MultipathAs AutheKeyChain BfdEnabled>
  Authentication key chain: chain1
  Authentication algorithm: hmac-sha-1-96
  Address families configured: inet-mvpn inet6-mvpn
  Local Address: 30.0.0.12 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 10.255.50.71   Local ID: 10.255.50.77   Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 1
  BFD: enabled, up
  NLRI for restart configured on peer: inet-mvpn inet6-mvpn
  NLRI advertised by peer: inet-mvpn inet6-mvpn
  NLRI for this session: inet-mvpn inet6-mvpn
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  NLRI that restart is negotiated for: inet-mvpn inet6-mvpn
  NLRI of received end-of-rib markers: inet-mvpn inet6-mvpn
  NLRI of all end-of-rib markers sent: inet-mvpn inet6-mvpn
  Peer supports 4 byte AS extension (peer-as 70)
  Peer does not support Addpath
```

Meaning The output indicates that the BGP neighborship with the route reflector is established with inet-mvpn and inet6-mvpn address families. All BGP message exchanges are authenticated. BFD over BGP is up.

Verify the Existence of the RSVP P2MP Inclusive Provider Tunnel in Sender PE2

Purpose View the status of RSVP P2MP LSPs in the sender PE device.

Action Issue the following command:

regress@host# **run show rsvp session p2mp**

```
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.50.49    10.255.50.77    Up       0  1 SE       -   413888 lsp-
to-PE4
10.255.51.168   10.255.50.77    Up       0  1 SE       -   413888 lsp-
to-PE6
```

Meaning The output indicates that the P2MP LSP is up and uses two sub-LSPs to two receiver PE devices.

Verify MVPN Routes in VRF Table in Sender PE2

Purpose View the MVPN routes in sender PE device mvpn1 mvpn routing table.

Action Issue the following command:

regress@host# **run show route table mvpn1.mvpn.0**

```
mvpn1.mvpn.0: 23 destinations, 45 routes (23 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.50.49:4004:10.255.50.49/240
                   *[BGP/170] 18:02:33, localpref 100, from 30.0.0.10
                      AS path: I, validation-state: unverified
                    > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
                     [BGP/170] 17:59:49, localpref 100, from 30.0.0.11
                      AS path: I, validation-state: unverified
                     > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
1:10.255.50.77:4004:10.255.50.77/240
                   *[MVPN/70] 18:04:21, metric2 1
                      Indirect
1:10.255.51.168:4004:10.255.51.168/240
                   *[BGP/170] 18:02:33, localpref 100, from 30.0.0.10
                      AS path: I, validation-state: unverified
                     > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
                     [BGP/170] 18:02:21, localpref 100, from 30.0.0.11
                      AS path: I, validation-state: unverified
                     > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
5:10.255.50.77:4004:32:2.255.0.6:32:230.2.0.1/240
                   *[PIM/105] 00:00:47
                      Multicast (IPv4) Composite
7:10.255.50.77:4004:70:32:2.255.0.6:32:230.2.0.1/240
                   *[PIM/105] 00:00:47
                      Multicast (IPv4) Composite
                     [BGP/170] 00:00:47, localpref 100, from 30.0.0.10
                      AS path: I, validation-state: unverified
                     > to 1.0.0.5 via ae0.0
                     [BGP/170] 00:00:47, localpref 100, from 30.0.0.11
                      AS path: I, validation-state: unverified
                     > to 1.0.0.21 via ae1.0
```

Meaning The output indicates that the PE device has generated a Type 1 route and has received a Type 1 route from every other PE device. Because this PE device is the rendezvous point, it has generated Type 5 routes. It has received Type 7 routes from other PE devices for the remote receivers.

Verify Multicast Route in Sender PE2

Purpose View the mvpn1 multicast routing table in the sender PE device.

Action Issue the following command:

regress@host# **run show multicast route extensive instance mvpn1**

```
Instance: mvpn1 Family: INET

Group: 230.2.0.1
    Source: 2.255.0.6/32
```

```
    Upstream interface: xe-1/2/1.4002
    Downstream interface list:
        ae0.0    Session description: Unknown
    Statistics: 60 kBps, 79 pps, 11174 packets
    Next-hop ID: 1949489
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:02:21
```

Issue the following command:

regress@host# **run show multicast route inet6 extensive instance mvpn1**

```
Instance: mvpn1 Family: INET6

Group: ff15:2::1
    Source: 2002::2ff:6/128
    Upstream interface: xe-1/2/1.4002
    Downstream interface list:
        ae0.0
    Session description: Unknown
    Statistics: 60 kBps, 78 pps, 3410 packets
    Next-hop ID: 1221972
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:00:43
```

Meaning The output indicates that the PE device has installed the routes for groups 230.2.0.1 and ff15:2::1. Source 2.255.0.6/32 and 2002::2ff:6/128 are the active sources sending traffic to these groups. The source is reachable through upstream interface xe-1/2/1.4002, and receivers are reachable through the downstream interface ae0. The upstream protocol is MVPN.

Verify RSVP P2MP Sessions in Sender PE Device

Purpose View the status of the RSVP point-to-multipoint inclusive provider tunnel.

Action Issue the following command:

regress@host# **run show rsvp session p2mp name mvpn-p2mp-lsp**

```
Ingress RSVP: 1127 sessions
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To              From           State  Rt Style Labelin Labelout LSPname
10.255.50.49    10.255.50.77   Up      0  1 SE       -    718141 lsp-to-PE4
10.255.51.168   10.255.50.77   Up      0  1 SE       -    718141 lsp-to-PE6
```

Meaning The output indicates that the P2MP LSP is up and has two sub-LSPs with a labelout value of 718141.

Verify RSVP Sessions

Purpose Verify RSVP point-to-multipoint sessions in P router and the receiver PE devices.

Action Issue the following command for router P0:

regress@host# **run show rsvp session p2mp name mvpn-p2mp-lsp**

```
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To            From          State  Rt Style Labelin Labelout LSPname
10.255.50.49  10.255.50.77  Up      0  1 SE  718141   402833 lsp-to-PE4
10.255.51.168 10.255.50.77  Up      0  1 SE  718141   402833 lsp-to-PE6
```

Issue the following command for bud and receiver device PE4:

regress@host# **run show rsvp session p2mp name mvpn-p2mp-lsp**

```
Egress RSVP: 1092 sessions
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To            From          State  Rt Style Labelin Labelout LSPname
10.255.50.49  10.255.50.77  Up      0  1 SE  402833        3 lsp-to-PE4

Transit RSVP: 47 sessions
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To            From          State  Rt Style Labelin Labelout LSPname
10.255.51.168 10.255.50.77  Up      0  1 SE  402833   368643 lsp-to-PE6
```

Issue the following command for receiver device PE6:

regress@host# **run show rsvp session p2mp name mvpn-p2mp-lsp**

```
Egress RSVP: 88 sessions
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 1
To            From          State  Rt Style Labelin Labelout LSPname
10.255.51.168 10.255.50.77  Up      0  1 SE  368643        3 lsp-to-PE6
```

Meaning The output indicates that the P2MP LSPs are up with labelin and labelout values.

Verify MPLS Table in Router P0

Purpose View the MPLS label table entry for the label 718141.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 718141**

```
mpls.0: 5653 destinations, 5653 routes (5653 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

718141             *[RSVP/7/2] 18:12:41, metric 1
                    > to 1.0.0.14 via ae4.0, Swap 402833
```

Meaning The output indicates that the P router expects the MPLS packet with a label of 718141. The router swaps the MPLS label with 402833 and sends the packet out using interface ae4.0.

Verify MPLS Table in the Device Acting as a P and PE Router for the Multicast VPN and Receiver PE4

Purpose View the MPLS label table entry for the label 402833.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 402833**

```
mpls.0: 5087 destinations, 5087 routes (5087 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both
```

```
402833             *[RSVP/7/2] 18:23:43, metric 1
                    > to 1.0.0.46 via ae6.0, Swap 368643
                      via vt-0/0/10.0, Pop -> primary vt interface
402833(S=0)        *[RSVP/7/2] 18:23:43, metric 1
                    > to 1.0.0.46 via ae6.0, Swap 368643
                      via vt-0/0/10.0, Pop
```

NOTE: Second route lookup occurs in mvpn1.inet.1 table in device PE4.

Issue the following command:

regress@host# **run show route table mvpn1.inet.1**

```
mvpn1.inet.1: 13 destinations, 13 routes (13 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

230.2.0.1,2.255.0.6/32*[MVPN/70] 00:24:12
                       Multicast (IPv4) Composite
```

Meaning The output indicates that device PE4 is expecting an MPLS packet with a label of 402833. The router swaps the label with label 368643 and sends the packet out using interface ae6.0. Because PE4 also has a local receiver attached, it also pops the label and sends the packet out using interface vt-0/0/10.0 for a second lookup using the mvpn1.inet.1 table.

Verify Multicast Route Table in the Device Acting as a P and PE Router for the Multicast VPN and Receiver PE4 Device

Purpose View the multicast route table in device PE4.

Action Issue the following command:

regress@host# **run show multicast route extensive instance mvpn1**

```
Instance: mvpn1 Family: INET

Group: 230.2.0.1
    Source: 2.255.0.6/32
    Upstream interface: vt-0/0/10.0
    Downstream interface list:
        xe-2/0/1.4002
    Session description: Unknown
    Statistics: 60 kBps, 79 pps, 130945 packets
    Next-hop ID: 1049431
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:27:29
```

Issue the following command:

regress@host# **run show multicast route extensive inet6 instance mvpn1**

```
Instance: mvpn1 Family: INET6

Group: ff15:2::1
    Source: 2002::2ff:6/128
    Upstream interface: vt-0/0/10.0
    Downstream interface list:
        xe-2/0/1.4002
    Session description: Unknown
    Statistics: 60 kBps, 79 pps, 167337 packets
    Next-hop ID: 1049427
```

```
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:36:00
```

Meaning The output indicates that device PE4 receives the multicast traffic from its upstream interface (vt-0/0/10.0) using MVPN and sends the packet to the multicast receivers through its downstream interface (xe-2/0/1.4002).

Verify C-PIM Database in Bud and Receiver PE4

Purpose View the customer PIM database in device PE4.

Action Issue the following command:

regress@host# **run show pim join extensive instance mvpn1**

