ï»¿

Internet Engineering Task Force (IETF)
Request for Comments: 9182
Category: Standards Track
ISSN: 2070-1721

S. Barguil
O. Gonzalez de Dios, Ed.
Telefonica
M. Boucadair, Ed.
Orange
L. Munoz
Vodafone
A. Aguado
Nokia

February 2022

A YANG Network Data Model for Layer 3 VPNs

Abstract

As a complement to the Layer 3 Virtual Private Network Service Model (L3SM), which is used for communication between customers and service providers, this document defines an L3VPN Network Model (L3NM) that can be used for the provisioning of Layer 3 Virtual Private Network (L3VPN) services within a service provider network. The model provides a network-centric view of L3VPN services.

The L3NM is meant to be used by a network controller to derive the configuration information that will be sent to relevant network devices. The model can also facilitate communication between a service orchestrator and a network controller/orchestrator.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9182.

Table of Contents

1. Introduction

[RFC8299] defines a YANG Layer 3 Virtual Private Network Service Model (L3SM) that can be used for communication between customers and service providers. Such a model focuses on describing the customer view of the Virtual Private Network (VPN) services and provides an abstracted view of the customer's requested services. That approach limits the usage of the L3SM to the role of a customer service model (as per [RFC8309]).

This document defines a YANG module called the "L3VPN Network Model" (L3NM). The L3NM is aimed at providing a network-centric view of Layer 3 (L3) VPN services. This data model can be used to facilitate communication between the service orchestrator and the network controller/orchestrator by allowing more network-centric information to be included. It enables such additional capabilities as resource management, or it serves as a multi-domain orchestration interface where logical resources (such as route targets or route distinguishers) must be coordinated.

This document uses the common VPN YANG module defined in [RFC9181].

This document does not obsolete [RFC8299]. These two modules are used for similar objectives but with different scopes and views.

The L3NM YANG module was initially built with a "prune and extend" approach, taking as a starting point the YANG module described in [RFC8299]. Nevertheless, the L3NM is not defined as an augment to the L3SM, because a specific structure is required to meet network-oriented L3 needs.

Some information captured in the L3SM can be passed by the orchestrator in the L3NM (e.g., customer) or be used to feed some

L3NM attributes (e.g., actual forwarding policies). Also, some information captured in the L3SM may be maintained locally within the orchestrator, which is in charge of maintaining the correlation between a customer view and its network instantiation. Likewise, some information captured and exposed using the L3NM can feed the service layer (e.g., capabilities) to drive VPN service order handling and thus the L3SM.

Section 5.1 of [RFC8969] illustrates how the L3NM can be used within the network management automation architecture.

The L3NM does not attempt to address all deployment cases, especially those where L3VPN connectivity is supported through the coordination of different VPNs in different underlying networks. More complex deployment scenarios involving the coordination of different VPN instances and different technologies to provide end-to-end VPN connectivity are addressed by complementary YANG modules, e.g., [YANG-Composed-VPN].

The L3NM focuses on Layer 3 VPNs based on BGP Provider Edges (PEs) as described in [RFC4026], [RFC4110], and [RFC4364]; and Multicast VPNs as described in [RFC6037] and [RFC6513].

The YANG data model in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8299], [RFC8309], and [RFC8453] and uses the terminology defined in those documents.

This document uses the term "network model" as defined in Section 2.1 of [RFC8969].

The meanings of the symbols in the tree diagrams are defined in [RFC8340].

This document makes use of the following terms:

Layer 3 VPN Service Model (L3SM): A YANG data model that describes the service requirements of an L3VPN that interconnects a set of sites from the point of view of the customer. The customer service model does not provide details on the service provider network. The L3VPN customer service model is defined in [RFC8299].

Layer 3 VPN Network Model (L3NM): A YANG data model that describes a VPN service in the service provider network. It contains information on the service provider network and might include allocated resources. It can be used by network controllers to manage and control the VPN service configuration in the service provider network. The corresponding YANG module can be used by a service orchestrator to request a VPN service to a network controller.