```
Instance: PIM.mvpn1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 230.2.0.1
    Source: *
    RP: 11.2.255.2 -> RP is PE2
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 18:00:22
    Downstream neighbors:
        Interface: xe-2/0/1.4002
            2.255.1.1 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 18:00:25 Time since last Join: 18:00:25
    Number of downstream interfaces: 1
```

Issue the following command:

regress@host# **run show pim join extensive inet6 instance mvpn1**

```
Instance: PIM.mvpn1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff15:2::1
    Source: *
    RP: 2002::b02:ff02
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 18:06:28
    Downstream neighbors:
        Interface: xe-2/0/1.4002
            fe80::6687:880f:a204:518d State: Join Flags: SRW  Timeout:
Infinity
            Uptime: 18:06:29 Time since last Join: 18:06:29
    Number of downstream interfaces: 1
```

Meaning The output indicates that device PE4 has installed a PIM (*,G) state in its mvpn1 database. The rendezvous point is R2, the upstream protocol is BGP, and the upstream neighbor is learned through MVPN. Device PE4 has multicast receivers reachable through interface xe-2/0/1.4002.

Verify MPLS Table in Receiver PE6

Purpose View the MPLS label table entry in device PE6 for label 368643.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 368643**

```
mpls.0: 2009 destinations, 2009 routes (2009 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

368643             *[RSVP/7/2] 18:27:35, metric 1
                    > via vt-1/0/0.0, Pop
```

NOTE: Second route lookup occurs in mvpn1.inet.1 table in device PE6.

Issue the following command:

regress@host# **run show route table mvpn1.inet.1**

```
mvpn1.inet.1: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

224.0.0.0/4        *[Multicast/180] 18:30:24
                      MultiResolve
224.0.0.0/24       *[Multicast/180] 18:30:24
                      MultiDiscard
230.2.0.1,2.255.0.6/32*[MVPN/70] 00:26:46
                      Multicast (IPv4) Composite
```

Meaning The output indicates that device PE6 expects MPLS packets with label 368643. The device pops the label and sends the packet out using interface vt-1/0/0.0 for a second lookup in the mvpn1.inet.1 table.

Verify Multicast Route Table in Receiver Device PE6

Purpose View the multicast routes in routing table mvpn1 of device PE6.

Action Issue the following command:

regress@host# **run show multicast route extensive instance mvpn1**

```
Instance: mvpn1 Family: INET

Group: 230.2.0.1
    Source: 2.255.0.6/32
    Upstream interface: vt-1/0/0.0
    Downstream interface list:
        ge-0/3/6.4
    Session description: Unknown
    Statistics: 61 kBps, 79 pps, 148573 packets
    Next-hop ID: 1048725
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:31:11
```

Issue the following command:

regress@host# **run show multicast route extensive inet6 instance mvpn1**

```
Instance: mvpn1 Family: INET6

Group: ff15:2::1
    Source: 2002::2ff:6/128
    Upstream interface: vt-1/0/0.0
    Downstream interface list:
        ge-0/3/6.4
```

```
    Session description: Unknown
    Statistics: 61 kBps, 78 pps, 175131 packets
    Next-hop ID: 1048577
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:37:38
```

Meaning The output indicates that the PE device receives multicast traffic over interface vt-1/0/0.0 using the MVPN protocol. The device has receivers connected to its downstream interface (ge-0/3/6.4).

Verify C-PIM Database in Receiver PE6

Purpose View the customer PIM database in device PE6.

Action Issue the following command:

regress@host# **run show pim join extensive instance mvpn1**

```
Instance: PIM.mvpn1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 230.2.0.1
    Source: *
    RP: 11.2.255.2 -> RP is PE2
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 18:02:55
    Downstream neighbors:
        Interface: ge-0/3/6.4
            1.0.0.114 State: Join Flags: SRW Timeout: 154
            Uptime: 18:02:55 Time since last Join: 00:00:55
    Number of downstream interfaces: 1
```

Issue the following command:

regress@host# **run show pim join extensive inet6 instance mvpn1**

```
Instance: PIM.mvpn1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff15:2::1
    Source: *
    RP: 2002::b02:ff02
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 18:08:46
    Downstream neighbors:
        Interface: ge-0/3/6.4
            fe80::fac0:100:41f:9016 State: Join Flags: SRW Timeout: 164
            Uptime: 18:08:46 Time since last Join: 00:00:46
    Number of downstream interfaces: 1
```

Meaning The output indicates that device PE6 has installed PIM (*,G) state in its mvpn1 database. The rendezvous site is device R2, the upstream protocol is BGP, and the upstream neighbor is learned using MVPN. The PE6 device has multicast receivers reachable through interface ge-0/3/6.4.

Verify Sender PE2 Interface Traffic Statistics

Purpose View the ingress interface statistics in the sender PE device.

Action Issue the following command:

regress@host# **run show interfaces xe-1/2/1 detail**

```
Physical interface: xe-1/2/1, Enabled, Physical link is Up
  Interface index: 168, SNMP ifIndex: 12053, Generation: 171
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: b0:a8:6e:76:29:ef, Hardware address:
b0:a8:6e:76:29:ef
  Last flapped   : 2013-07-15 07:20:51 PDT (18:59:23 ago)
  Statistics last cleared: 2013-07-16 02:18:21 PDT (00:01:53 ago)
  Traffic statistics:
   Input  bytes  :      138655050          9795360 bps
   Output bytes  :         551395            35200 bps
   Input  packets:         192016             1778 pps
   Output packets:           6795               53 pps
   IPv6 transit statistics:
    Input  bytes  :      69080895
    Output bytes  :        111672
    Input  packets:         88871
    Output packets:          1551
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :       61974241                0 bps
   Input  packets:          81506                0 pps
   Drop   bytes  :              0                0 bps
   Drop   packets:              0                0 pps
  Ingress queues: 8 supported, 8 in use
  Queue counters:  Queued packets  Transmitted packets  Dropped packets
    0 fc-be                75760                75760                0
    1 fc-ef                    0                    0                0
    2 fc-af1                   0                    0                0
    3 fc-nc                 5746                 5746                0
    4 fc-af2                   0                    0                0
    5 fc-af3                   0                    0                0
    6 fc-af4                   0                    0                0
    7 fc-af-extra              0                    0                0
```

Meaning The output indicates that the PE device is receiving multicast data from the source and placing the data in the BE forwarding class.

Verify Receiver PE4 Interface Traffic Statistics

Purpose View the egress interface statistics in receiver device PE4.

Action Issue the following command:

regress@host# **run show interfaces xe-2/0/1 detail**

```
Physical interface: xe-2/0/1, Enabled, Physical link is Up
  Interface index: 1294, SNMP ifIndex: 9779, Generation: 3436
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
```

```
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues      : 8 supported, 8 maximum usable queues
  Schedulers      : 0
  Hold-times      : Up 0 ms, Down 0 ms
  Current address: 64:87:88:04:51:8d, Hardware address:
64:87:88:04:51:8d
  Last flapped    : 2013-07-14 08:02:30 PDT (1d 18:18 ago)
  Statistics last cleared: 2013-07-16 02:20:10 PDT (00:00:57 ago)
  Traffic statistics:
   Input  bytes  :         331255            72824 bps
   Output bytes  :       67854056          9571744 bps
   Input  packets:           6011              151 pps
   Output packets:          91214             1632 pps
   IPv6 transit statistics:
    Input  bytes  :          2714
    Output bytes  :      33869592
    Input  packets:            33
    Output packets:         44423
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :         361281                0 bps
   Input  packets:           3919                0 pps
   Drop   bytes  :              0                0 bps
   Drop   packets:              0                0 pps
  Egress queues: 8 supported, 8 in use
  Queue counters:  Queued packets  Transmitted packets  Dropped packets
    0 fc-be                19208                19208                0
    1 fc-ef                    0                    0                0
    2 fc-af1                   0                    0                0
    3 fc-nc                  221                  221                0
    4 fc-af2                   0                    0                0
    5 fc-af3                   0                    0                0
    6 fc-af4                   0                    0                0
    7 fc-af-extra              0                    0                0
```

Issue the following command:

regress@host# **run show interfaces ae6**

```
Physical interface: ae6, Enabled, Physical link is Up
  Interface index: 289, SNMP ifIndex: 693
  Link-level type: Ethernet, MTU: 9192, Speed: 20Gbps, BPDU Error:
None, MAC-REWRITE Error: None, Loopback: Disabled, Source filtering:
Disabled, Flow control: Disabled
  Pad to minimum frame size: Disabled
  Minimum links needed: 1, Minimum bandwidth needed: 0
  Device flags    : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Current address: 66:87:88:04:50:23, Hardware address:
66:87:88:04:50:23
  Last flapped    : 2013-11-18 00:58:17 PST (03:54:44 ago)
  Input rate      : 188544 bps (451 pps)
  Output rate     : 9837344 bps (2026 pps)

  Logical interface ae6.0 (Index 814) (SNMP ifIndex 1147)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :         30721        451       1617729       187560
        Output:        140697       1968      86019527      9796880
```

```
    Adaptive Statistics:
        Adaptive Adjusts:          0
        Adaptive Scans  :          0
        Adaptive Updates:          0
    Protocol inet, MTU: 9178
      Flags: Sendbcast-pkt-to-re
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 1.0.0.44/30, Local: 1.0.0.45, Broadcast: 1.0.0.47
    Protocol iso, MTU: 9175
    Protocol inet6, MTU: 9178
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 2002::100:2c/126, Local: 2002::100:2d
      Addresses, Flags: Is-Preferred
        Destination: fe80::/64, Local: fe80::6687:88ff:fe04:5023
    Protocol mpls, MTU: 9166, Maximum labels: 3
    Protocol multiservice, MTU: Unlimited
```

Meaning The output indicates that device PE4 is receiving multicast data from the sender and it is sending the data through two downstream interfaces xe-2/0/1 and ae6.

Verify Receiver PE6 Interface Traffic Statistics

Purpose View the egress interface statistics in the receiver PE6 device.

Action Issue the following command:

regress@host# **run show interfaces ge-0/3/6 detail**

```
Physical interface: ge-0/3/6, Enabled, Physical link is Up
  Interface index: 1252, SNMP ifIndex: 5122, Generation: 1317
  Link-level type: Ethernet, MTU: 1518, Speed: 1000mbps, BPDU Error:
None, MAC-REWRITE Error: None, Loopback: Disabled, Source filtering:
Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
  Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 78:19:f7:52:c1:f5, Hardware address:
78:19:f7:52:c1:f5
  Last flapped   : 2013-07-05 08:52:47 PDT (1w3d 17:30 ago)
  Statistics last cleared: 2013-07-16 02:22:32 PDT (00:00:40 ago)
  Traffic statistics:
   Input  bytes  :              409661                77792 bps
   Output bytes  :            48397017              9684728 bps
   Input  packets:                4347                  107 pps
   Output packets:               63109                 1584 pps
   IPv6 transit statistics:
    Input  bytes  :                 192
    Output bytes  :            24214920
    Input  packets:                   2
    Output packets:               31186
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :              219832                    0 bps
   Input  packets:                1810                    0 pps
   Drop   bytes  :                   0                    0 bps
   Drop   packets:                   0                    0 pps
```