Service orchestrator: A functional entity that interacts with the customer of an L3VPN. The service orchestrator interacts with the customer using the L3SM. The service orchestrator is responsible for the Customer Edge to Provider Edge (CE-PE) attachment circuits, the PE selection, and requesting the VPN service to the network controller.

Network orchestrator: A functional entity that is hierarchically intermediate between a service orchestrator and network

controllers. A network orchestrator can manage one or several network controllers.

Network controller: A functional entity responsible for the control and management of the service provider network.

VPN node: An abstraction that represents a set of policies applied on a PE and belonging to a single VPN service. A VPN service involves one or more VPN nodes. As it is an abstraction, the network controller will decide how to implement a VPN node. For example, in a BGP-based VPN, a VPN node could typically be mapped to a Virtual Routing and Forwarding (VRF) instance.

VPN network access: An abstraction that represents the network interfaces that are associated with a given VPN node. Traffic coming from the VPN network access belongs to the VPN. The attachment circuits (bearers) between CEs and PEs are terminated in the VPN network access. A reference to the bearer is maintained to allow keeping the link between the L3SM and L3NM when both models are used in a given deployment.

VPN site: A VPN customer's location that is connected to the service provider network via a CE-PE link, which can access at least one VPN [RFC4176].

VPN service provider: A service provider that offers VPN-related services [RFC4176].

Service provider network: A network that is able to provide VPN-related services.

This document is aimed at modeling BGP PE-based VPNs in a service provider network, so the terms defined in [RFC4026] and [RFC4176] are used in this document as well.

3. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document:

ACL	Access Control List
AS	Autonomous System
ASM	Any-Source Multicast
ASN	AS Number
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
BSR	Bootstrap Router
CE	Customer Edge
CsC	Carriers' Carriers
IGMP	Internet Group Management Protocol
L3NM	L3VPN Network Model
L3SM	L3VPN Service Model
L3VPN	Layer 3 Virtual Private Network
MLD	Multicast Listener Discovery
MSDP	Multicast Source Discovery Protocol
MVPN	Multicast VPN
NAT	Network Address Translation
OAM	Operations, Administration, and Maintenance
OSPF	Open Shortest Path First
PE	Provider Edge
PIM	Protocol Independent Multicast
QoS	Quality of Service
RD	Route Distinguisher
RP	Rendezvous Point
RT	Route Target
SA	Security Association
SSM	Source-Specific Multicast
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding

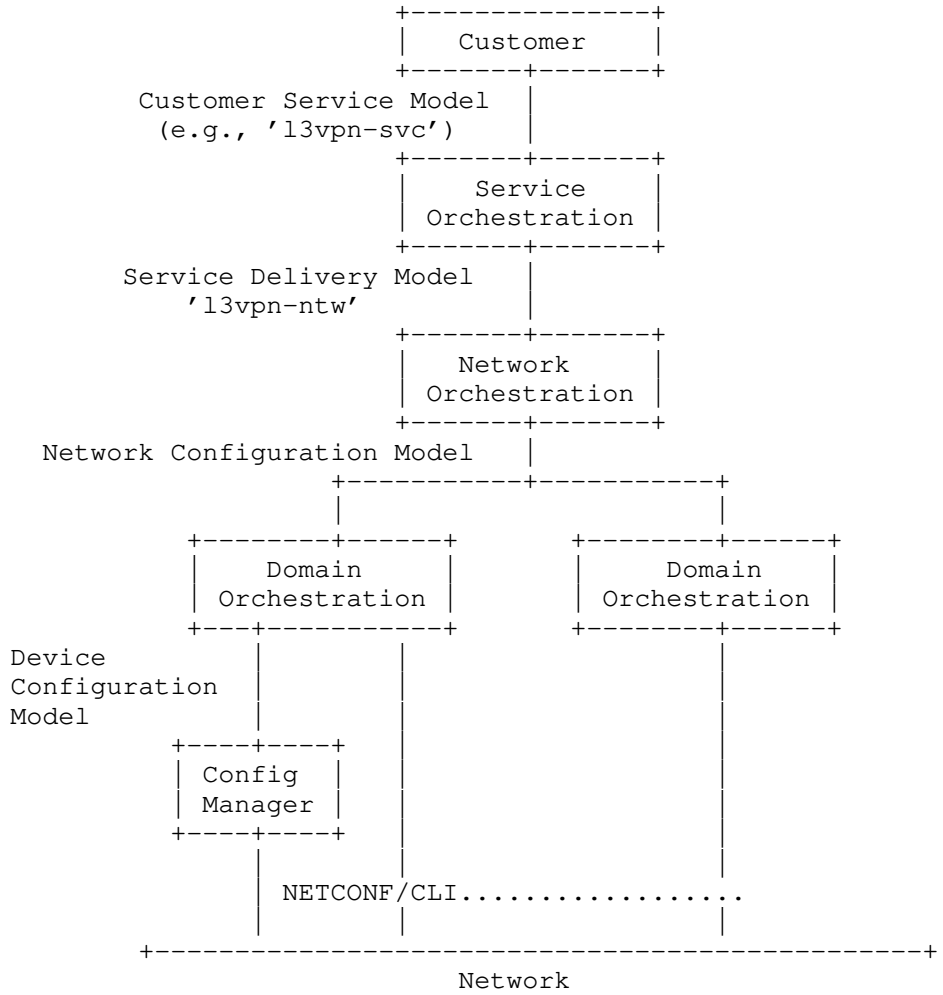
4. L3NM Reference Architecture

Figure 1 depicts the reference architecture for the L3NM. The figure is an expansion of the architecture presented in Section 5 of [RFC8299]; it decomposes the box marked "orchestration" in that section into three separate functional components: service orchestration, network orchestration, and domain orchestration.

Although some deployments may choose to construct a monolithic orchestration component (covering both service and network matters), this document advocates for a clear separation between service and network orchestration components for the sake of better flexibility. Such a design adheres to the L3VPN reference architecture defined in Section 1.3 of [RFC4176]. This separation relies upon a dedicated communication interface between these components and appropriate YANG modules that reflect network-related information. Such information is hidden from customers.

The intelligence for translating customer-facing information into network-centric information (and vice versa) is implementation specific.

The terminology from [RFC8309] is used here to show the distinction between the customer service model, the service delivery model, the network configuration model, and the device configuration model. In that context, the "domain orchestration" and "config manager" roles may be performed by "controllers".



NETCONF: Network Configuration Protocol
CLI: Command-Line Interface

Figure 1: L3NM Reference Architecture

The customer may use a variety of means to request a service that may trigger the instantiation of an L3NM. The customer may use the L3SM or more abstract models to request a service that relies upon an L3VPN service. For example, the customer may supply an IP Connectivity Provisioning Profile (CPP) that characterizes the

requested service [RFC7297], an enhanced VPN (VPN+) service [Enhanced-VPN-Framework], or an IETF network slice service [Network-Slices-Framework].

Note also that both the L3SM and the L3NM may be used in the context of the Abstraction and Control of TE Networks (ACTN) framework [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), the Provisioning Network Controller (PNC) components, and the interfaces where the L3SM and L3NM are used.