```
Egress queues: 8 supported, 8 in use
Queue counters:  Queued packets  Transmitted packets  Dropped packets
  0 fc-be                20144                20144                0
  1 fc-ef                    0                    0                0
  2 fc-af1                   0                    0                0
  3 fc-nc                  644                  644                0
  4 fc-af2                   0                    0                0
  5 fc-af3                   0                    0                0
  6 fc-af4                   0                    0                0
  7 fc-af-extra              0                    0                0
```

Meaning The output indicates that device PE6 is receiving multicast data from the sender and it is sending the data through one downstream interface ge-0/3/6.

Verify MPLS P2MP LSP Statistics in Sender PE Device

Purpose View the traffic statistics in the point-to-multipoint tunnels in the sender PE device.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp statistics**

```
Ingress LSP: 84 sessions
P2MP name: mvpn-p2mp-lsp, P2MP branch count: 2
To              From            State     Packets          Bytes LSPname
10.255.50.49  10.255.50.77    Up          18950    14433240 lsp-to-PE4
10.255.51.168 10.255.50.77    Up          18950    14433240 lsp-to-PE6
```

Meaning The output indicates that the PE device is sending multicast data over MPLS LSPs to the receiver PE devices.

Configuration Scenario: Next-Generation Multicast VPN Services Over a Layer 3 VPN Network with Extranet Multicast

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for an NG-MVPN configuration in which extranet multicast is implemented in a Layer 3 VPN topology:

- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Topology

Figure 31 shows the topology tested for the added next-generation MVPN service.

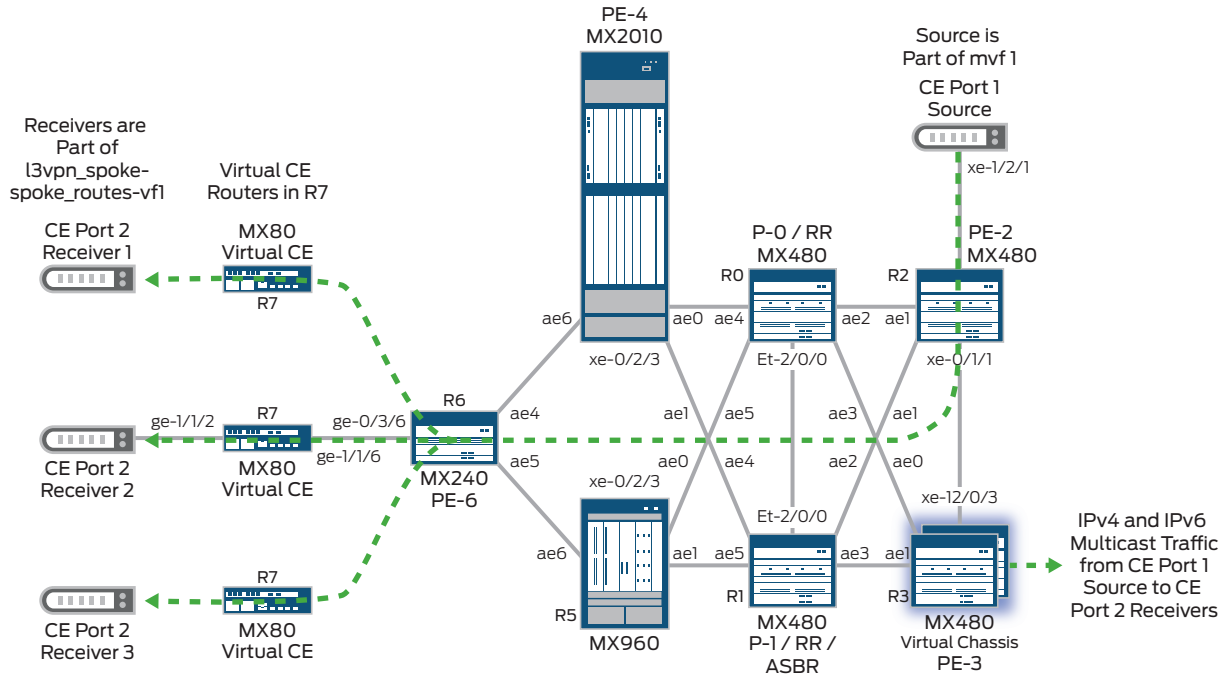

Figure 31: NG MVPN extranet multicast configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*
- *Router PE-6 Business Edge Solution Test Lab Configuration*
- *Router R7 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this scenario, the following guidelines were followed when configuring the devices:

- Multicast sources and receivers are located in different VPNs.
- There are two multicast sources and six multicast receivers. Each source sends traffic to 10 multicast groups.
- IGMPv3 and MLDv2 are used as the receiver side protocols.
- PIM-SM is used in the customer instance.
- A P2MP inclusive provider tunnel is configured for all groups.
- IPv4 and IPv6 traffic from all sources to all receivers utilize random frame sizes ranging from 74 to 1,500 bytes and a line rate of 5 Mbps. All traffic is sent to the BE queue.

Verification

The following sections provide methods for verifying this Layer 3 VPN configuration scenario along with sample command outputs:

- Verify MP-IBGP Peering with the Route Reflector
- Verify the Existence of the RSVP P2MP Inclusive Provider Tunnel in Sender PE2
- Verify MVPN Routes in the VRF Table in Sender PE2
- Verify Multicast Routes in Sender PE2
- Verify RSVP P2MP Sessions in the Sender PE Device
- Verify RSVP Sessions
- Verify MPLS Table Entries on Receiver Device P0
- Verify MPLS Table on Receiver Device PE6
- Verify the Multicast Route Table on Receiver Device PE6
- Verify the C-PIM Database on Device PE6
- Verify MPLS P2MP LSP Statistics in the Sender PE2 Device
- Verify the Sender PE2 Device Interface Traffic Statistics
- Verify the Receiver PE6 Device Interface Traffic Statistics

Verify MP-IBGP Peering with the Route Reflector

Purpose View the status of MP-IBGP neighborship with the route reflector in the PE devices.

Action Issue the following command:

regress@host# **run show bgp neighbor 30.0.0.10**

```
Peer: 30.0.0.10+179 AS 70      Local: 30.0.0.12+64284 AS 70
  Type: Internal    State: Established    Flags: <ImportEval Sync>
  Last State: EstabSync     Last Event: RecvKeepAlive
  Last Error: None
  Export: [ next-hop-self ]
  Options: <Preference LocalAddress AddressFamily Multipath Rib-group
Refresh>
  Options: <MultipathAs AutheKeyChain BfdEnabled>
  Authentication key chain: chain1
  Authentication algorithm: hmac-sha-1-96
  Address families configured: inet-mvpn inet6-mvpn
  Local Address: 30.0.0.12 Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 10.255.50.71    Local ID: 10.255.50.77   Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 1
  BFD: enabled, up
  NLRI for restart configured on peer: inet-mvpn inet6-mvpn
  NLRI advertised by peer: inet-mvpn inet6-mvpn
  NLRI for this session: inet-mvpn inet6-mvpn
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
```

```
  NLRI that restart is negotiated for: inet-mvpn inet6-mvpn
  NLRI of received end-of-rib markers: inet-mvpn inet6-mvpn
  NLRI of all end-of-rib markers sent: inet-mvpn inet6-mvpn
  Peer supports 4 byte AS extension (peer-as 70)
  Peer does not support Addpath
```

Meaning The output indicates that the BGP neighborship with the route reflector is established with inet-mvpn and inet6-mvpn address families. All BGP message exchanges are authenticated. BFD over BGP is up.

Verify the Existence of the RSVP P2MP Inclusive Provider Tunnel in Sender PE2

Purpose View the status of the RSVP P2MP inclusive provider tunnel in device PE2.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp**

```
P2MP name: 10.255.50.77:4003:mvpn:mvrf1, P2MP branch count: 1
To              From            State Rt P     ActivePath       LSPname
10.255.51.168   10.255.50.77    Up     0 *
10.255.51.168:10.255.50.77:4003:mvpn:mvrf1
```

Meaning The output indicates that the P2MP tunnel is up with one sub-LSP.

Verify MVPN Routes in the VRF Table in Sender PE2

Purpose View the MVPN routes in the mvrf1 routing table of device PE2.

Action Issue the following command:

regress@host# **run show route table mvrf1.mvpn.0**

```
mvrf1.mvpn.0: 22 destinations, 43 routes (22 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.50.77:4003:10.255.50.77/240
                   *[MVPN/70] 19:21:18, metric2 1
                      Indirect
1:10.255.51.168:4001:10.255.51.168/240
                   *[BGP/170] 19:19:30, localpref 100, from 30.0.0.10
                      AS path: I, validation-state: unverified
                    > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
                    [BGP/170] 19:19:18, localpref 100, from 30.0.0.11
                      AS path: I, validation-state: unverified
                    > to 1.0.0.5 via ae0.0
                      to 1.0.0.21 via ae1.0
5:10.255.50.77:4003:32:2.255.0.2:32:230.1.0.1/240
                   *[PIM/105] 00:03:08
                      Multicast (IPv4) Composite
7:10.255.50.77:4003:70:32:2.255.0.2:32:230.1.0.1/240
                   *[PIM/105] 00:03:08
                      Multicast (IPv4) Composite
                    [BGP/170] 00:03:07, localpref 100, from 30.0.0.10
                      AS path: I, validation-state: unverified
                    > to 1.0.0.5 via ae0.0
                    [BGP/170] 00:03:07, localpref 100, from 30.0.0.11
                      AS path: I, validation-state: unverified
                    > to 1.0.0.21 via ae1.0
```

Meaning The output indicates that device PE2 has a send Type 1 route and a Type 5 route and has received Type 1 and Type 7 routes from other PE devices.

Verify Multicast Routes in Sender PE2

Purpose View the mvrf1 multicast routing table in device PE2.

Action Issue the following command:

regress@host# **run show multicast route extensive instance mvrf1**

```
Instance: mvrf1 Family: INET

Group: 230.1.0.1
    Source: 2.255.0.2/32
    Upstream interface: xe-1/2/1.4001
    Downstream interface list:
        ae0.0
    Session description: Unknown
    Statistics: 61 kBps, 79 pps, 21257 packets
    Next-hop ID: 1153708
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:04:42
```

Issue the following command:

regress@host# **run show multicast route inet6 extensive instance mvrf1**

```
Instance: mvrf1 Family: INET6

Group: ff15:1::1
    Source: 2002::2ff:2/128
    Upstream interface: xe-1/2/1.4001
    Downstream interface list:
        ae0.0
    Session description: Unknown
    Statistics: 63 kBps, 78 pps, 26105 packets
    Next-hop ID: 1142092
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:05:49
```

Meaning The output indicates that device PE2 has installed multicast routes in the mvrf1 table for groups 230.1.0.1 and ff15:1::1 and for sources 2.255.0.2/32 and 2002::2ff:2/128. Sources are reachable through upstream interface xe-1/2/1.4001, and the receivers are reachable through downstream interface ae0.0.

Verify RSVP P2MP Sessions in the Sender PE Device

Purpose View the status of the RSVP P2MP session in device PE2.

Action Issue the following command:

regress@host# **run show rsvp session p2mp name 10.255.50.77:4003:mvpn:mvrf1**

```
P2MP name: 10.255.50.77:4003:mvpn:mvrf1, P2MP branch count: 1
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.51.168   10.255.50.77    Up       0  1 SE       -   718157
10.255.51.168:10.255.50.77:4003:mvpn:mvrf1
```

Meaning The output indicates that the P2MP session is up and the labelout value is 718157.

Verify RSVP Sessions

Purpose View the status of RSVP P2MP sessions in device P0 and PE6.

Action Issue the following command for device P0

regress@host# **run show rsvp session p2mp name 10.255.50.77:4003:mvpn:mvrf1**

```
P2MP name: 10.255.50.77:4003:mvpn:mvrf1, P2MP branch count: 1
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.51.168   10.255.50.77    Up       0  1 SE  718157   402913
10.255.51.168:10.255.50.77:4003:mvpn:mvrf1
```

Issue the following command on receiver device PE6:

regress@host# **run rsvp session p2mp name 10.255.50.77:4003:mvpn:mvrf1**

```
P2MP name: 10.255.50.77:4003:mvpn:mvrf1, P2MP branch count: 1
To              From            State   Rt Style Labelin Labelout
LSPname
10.255.51.168   10.255.50.77    Up       0  1 SE      16        -
10.255.51.168:10.255.50.77:4003:mvpn:mvrf1
```

Meaning The output indicates that the P2MP sessions are up.

Verify MPLS Table Entries on Receiver Device P0

Purpose View the MPLS label table entry in device P0 for label 718157.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 718157**

```
mpls.0: 5623 destinations, 5623 routes (5623 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

718157             *[RSVP/7/2] 19:27:23, metric 1
                    > to 1.0.0.14 via ae4.0, Swap 402913
```

Meaning The output indicates that device P0 swaps label 718157 with a label of 402913 and sends the packet out through interface ae4.0.

Verify MPLS Table on Receiver Device PE6

Purpose View the MPLS label table entry in PE6 for label 16.

Action Issue the following command:

regress@host# **run show route table mpls.0 label 16**

```
mpls.0: 2009 destinations, 2009 routes (2009 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

16                 *[VPN/0] 19:30:26
                      to table l3vpn_spoke-spoke_routes-vrf1.inet.0,
Pop
```

NOTE: Second route lookup occurs in the l3vpn_spoke-spoke_routes-vrf1.inet.1 table in device PE6.

Issue the following command:

regress@host# **run show route table l3vpn_spoke-spoke_routes-vrf1.inet.1**

```
l3vpn_spoke-spoke_routes-vrf1.inet.1: 13 destinations, 13 routes (13
active, 0 holddown, 0 hidden)
```

```
+ = Active Route, - = Last Active, * = Both

230.1.0.1,2.255.0.2/32*[MVPN/70] 00:13:21
                      Multicast (IPv4) Composite
```

Meaning The output indicates that device PE6 pops the label and performs a second lookup in the l3vpn_spoke-spoke_routes-vrf1.inet.1 table.

Verify the Multicast Route Table on Receiver Device PE6e

Purpose View the instance l3vpn_spoke-spoke_routes-vrf1 multicast route table in device PE6.

Action Issue the following command:

regress@host# **run show multicast route extensive instance l3vpn_spoke-spoke_routes-vrf1**

```
Instance: l3vpn_spoke-spoke_routes-vrf1 Family: INET

Group: 230.1.0.1
    Source: 2.255.0.2/32
    Upstream interface: lsi.4608
    Downstream interface list:
        ge-0/3/6.1 ge-0/3/6.2 ge-0/3/6.3
    Session description: Unknown
    Statistics: 61 kBps, 79 pps, 66707 packets
    Next-hop ID: 1048749
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:14:16
```

Issue the following command:

regress@host# **run show multicast route inet6 extensive instance l3vpn_spoke-spoke_routes-vrf1**

```
Instance: l3vpn_spoke-spoke_routes-vrf1 Family: INET6

Group: ff15:1::1
    Source: 2002::2ff:2/128
    Upstream interface: lsi.4608
    Downstream interface list:
        ge-0/3/6.1 ge-0/3/6.2 ge-0/3/6.3
    Session description: Unknown
    Statistics: 60 kBps, 78 pps, 69246 packets
    Next-hop ID: 1048737
    Upstream protocol: MVPN
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 00:14:59
```

Meaning The output indicates that the PE device has installed the multicast routes for groups 230.1.0.1 and ff15:1::1 and for sources 2.255.0.2/32 and 2002::2ff:2/128. Upstream interfaces are the LSI interfaces facing the core. The upstream protocol is MVPN and there are three downstream receivers on interfaces ge-0/3/6.1, ge-0/3/6.2, and ge-0/3/6.3.

Verify the C-PIM Database on Device PE6

Purpose View the customer PIM database in device PE6.

Action Issue the following command:

regress@host# **run show pim join extensive instance l3vpn_spoke-spoke_routes-vrf1**

```
Instance: PIM.l3vpn_spoke-spoke_routes-vrf1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 230.1.0.1
    Source: *
    RP: 11.2.255.1
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 00:32:04
    Downstream neighbors:
        Interface: ge-0/3/6.1
            1.0.0.54 State: Join Flags: SRW Timeout: 161
            Uptime: 00:32:02 Time since last Join: 00:00:48
    Downstream neighbors:
        Interface: ge-0/3/6.2
            1.0.0.58 State: Join Flags: SRW Timeout: 161
            Uptime: 00:31:58 Time since last Join: 00:00:48
    Downstream neighbors:
        Interface: ge-0/3/6.3
            1.0.0.62 State: Join Flags: SRW Timeout: 161
            Uptime: 00:32:04 Time since last Join: 00:00:48
    Number of downstream interfaces: 3
```

Issue the following command:

regress@host# **run show pim join inet6 extensive instance l3vpn_spoke-spoke_routes-vrf1**

```
Instance: PIM.l3vpn_spoke-spoke_routes-vrf1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff15:1::1
    Source: *
    RP: 2002::b02:ff01
    Flags: sparse,rptree,wildcard
    Upstream protocol: BGP
    Upstream interface: Through BGP
    Upstream neighbor: Through MVPN
    Upstream state: Join to RP
    Uptime: 00:32:41
    Downstream neighbors:
        Interface: ge-0/3/6.1
            fe80::fac0:100:11f:9016 State: Join Flags: SRW Timeout: 182
            Uptime: 00:32:40 Time since last Join: 00:00:27
    Downstream neighbors:
        Interface: ge-0/3/6.2
            fe80::fac0:100:21f:9016 State: Join Flags: SRW Timeout: 188
            Uptime: 00:32:41 Time since last Join: 00:00:21
        Interface: ge-0/3/6.3
            fe80::fac0:100:31f:9016 State: Join Flags: SRW Timeout: 153
            Uptime: 00:32:38 Time since last Join: 00:00:57
    Number of downstream interfaces: 3
```

Meaning The output indicates that device PE6 has installed PIM (*,G) state in its l3vpn_spoke-spoke_routes-vrf1 database. The rendezvous point is device PE2, the upstream protocol is BGP, and the upstream neighbor is learned through MVPN. The PE6 device multicast receivers are reachable through interfaces ge-0/3/6.1, ge-0/3/6.2, and ge-0/3/6.3.

Verify MPLS P2MP LSP Statistics in the Sender PE2 Device

Purpose View the ingress interface statistics for device PE2.

Action Issue the following command:

regress@host# **run show mpls lsp p2mp name 10.255.50.77:4003:mvpn:mvrf1 statistics**

```
P2MP name: 10.255.50.77:4003:mvpn:mvrf1, P2MP branch count: 1
To              From            State     Packets         Bytes LSPname
10.255.51.168   10.255.50.77    Up          17372      13292783
10.255.51.168:10.255.50.77:4003:mvpn:mvrf1
```

Meaning The output indicates that device PE2 is sending multicast traffic over the P2MP LSP to the receiver PE device.

Verify the Sender PE2 Device Interface Traffic Statistics

Purpose View the MPLS LSP statistics for device PE2.

Action Issue the following command:

regress@host# **run show interfaces xe-1/2/1 detail**

```
Physical interface: xe-1/2/1, Enabled, Physical link is Up
  Interface index: 168, SNMP ifIndex: 12053, Generation: 171
  Link-level type: Flexible-Ethernet, MTU: 9122, LAN-PHY mode, Speed:
10Gbps, BPDU Error: None, Loopback: None, Source filtering: Disabled,
Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: b0:a8:6e:76:29:ef, Hardware address:
b0:a8:6e:76:29:ef
  Last flapped   : 2013-07-15 07:20:51 PDT (19:56:49 ago)
  Statistics last cleared: 2013-07-16 03:15:03 PDT (00:02:37 ago)
  Traffic statistics:
   Input  bytes  :            192150910              9582624 bps
   Output bytes  :               763396                45408 bps
   Input  packets:               265970                 1729 pps
   Output packets:                 9400                   66 pps
   IPv6 transit statistics:
    Input  bytes  :             95685534
    Output bytes  :               154872
    Input  packets:               123186
    Output packets:                 2151
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :             25204482                    0 bps
   Input  packets:                42811                    0 pps
   Drop   bytes  :                    0                    0 bps
   Drop   packets:                    0                    0 pps
  Ingress queues: 8 supported, 8 in use
  Queue counters: Queued packets  Transmitted packets  Dropped packets
    0 fc-be                29831                29831                0
    1 fc-ef                    0                    0                0
    2 fc-af1                   0                    0                0
    3 fc-nc                12980                12980                0
```