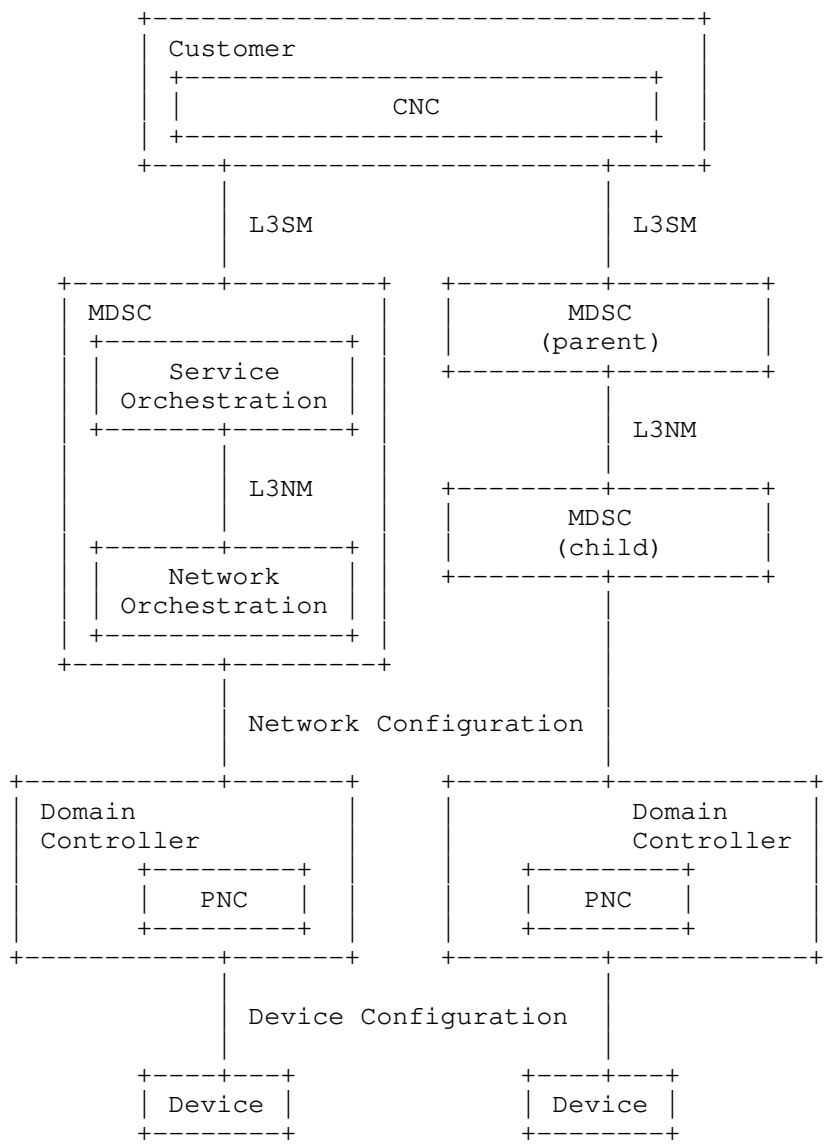


Figure 2: L3SM and L3NM in the Context of the ACTN

5. Relationship to Other YANG Data Models

The "ietf-vpn-common" module [RFC9181] includes a set of identities, types, and groupings that are meant to be reused by VPN-related YANG modules independently of the layer (e.g., Layer 2, Layer 3) and the type of the module (e.g., network model, service model), including future revisions of existing models (e.g., [RFC8299] or [RFC8466]). The L3NM reuses these common types and groupings.

In order to avoid data duplication and to ease passing data between layers when required (service layer to network layer and vice versa), early versions of the L3NM reused many of the data nodes that are defined in [RFC8299]. Nevertheless, that approach was abandoned in favor of the "ietf-vpn-common" module because that initial design was interpreted as if the deployment of the L3NM depends on the L3SM, while this is not the case. For example, a service provider may decide to use the L3NM to build its L3VPN services without exposing the L3SM.

As discussed in Section 4, the L3NM is meant to manage L3VPN services within a service provider network. The module provides a network view of the service. Such a view is only visible within the service provider and is not exposed outside (to customers, for example). The items below discuss how the L3NM interfaces with other YANG modules:

L3SM: The L3NM is not a customer service model.

The internal view of the service (i.e., the L3NM) may be mapped to an external view that is visible to customers: the L3VPN Service Model (L3SM) [RFC8299].

The L3NM can be fed with inputs that are requested by customers. Such requests typically rely upon an L3SM template. Concretely, some parts of the L3SM module can be directly mapped to the L3NM, while other parts are generated as a function of the requested service and local guidelines. Some other parts are local to the service provider and do not map directly to the L3SM.

Note that using the L3NM within a service provider does not assume, nor does it preclude, exposing the VPN service via the L3SM. This is deployment specific. Nevertheless, the design of the L3NM tries to align as much as possible with the features supported by the L3SM to ease the grafting of both the L3NM and the L3SM for the sake of highly automated VPN service provisioning and delivery.