```
    4 fc-af2                  0                    0                    0
    5 fc-af3                  0                    0                    0
    6 fc-af4                  0                    0                    0
    7 fc-af-extra             0                    0                    0
```

Meaning The output indicates that device PE2 is receiving multicast traffic and placing the data in the BE forwarding class.

Verify the Receiver PE6 Device Interface Traffic Statistics

Purpose View the egress interface statistics for device PE6.

Action Issue the following command:

regress@host# **run show interfaces ge-0/3/6 detail**

```
Physical interface: ge-0/3/6, Enabled, Physical link is Up
  Interface index: 1252, SNMP ifIndex: 5122, Generation: 1317
  Link-level type: Ethernet, MTU: 1518, Speed: 1000mbps, BPDU Error:
None, MAC-REWRITE Error: None, Loopback: Disabled, Source filtering:
Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
  Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Schedulers     : 0
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 78:19:f7:52:c1:f5, Hardware address:
78:19:f7:52:c1:f5
  Last flapped   : 2013-07-05 08:52:47 PDT (1w3d 18:26 ago)
  Statistics last cleared: 2013-07-16 03:19:00 PDT (00:00:35 ago)
  Traffic statistics:
   Input  bytes  :         345149           78024 bps
   Output bytes  :      122908127        28997104 bps
   Input  packets:           3688             108 pps
   Output packets:         160355            4741 pps
   IPv6 transit statistics:
    Input  bytes  :             0
    Output bytes  :      61446648
    Input  packets:             0
    Output packets:         79524
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :         252188               0 bps
   Input  packets:           2117               0 pps
   Drop   bytes  :              0               0 bps
   Drop   packets:              0               0 pps
  Egress queues: 8 supported, 8 in use
  Queue counters:  Queued packets  Transmitted packets  Dropped packets
    0 fc-be                61080                61080                0
    1 fc-ef                    0                    0                0
    2 fc-af1                   0                    0                0
    3 fc-nc                 1390                 1390                0
    4 fc-af2                   0                    0                0
    5 fc-af3                   0                    0                0
    6 fc-af4                   0                    0                0
    7 fc-af-extra              0                    0                0
```

Meaning The output indicates that device PE6 is receiving the multicast traffic from the core and placing the data in the BE forwarding class for delivery to the receivers.

CHAPTER 8 Value-Added Services

Configuration Scenario: Layer 3 VPN Network Hub-and-Spoke Configuration with Service Provider Firewall and NAT Service

The following sections provide the configuration scenario topology, links to router configurations, configuration guidelines, and verifications for this tested firewall and NAT service configuration that functions over a Layer 3 VPN business edge hub-and-spoke topology:

- Topology
- Router Configurations
- Configuration Guidelines
- Verification

Topology

Figure 32 shows the topology tested for the added hub-and-spoke firewall/NAT value-added service.

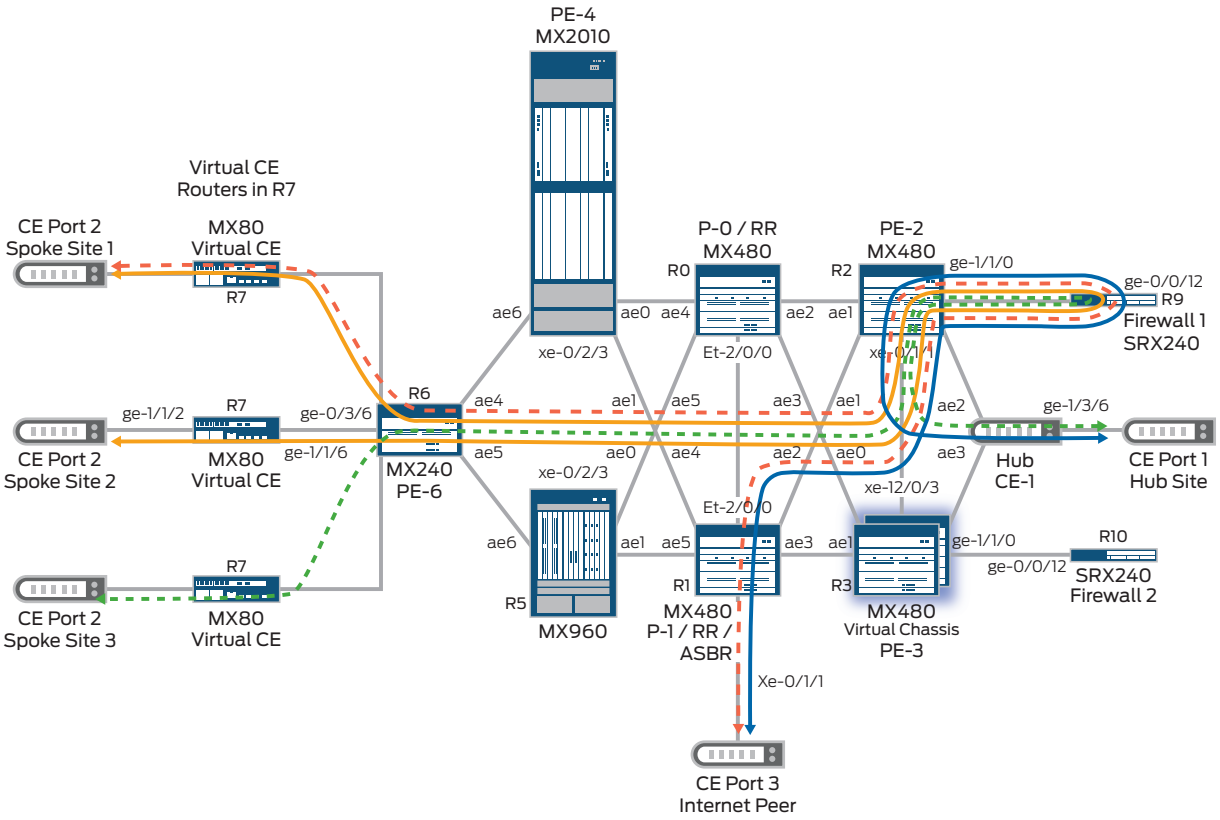

Figure 32: Hub-and-spoke firewall and NAT service configuration topology

Router Configurations

You can access the configurations for each router in this tested scenario using the following links:

- *Router P-0 Business Edge Solution Test Lab Configuration*
- *Router P-1 Business Edge Solution Test Lab Configuration*
- *Router PE-2 Business Edge Solution Test Lab Configuration*
- *Router PE-3 Business Edge Solution Test Lab Configuration*
- *Router PE-4 Business Edge Solution Test Lab Configuration*
- *Router PE-5 Business Edge Solution Test Lab Configuration*
- *Router PE-6 Business Edge Solution Test Lab Configuration*
- *Router R7 Business Edge Solution Test Lab Configuration*
- *Router CE-1 Business Edge Solution Test Lab Configuration*
- *Router R9 Business Edge Solution Test Lab Configuration*
- *Router R10 Business Edge Solution Test Lab Configuration*

NOTE: The configurations provided use configuration groups to simplify the management of each router interface and ultimately decrease the length of the router configuration. Configuration groups (simply called *groups* in the general Junos OS documentation) are reusable snippets of configurations that you configure at the **[edit groups]** hierarchy level. Once configured, you can apply individual groups (using the **apply-groups** statement) to different locations in the configuration hierarchy.

Configuration Guidelines

In this scenario, the following guidelines were followed when configuring the devices:

- This example provides Layer 3 VPN service between customer central and branch sites. Internet service and firewall/NAT service is also provided to the customer central and branch sites.
- Customer branch site-to-branch site communication, branch site-to-central site communication, and central site-to-branch site communication occur through the firewall device (SRX Series) located in the service provider network.
- The service provider is also providing NAT service for Internet access from the customer central site and branch sites.
- Customer branch sites are connected to the spoke PE device (R6); central sites are connected to the hub PE devices (R2 and R3); firewall/NAT devices are also connected to the hub PE devices (R2 and R3).
- Router R1 functions as an AS boundary router for the purpose of Internet peering.
- To increase scalability, all customer branch site attachment circuits are connecting to one routing instance (l3vpn_spoke-spoke_routes-vrf1) configured on the spoke PE device (R6).
- Branch sites peer with the spoke PE device (R6) using RIP/RIPng.
- A second routing instance configured on the spoke PE device (l3vpn_spoke-hub_routes-vrf1) imports and exports routes from the hub PE devices (R2 and R3).
- Each hub PE device contains two routing instances per customer.
- The hub PE devices peer with the SRX Series firewall/NAT device and customer central sites using EBGP.
- All ingress customer branch site traffic to the spoke PE device is redirected to the l3vpn_spoke-hub_routes-vrf1 routing instance through the use of either a firewall filter or a static default route as defined in the l3vpn_spoke-spoke_routes-vrf1 routing instance.
- To facilitate Internet access, a static default route is configured in the global routing table on each hub PE device and exchanged with the spoke PE device through the SRX Series firewall/NAT device.
- The hub PE devices maintain all Internet routes in their global routing tables.
- Two security policies are implemented in the SRX Series device. One policy facilitates customer branch-to-branch site communication. The other policy facilitates bidirectional branch site-to-central site communication.
- The SRX Series devices perform source NAT on traffic received from customer branch sites and central sites that is destined for the Internet. Conversely, the SRX Series devices also perform destination NAT on traffic received from the Internet that is destined for customer branch sites and central sites.
- Screens are applied in SRX Series zones to impose session limits.
- One virtual routing instance is configured on each SRX Series device per customer.
- Connectivity fault management (CFM) and BFD are configured between hub-and-spoke PE devices and CE devices to facilitate faster link-level failure detection.

- Network-level redundancy is configured between the two SRX Series devices resulting in only one SRX Series device serving Layer 3 VPN customers at a time. This configuration results in the firewall/NAT devices, each residing in a different location, electing a primary router to become active by sending route announcements to the hub PE device.
- This example configuration was tested with fully meshed, bidirectional IPv4 and IPv6 traffic using a fixed frame size of 256 bytes and a line rate of 1 percent passing from spoke sites to spoke sites, spoke to hub sites, the hub site to spoke sites, spoke-and-hub sites to the Internet, and the Internet to spoke-and-hub sites. All traffic is sent to the BE queue.

Verification

The following sections provide methods for verifying this Layer 3 VPN configuration scenario along with sample command outputs:

- Verify Local IPv4 Spoke Site Routes for Spoke Device PE6
- Verify Local IPv6 Spoke Site Routes for Spoke Device PE6
- Verify Hub Site IPv4 Routes from Hub Devices to Spoke Devices
- Verify Hub Site IPv6 Routes from Hub Devices to Spoke Devices
- Verify IPv4 Default Routes in the Spoke PE6 Device from Hub Device PE2
- Verify IPv6 Default Route in the Spoke PE6 Device from Hub Device PE2
- Verify Static IPv4 Default Route in the Spoke PE6 Device
- Verify Static IPv6 Default Route in the Spoke PE6 Device
- Verify Spoke Site IPv4 Routes in Hub PE2 from Remote Spoke Device PE6
- Verify Spoke Site IPv6 Routes in Hub PE2 from Remote Spoke Device PE6
- Verify Local Hub Site IPv4 Route in Hub Device PE2
- Verify Local Hub Site IPv6 Route in Hub Device PE2
- Verify Spoke-and-Hub Site Routes in the SRX from Hub Device PE2
- Verify Default Routes in the SRX Series from Hub Device PE2
- Verify Spoke-and-Hub Site Routes in Device PE2 from the SRX Series Device
- Verify the Default Route in Hub Device PE2
- Verify the Static Destination NAT Pool Route in the SRX Series Device
- Verify the Destination NAT Pool Route in Hub Device PE2 from the SRX Series Device
- Verify Spoke Device PE6 Interface Statistics
- Verify Hub PE2 Interface Statistics
- Verify SRX Interface Statistics
- Verify AS Boundary Router Device Interface Statistics

Verify Local IPv4 Spoke Site Routes for Spoke Device PE6

Purpose View the local branch site IPv4 routes in the spoke PE device.

Action Issue the following command to verify local spoke site 100.10.10.0/30 in spoke device PE6:

regress@host# **run show route 100.10.10.0/30**

```
l3vpn_spoke-spoke_routes-vrf1.inet.0: 313 destinations, 315 routes (313
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100.10.10.0/30     *[RIP/100] 21:03:40, metric 2, tag 0
                    > to 1.0.0.54 via ge-0/3/6.1
```

Issue the following command to verify local spoke site 110.10.10.0/30 in spoke device PE6:

regress@host# **run show route 110.10.10.0/30**

```
l3vpn_spoke-spoke_routes-vrf1.inet.0: 313 destinations, 315 routes (313
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

110.10.10.0/30     *[RIP/100] 21:04:53, metric 2, tag 0
                    > to 1.0.0.58 via ge-0/3/6.2
```

Issue the following command to verify local spoke site 120.10.10.0/30 in spoke device PE6:

regress@host# **run show route 120.10.10.0/30**

```
l3vpn_spoke-spoke_routes-vrf1.inet.0: 313 destinations, 315 routes (313
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

120.10.10.0/30     *[RIP/100] 21:05:24, metric 2, tag 0
                    > to 1.0.0.62 via ge-0/3/6.3
```

Meaning The output indicates that the PE device has learned the IPv4 routes from its directly connected branch sites through RIP and installed them in the **l3vpn_spoke-spoke_routes-vrf1** routing table.

Verify Local IPv6 Spoke Site Routes for Spoke Device PE6

Purpose View the local branch site IPv6 routes in the spoke PE device.

Action Issue the following command to verify local spoke site 2002::640a:a00/126 in spoke device PE6.

regress@host# **run show route 2002::640a:a00/126**

```
2002::640a:a00/126 *[RIPng/100] 21:06:15, metric 2, tag 0
> to fe80::fac0:100:11f:9016 via ge-0/3/6.1
```

Issue the following command to verify local spoke site 2002::6e0a:a00/126 in spoke device PE6.

regress@host# **run show route 2002::6e0a:a00/126**

```
2002::6e0a:a00/126 *[RIPng/100] 21:06:15, metric 2, tag 0
> to fe80::fac0:100:21f:9016 via ge-0/3/6.2
```

Issue the following command to verify local spoke site 2002::780a:a00/126 in spoke device PE6.

regress@host# **run show route 2002::780a:a00/126**

```
2002::780a:a00/126 *[RIPng/100] 21:06:14, metric 2, tag 0
> to fe80::fac0:100:31f:9016 via ge-0/3/6.3
```

Meaning The output indicates that the PE device has learned the IPv4 routes from its directly connected branch sites through RIPng and installed them in the **l3vpn_spoke-spoke_routes-vrf1** routing table.

Verify Hub Site IPv4 Routes from Hub Devices to Spoke Devices

Purpose View the central site IPv4 routes in the spoke PE device that are learned from the hub PE device.

Action Issue the following command:

regress@host# **run show route 200.10.10.0/30**

```
l3vpn_spoke-hub_routes-vrf1.inet.0: 103 destinations, 206 routes (103
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

200.10.10.0/30 *[BGP/170] 1d 18:13:40, localpref 100, from 10.255.50.71
                 AS path: 64513 64513 71 I, validation-state: unverified
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
                [BGP/170] 1d 18:08:47, localpref 100, from 10.255.50.73
                 AS path: 64513 64513 71 I, validation-state: unverified
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                 to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                 to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
```

Meaning The output indicates that the spoke PE device has learned the central site routes from the hub PE device through the route reflectors and installed them in the **l3vpn_spoke-hub_routes-vrf1** routing table.

Verify Hub Site IPv6 Routes from Hub Devices to Spoke Devices

Purpose View the central site IPv6 routes in the spoke PE device that have been learned from the hub PE device.

Action Use the following command to show hub site routes in PE6 received from remote hub device PE2.

regress@host# **run show route 2002::c80a:a00/126**

```
2002::c80a:a00/126 *[BGP/170] 1d 18:16:28, localpref 100, from
10.255.50.71
                      AS path: 64513 64513 71 I, validation-state: unverified
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
                    [BGP/170] 1d 18:11:14, localpref 100, from 10.255.50.73
                      AS path: 64513 64513 71 I, validation-state: unverified
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                      to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                      to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
```

Meaning The output indicates that the spoke PE device has learned the central site routes from the hub PE device through the route reflectors using BGP and installed them in the **l3vpn_spoke-hub_routes-vrf1** routing table.

Verify IPv4 Default Routes in the Spoke PE6 Device from Hub Device PE2

Purpose View the default IPv4 route in the **l3vpn_spoke-hub_routes-vrf1** routing table in the spoke PE device.

Action Issue the following command:

regress@host# **run show route 0.0.0.0**

```
l3vpn_spoke-hub_routes-vrf1.inet.0: 103 destinations, 206 routes (103
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0      *[BGP/170] 1d 18:47:30, localpref 100, from 10.255.50.73
                  AS path: 64513 64513 I, validation-state: unverified
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
                [BGP/170] 1d 18:19:55, localpref 100, from 10.255.50.71
                  AS path: 64513 64513 I, validation-state: unverified
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
```

Meaning The output indicates that the spoke PE device has learned the default route from the hub PE device through the route reflectors using BGP and installed them in the **l3vpn_spoke-hub_routes-vrf1** routing table.

Verify IPv6 Default Route in the Spoke PE6 Device from Hub Device PE2

Purpose View the default IPv6 route in the **l3vpn_spoke-hub_routes-vrf1** routing table in the spoke PE device.

Action Issue the following command:

regress@host# **run show route 0::0**

```
l3vpn_spoke-hub_routes-vrf1.inet6.0: 103 destinations, 206 routes (103
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::/0           *[BGP/170] 1d 18:48:56, localpref 100, from 10.255.50.71
                  AS path: 64513 64513 I, validation-state: unverified
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
                [BGP/170] 1d 18:48:56, localpref 100, from 10.255.50.73
                  AS path: 64513 64513 I, validation-state: unverified
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-1
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-2
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-3
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-4
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-5
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-6
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-7
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-8
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-9
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-10
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-11
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-12
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-13
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-14
                  to 1.0.0.49 via ae5.0, label-switched-path r6-to-r2-15
                  to 1.0.0.45 via ae4.0, label-switched-path r6-to-r2-16
```

Meaning The output indicates that the spoke PE device has learned the default route from the hub PE device through the route reflectors using BGP and installed them in the **l3vpn_spoke-hub_routes-vrf1** routing table.

Verify Static IPv4 Default Route in the Spoke PE6 Device

Purpose View the static IPv4 default route in the spoke PE device.

Action Issue the following command:

regress@host# **run show route 0.0.0.0 table l3vpn_spoke-spoke_routes-vrf1.inet.0**

```
l3vpn_spoke-spoke_routes-vrf1.inet.0: 313 destinations, 315 routes (313
active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 1d 18:52:08
                      to table l3vpn_spoke-hub_routes-vrf1.inet.0
```

Meaning The output indicates that the PE6 device has installed a static IPv4 default route in its l3vpn_spoke-spoke_routes-vrf1.inet.0 routing table with a next hop pointing to the l3vpn_spoke-hub_routes-vrf1.inet.0 routing table.

Verify Static IPv6 Default Route in the Spoke PE6 Device

Purpose View the static IPv6 default route in the spoke PE device.

Action Issue the following command:

regress@host# **run show route 0::0 table l3vpn_spoke-spoke_routes-vrf1.inet6.0**

```
l3vpn_spoke-spoke_routes-vrf1.inet6.0: 317 destinations, 321 routes
(317 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[Static/5] 1d 18:53:34
                      to table l3vpn_spoke-hub_routes-vrf1.inet6.0
```

Meaning The output indicates that the PE6 device has installed a static IPv6 default route in its l3vpn_spoke-spoke_routes-vrf1.inet.0 routing table with a next hop pointing to the l3vpn_spoke-hub_routes-vrf1.inet.0 routing table.

Verify Spoke Site IPv4 Routes in Hub PE2 from Remote Spoke Device PE6

Purpose View the branch site IPv4 routes in the hub PE2 device.

Action Issue the following command:

regress@host# **run show route 100.10.10.0/30 table l3vpn_hub-vrf1.inet.0**

```
l3vpn_hub-vrf1.inet.0: 407 destinations, 1013 routes (407 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

100.10.10.0/30 *[BGP/170] 00:01:15, MED 2, localpref 100, from
10.255.50.73
                    AS path: I, validation-state: unverified
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-1
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-2
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-3
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-4
                  > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-5
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-6
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-8
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-9
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-10
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-11
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-12
```

```
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-13
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-14
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-15
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-16
[BGP/170] 00:01:14, MED 2, localpref 100, from 10.255.50.71
                    AS path: I, validation-state: unverified
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-1
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-2
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-3
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-4
                  > to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-5
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-6
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-8
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-9
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-10
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-11
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-12
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-13
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-14
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-15
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-16
```

Meaning The output indicates that the hub PE2 device has received the branch site IPv4 routes from spoke device PE6 through the route reflectors using BGP and has installed those routes in the l3vpn_hub-vrf1.inet.0 table.

Verify Spoke Site IPv6 Routes in Hub PE2 from Remote Spoke Device PE6

Purpose View the branch site IPv6 routes in the hub PE2 device.

Action Issue the following command:

regress@host# **run show route 2002::640a:a00/126 table l3vpn_hub-vrf1.inet6.0**

```
l3vpn_hub-vrf1.inet6.0: 409 destinations, 1015 routes (409 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

2002::640a:a00/126 *[BGP/170] 00:04:22, MED 2, localpref 100, from
10.255.50.73
                    AS path: I, validation-state: unverified
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-1
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-2
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-3
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-4
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-5
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-6
                  > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-7
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-8
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-9
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-10
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-11
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-12
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-13
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-14
                    to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-15
                    to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-16
```

```
                    [BGP/170] 00:04:22, MED 2, localpref 100, from 10.255.50.71
                         AS path: I, validation-state: unverified
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-1
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-2
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-3
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-4
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-5
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-6
                       > to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-7
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-8
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-9
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-10
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-11
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-12
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-13
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-14
                         to 1.0.0.5 via ae0.0, label-switched-path r2-to-r6-15
                         to 1.0.0.21 via ae1.0, label-switched-path r2-to-r6-16
```

Meaning The output indicates that the hub PE2 device has received the branch site IPv6 routes from the spoke PE6 device through the route reflectors using BGP and has installed those routes in the l3vpn_hub-vrf1.inet6.0 table.

Verify Local Hub Site IPv4 Route in Hub Device PE2

Purpose View the central site IPv4 routes in the hub PE2 device.

Action Issue the following command:

regress@host# **run show route 200.10.10.0/30 table l3vpn_hub-any-any-vrf1.inet.0**

```
l3vpn_hub-any-any-vrf1.inet.0: 409 destinations, 711 routes (409
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both

200.10.10.0/30     *[BGP/170] 1d 19:21:32, MED 1, localpref 100
                      AS path: 71 I, validation-state: unverified
                    > to 1.0.0.66 via ae8.1
```

Meaning The output indicates that the PE2 device has received the central site routes from its directly attached CE device and has installed the routes in the l3vpn_hub-any-any-vrf1.inet.0 table.

Verify Local Hub Site IPv6 Route in Hub Device PE2

Purpose View the central site IPv6 routes in the hub PE2 device.

Action Issue the following command:

regress@host# **run show route 2002::c80a:a00/126 table l3vpn_hub-any-any-vrf1.inet6.0**

```
l3vpn_hub-any-any-vrf1.inet6.0: 412 destinations, 715 routes (412
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c80a:a00/126 *[BGP/170] 1d 19:23:23, MED 1, localpref 100
                      AS path: 71 I, validation-state: unverified
                    > to 2002::100:42 via ae8.1
```

Meaning The output indicates that the PE2 device has received the central site routes from its directly attached CE device and has installed the routes in the l3vpn_hub-any-any-vrf1.inet6.0 table.

Verify Spoke-and-Hub Site Routes in the SRX Series from Hub Device PE2

Purpose View the branch and central site routes in the SRX Series device.

Action Issue the following command for the spoke site IPv4 route:

regress@host# **run show route 100.10.10.0/30 table vr1.inet.0**

```
vr1.inet.0: 412 destinations, 818 routes (412 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

100.10.10.0/30     *[BGP/170] 01:00:04, localpref 100
                      AS path: 70 I, validation-state: unverified
                    > to 1.0.0.85 via ge-0/0/12.2
```

Issue the following command for the hub site IPv4 route:

regress@host# **run show route 200.10.10.0/30 table vr1.inet.0**

```
vr1.inet.0: 412 destinations, 818 routes (412 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

200.10.10.0/30     *[BGP/170] 1d 19:25:50, localpref 100
                      AS path: 70 71 I, validation-state: unverified
                    > to 1.0.0.97 via ge-0/0/12.3
```

Issue the following command for the spoke site IPv6 route:

regress@host# **run show route 2002::640a:a00/126 table vr1.inet6.0**

```
vr1.inet6.0: 416 destinations, 824 routes (416 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

2002::640a:a00/126 *[BGP/170] 01:01:27, localpref 100
                      AS path: 70 I, validation-state: unverified
                    > to 2002::100:55 via ge-0/0/12.2
```

Issue the following command for the hub site IPv6 route:

regress@host# **run show route 2002::c80a:a00/126 table vr1.inet6.0**

```
vr1.inet6.0: 416 destinations, 824 routes (416 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

2002::c80a:a00/126 *[BGP/170] 1d 19:26:43, localpref 100
                      AS path: 70 71 I, validation-state: unverified
                    > to 2002::100:61 via ge-0/0/12.3
```

Meaning The output indicates that the SRX Series device has learned the branch and central site routes from the PE2 device and installed them in the vr1.inet.0 and vrf1.inet6.0 tables.

Verify Default Routes in the SRX Series from Hub Device PE2

Purpose View the default route in the SRX Series device.

Action Issue the following command:

regress@host# **run show route 0.0.0.0 table vr1.inet.0**

```
vr1.inet.0: 412 destinations, 818 routes (412 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 1d 20:11:15, localpref 100
                      AS path: 70 I, validation-state: unverified
                    > to 1.0.0.81 via ge-0/0/12.1
```

Issue the following command:

regress@host# **run show route 0::0 table vr1.inet6.0**

```
vr1.inet6.0: 416 destinations, 824 routes (416 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[BGP/170] 1d 20:12:00, localpref 100
                      AS path: 70 I, validation-state: unverified
                    > to 2002::100:51 via ge-0/0/12.1
```

Meaning The output indicates that the SRX Series device has received IPv4 and IPv6 default routes from the PE2 device using BGP and installed the routes in the vrf1 routing table.

Verify Spoke-and-Hub Site Routes in Device PE2 from the SRX Series Device

Purpose View the branch and central site routes in the PE2 device.

Action Issue the following command:

regress@host# **run show route 100.10.10.0/30 table l3vpn_hub-any-any-vrf1.inet.0**

```
l3vpn_hub-any-any-vrf1.inet.0: 409 destinations, 711 routes (409
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both

100.10.10.0/30     *[BGP/170] 01:04:42, localpref 100
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 1.0.0.98 via ge-1/1/0.3
```

Issue the following command:

regress@host# **run show route 200.10.10.0/30 table l3vpn_hub-vrf1.inet.0**

```
l3vpn_hub-vrf1.inet.0: 407 destinations, 1013 routes (407 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

200.10.10.0/30     *[BGP/170] 1d 19:30:11, localpref 100
                      AS path: 64513 64513 71 I, validation-state:
unverified
                    > to 1.0.0.86 via ge-1/1/0.2
```

Issue the following command:

regress@host# **run show route 2002::640a:a00/126 table l3vpn_hub-any-any-vrf1.inet6.0**

```
l3vpn_hub-any-any-vrf1.inet6.0: 412 destinations, 715 routes (412
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
2002::640a:a00/126 *[BGP/170] 01:06:46, localpref 100
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 2002::100:62 via ge-1/1/0.3
```

Issue the following command:

regress@host# **run show route 2002::c80a:a00/126 table l3vpn_hub-vrf1.inet6.0**

```
l3vpn_hub-vrf1.inet6.0: 409 destinations, 1015 routes (409 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

2002::c80a:a00/126 *[BGP/170] 1d 19:32:37, localpref 100
                      AS path: 64513 64513 71 I, validation-state:
unverified
                    > to 2002::100:56 via ge-1/1/0.2
```

Meaning The output indicates that the PE2 device has received branch and central site routes from the SRX Series device. It has installed the branch site routes in the l3vpn_hub-any-any-vrf1.inet.0 table and central site routes in the l3vpn_hub-vrf1.inet.0 table.

Verify the Default Route in Hub Device PE2

Purpose View the default route in the PE2 device.

Action Issue the following command:

regress@host# **run show route 0.0.0.0**

```
inet.0: 309116 destinations, 595193 routes (309107 active, 0 holddown,
9 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 1d 20:08:08 -> Static default route
configured in global table
                      Discard

l3vpn_hub-any-any-vrf1.inet.0: 409 destinations, 711 routes (409
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 1d 20:05:50, localpref 100 -> Default
route learned from SRX
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 1.0.0.98 via ge-1/1/0.3

l3vpn_hub-vrf1.inet.0: 407 destinations, 1013 routes (407 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 1d 20:05:54, localpref 100 -> Default
route learned from SRX
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 1.0.0.86 via ge-1/1/0.2
```

Issue the following command:

regress@host# **run show route 0::0**

```
inet6.0: 308252 destinations, 594357 routes (308252 active, 0 holddown,
0 hidden)
```

```
+ = Active Route, - = Last Active, * = Both

::/0               *[Static/5] 1d 20:09:57
                      Discard

l3vpn_hub-any-any-vrf1.inet6.0: 412 destinations, 715 routes (412
active, 0 holddown, 101 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[BGP/170] 1d 20:07:47, localpref 100
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 2002::100:62 via ge-1/1/0.3

l3vpn_hub-vrf1.inet6.0: 409 destinations, 1015 routes (409 active, 0
holddown, 303 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[BGP/170] 1d 20:07:47, localpref 100
                      AS path: 64513 64513 I, validation-state:
unverified
                    > to 2002::100:56 via ge-1/1/0.2
```

Meaning The output indicates that the PE2 device has installed a static default route in its global table. It has also received default routes from the SRX Series device through BGP and has installed them in the l3vpn_hub-any-any-vrf1 and l3vpn_hub-vrf1 routing tables.

Verify the Static Destination NAT Pool Route in the SRX Series Device

Purpose View the static destination NAT pool route in the SRX Series device.

Action Issue the following command:

regress@host# **run show route 1.1.1.96/28 table vr1.inet.0**

```
vr1.inet.0: 412 destinations, 818 routes (412 active, 0 holddown, 406
hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.96/28        *[Static/5] 1d 20:17:10
                      Discard
```

Meaning The output indicates that the SRX Series device has installed a static destination NAT pool route in its vr1 routing table.

Verify the Destination NAT Pool Route in Hub Device PE2 from the SRX Series Device

Purpose View the static destination NAT pool route in the PE2 device.

Action Issue the following command:

regress@host# **run show route 1.1.1.96/28**

```
inet.0: 309116 destinations, 595193 routes (309107 active, 0 holddown,
9 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.96/28        *[BGP/170] 1d 20:16:21, localpref 100
                      AS path: 64513 I, validation-state: unverified
                    > to 1.0.0.82 via ge-1/1/0.1
```

Meaning The output indicates that the PE2 device has received the static destination NAT pool route from the SRX Series device using BGP and has installed the route in its global routing table.

Verify Spoke Device PE6 Interface Statistics

Purpose View the customer-facing interface statistics in the spoke PE device.

Action Issue the following command:

regress@host# **run show interfaces ge-0/3/6 detail**

```
Logical interface ge-0/3/6.1 (Index 338) (SNMP ifIndex 6259)
(Generation 23711)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.1 ]  Encapsulation:
ENET2
    Traffic statistics:
     Input  bytes  :             31607912
     Output bytes  :             52707178
     Input  packets:               135159
     Output packets:               225295
     IPv6 transit statistics:
      Input  bytes  :            15792784
      Output bytes  :            26348400
      Input  packets:               67497
      Output packets:              112600
    Local statistics:
     Input  bytes  :                13108
     Output bytes  :                22962
     Input  packets:                   27
     Output packets:                   43
    Transit statistics:
     Input  bytes  :             31594804              4414224 bps
     Output bytes  :             52684216              7077032 bps
     Input  packets:               135132                 2363 pps
     Output packets:               225252                 3782 pps
     IPv6 transit statistics:
      Input  bytes  :            15792784
      Output bytes  :            26348400
      Input  packets:               67497
      Output packets:              112600
Logical interface ge-0/3/6.2 (Index 339) (SNMP ifIndex 6258)
(Generation 23712)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.2 ]  Encapsulation:
ENET2
    Traffic statistics:
     Input  bytes  :             31666756
     Output bytes  :             52689488
     Input  packets:               135409
     Output packets:               225214
     IPv6 transit statistics:
      Input  bytes  :            15817230
      Output bytes  :            26334594
      Input  packets:               67595
      Output packets:              112541
    Local statistics:
     Input  bytes  :                10820
     Output bytes  :                22822
     Input  packets:                   23
     Output packets:                   37
    Transit statistics:
     Input  bytes  :             31655936              4420968 bps
     Output bytes  :             52666666              6912272 bps
     Input  packets:               135386                 2363 pps
     Output packets:               225177                 3694 pps
```

```
     IPv6 transit statistics:
      Input  bytes  :            15817230
      Output bytes  :            26334594
      Input  packets:               67595
      Output packets:              112541

Logical interface ge-0/3/6.3 (Index 340) (SNMP ifIndex 6257)
(Generation 23713)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.3 ]  Encapsulation:
ENET2
    Traffic statistics:
     Input  bytes  :            31609822
     Output bytes  :            52724564
     Input  packets:              135165
     Output packets:              225378
     IPv6 transit statistics:
      Input  bytes  :            15795200
      Output bytes  :            26346996
      Input  packets:               67500
      Output packets:              112594
    Local statistics:
     Input  bytes  :                10548
     Output bytes  :                18404
     Input  packets:                   22
     Output packets:                   33
    Transit statistics:
     Input  bytes  :            31599274              4511560 bps
     Output bytes  :            52706160              6903768 bps
     Input  packets:              135143                 2412 pps
     Output packets:              225345                 3689 pps
     IPv6 transit statistics:
      Input  bytes  :            15795200
      Output bytes  :            26346996
      Input  packets:               67500
      Output packets:              112594
```

Meaning The output indicates that the spoke PE device is receiving and sending traffic from and to the branch sites.

Verify Hub PE2 Interface Statistics

Purpose View the egress interface statistics in the hub PE device.

Action Issue the following command for the interface connected to the SRX Series device:

regress@host# **run show interfaces ge-1/1/0.2 detail**

```
Logical interface ge-1/1/0.2 (Index 367) (SNMP ifIndex 12048)
(Generation 176)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.2 ]  Encapsulation:
ENET2
    Traffic statistics:
     Input  bytes  :            36891433
     Output bytes  :            22154356
     Input  packets:              157657
     Output packets:               94678
     IPv6 transit statistics:
      Input  bytes  :            18456984
      Output bytes  :            11077092
      Input  packets:               78876
      Output packets:               47338
```

```
  Local statistics:
   Input bytes :         163
   Output bytes :        172
     Input  packets:                    2
     Output packets:                    2
    Transit statistics:
     Input  bytes  :             36891270             21267344 bps
     Output bytes  :             22154184             12814064 bps
     Input  packets:               157655                11360 pps
     Output packets:                94676                 6845 pps
     IPv6 transit statistics:
      Input  bytes  :            18456984
      Output bytes  :            11077092
      Input  packets:               78876
      Output packets:               47338
```

Issue the following command for the interface connected to the SRX Series device:

regress@host# **run show interfaces ge-1/1/0.3 detail**

```
Logical interface ge-1/1/0.3 (Index 366) (SNMP ifIndex 12049)
(Generation 175)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.3 ]  Encapsulation:
ENET2
    Traffic statistics:
     Input  bytes  :              7394158
     Output bytes  :             22131736
     Input  packets:                31601
     Output packets:                94582
     IPv6 transit statistics:
      Input  bytes  :             3697200
      Output bytes  :            11077092
      Input  packets:               15800
      Output packets:               47338
    Local statistics:
     Input  bytes  :                  226
     Output bytes  :                  250
     Input  packets:                    3
     Output packets:                    3
    Transit statistics:
     Input  bytes  :              7393932              4298656 bps
     Output bytes  :             22131486             12753816 bps
     Input  packets:                31598                 2296 pps
     Output packets:                94579                 6812 pps
     IPv6 transit statistics:
      Input  bytes  :             3697200
      Output bytes  :            11077092
      Input  packets:               15800
      Output packets:               47338
```

Issue the following command for the interface connected to the hub CE device:

regress@host# **run show interfaces ae8.1 detail**

```
Logical interface ae8.1 (Index 365) (SNMP ifIndex 720) (Generation 174)
    Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.1 ]  Encapsulation:
ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :        148707       6824      34783658     12772192
        Output:         49628       2266      11598910      4238152
```

```
Link:
  ge-1/0/3.1
    Input :         71425       3277      16699670      6132144
    Output:         25631       1172       5992712      2193216
  ge-1/0/5.1
    Input :         77282       3547      18083988      6640048
    Output:         23997       1094       5606198      2044936
```

Meaning The output indicates that the hub PE device is sending and receiving traffic to and from the SRX Series device and the hub CE device.

Verify SRX Interface Statistics

Purpose View the SRX Series device interface statistics.

Action Issue the following command:

regress@host# **run show interfaces ge-0/0/12.2 detail**

```
Logical interface ge-0/0/12.2 (Index 74) (SNMP ifIndex 681) (Generation
148)
    Flags: SNMP-Traps 0x0 VLAN-Tag [ 0x8100.2 ]  Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :             35031340
     Output bytes  :             58392567
     Input  packets:               149710
     Output packets:               249544
    Local statistics:
     Input  bytes  :                  370
     Output bytes  :                  441
     Input  packets:                    5
     Output packets:                    5
    Transit statistics:
     Input  bytes  :             35030970             12780304 bps
     Output bytes  :             58392126             21315160 bps
     Input  packets:               149705                 6827 pps
     Output packets:               249539                11386 pps
```

Issue the following command:

regress@host# **run show interfaces ge-0/0/12.3 detail**

```
Logical interface ge-0/0/12.3 (Index 75) (SNMP ifIndex 542) (Generation
149)
    Flags: SNMP-Traps 0x0 VLAN-Tag [ 0x8100.3 ]  Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :             35023670
     Output bytes  :             11661218
     Input  packets:               149678
     Output packets:                49838
    Local statistics:
     Input  bytes  :                  422
     Output bytes  :                  530
     Input  packets:                    6
     Output packets:                    6
    Transit statistics:
     Input  bytes  :             35023248             12750368 bps
     Output bytes  :             11660688              4209888 bps
     Input  packets:               149672                 6811 pps
     Output packets:                49832                 2248 pps
```

Meaning The output indicates that the SRX Series device is receiving and sending traffic from and to the hub PE device.

Verify AS Boundary Router Device Interface Statistics

Purpose View the AS boundary router device interface statistics.

Action Issue the following command:

regress@host# **run show interfaces xe-0/1/1.4000 detail**

```
  Logical interface xe-0/1/1.4000 (Index 356) (SNMP ifIndex 818)
(Generation 286)
    Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.4000 ]
Encapsulation: ENET2
    Traffic statistics:
     Input  bytes  :         829986549807
     Output bytes  :        4459902995960
     Input  packets:           3546934734
     Output packets:          19059427546
     IPv6 transit statistics:
      Input  bytes  :         276660062316
      Output bytes  :        2231186283720
      Input  packets:           1182307960
      Output packets:           9534991486
    Local statistics:
     Input  bytes  :              6364245
     Output bytes  :              3258362
     Input  packets:                10862
     Output packets:                10901
    Transit statistics:
     Input  bytes  :         829980185562            254346160 bps
     Output bytes  :        4459899737598           1373466544 bps
     Input  packets:           3546923872              135869 pps
     Output packets:          19059416645              733690 pps
```

Meaning The output indicates that the AS boundary router device is sending and receiving traffic to and from the Internet peer.

About Juniper Networks

Juniper Networks is in the business of network innovation. From devices to data centers, from consumers to cloud providers, Juniper Networks delivers the software, silicon and systems that transform the experience and economics of networking. The company serves customers and partners worldwide. Additional information can be found at **www.juniper.net**.

Corporate and Sales Headquarters
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or +1.408.745.2000
Fax: +1.408.745.2100
www.juniper.net

APAC and EMEA Headquarters
Juniper Networks International B.V.
Boeing Avenue 240
1119 PZ Schiphol-Rijk
Amsterdam, The Netherlands
Phone: +31.0.207.125.700
Fax: +31.0.207.125.701

To purchase Juniper Networks solutions, please contact your Juniper Networks representative at +1-866-298-6428 or authorized reseller.

8020019-002-EN March 2014