Network Topology Modules: An L3VPN involves nodes that are part of a topology managed by the service provider network. The topology can be represented using the network topology YANG module defined in [RFC8345] or its extension, such as a network YANG module for Service Attachment Points (SAPs) [YANG-SAPs].

Device Modules: The L3NM is not a device model.

Once a global VPN service is captured by means of the L3NM, the actual activation and provisioning of the VPN service will involve a variety of device modules to tweak the required functions for the delivery of the service. These functions are supported by the VPN nodes and can be managed using device YANG modules. A non-comprehensive list of such device YANG modules is provided below:

* Routing management [RFC8349].

* BGP [BGP-YANG].

* PIM [PIM-YANG].

* NAT management [RFC8512].

* QoS management [QoS-YANG].

* ACLs [RFC8519].

How the L3NM is used to derive device-specific actions is implementation specific.

6. Sample Uses of the L3NM Data Model

This section provides a non-exhaustive list of examples that illustrate contexts where the L3NM can be used.

6.1. Enterprise Layer 3 VPN Services

Enterprise L3VPNs are one of the most demanded services for carriers; therefore, L3NM can be useful for automating the provisioning and maintenance of these VPNs. Templates and batch processes can be built, and as a result many parameters are needed for the creation from scratch of a VPN that can be abstracted to the upper Software-Defined Networking (SDN) layer [RFC7149] [RFC7426], but some manual

intervention will still be required.

A common function that is supported by VPNs is the addition or removal of VPN nodes. Workflows can use the L3NM in these scenarios to add or prune nodes from the network data model as required.

6.2. Multi-Domain Resource Management

The implementation of L3VPN services that span administratively separated domains (i.e., that are under the administration of different management systems or controllers) requires some network resources to be synchronized between systems. Particularly, resources must be adequately managed in each domain to avoid broken configurations.

For example, route targets (RTs) shall be synchronized between PEs. When all PEs are controlled by the same management system, RT allocation can be performed by that management system. In cases where the service spans multiple management systems, the task of allocating RTs has to be aligned across the domains; therefore, the network model must provide a way to specify RTs. In addition, route distinguishers (RDs) must also be synchronized to avoid collisions of RD allocations between separate management systems. An incorrect allocation might lead to the same RD and IP prefixes being exported by different PEs.

6.3. Management of Multicast Services

Multicast services over L3VPNs can be implemented using dual PIM MVPNs (also known as the draft-rosen model) [RFC6037] or MVPNs based on Multiprotocol BGP (MP-BGP) [RFC6513] [RFC6514]. Both methods are supported and equally effective, but the main difference is that MP-BGP-based MVPNs do not require multicast configuration on the service provider network. MP-BGP MVPNs employ the intra-AS BGP control plane and PIM Sparse Mode [RFC7761] as the data plane. The PIM state information is maintained between PEs using the same architecture that is used for unicast VPNs.

On the other hand, [RFC6037] has limitations, such as reduced options for transport, control plane scalability, availability, operational inconsistency, and the need to maintain state in the backbone. Because of these limitations, MP-BGP MVPNs provide the architectural model that has been taken as the base for implementing multicast services in L3VPNs. In this scenario, BGP is used to autodiscover MVPN PE members and the customer PIM signaling is sent across the provider's core through MP-BGP. The multicast traffic is transported on MPLS Point-to-Multipoint (P2MP) Label Switched Paths (LSPs).

7. Description of the L3NM YANG Module

The L3NM ("ietf-l3vpn-ntw") is defined to manage L3VPNs in a service provider network. In particular, the "ietf-l3vpn-ntw" module can be used to create, modify, and retrieve L3VPN services of a network.

The full tree diagram of the module can be generated using the "pyang" tool [PYANG]. That tree is not included here because it is too long (Section 3.3 of [RFC8340]). Instead, subtrees are provided for the reader's convenience.

7.1. Overall Structure of the Module

The "ietf-l3vpn-ntw" module uses two main containers: 'vpn-profiles' and 'vpn-services' (see Figure 3).

The 'vpn-profiles' container is used by the provider to maintain a set of common VPN profiles that apply to one or several VPN services (Section 7.2).

The 'vpn-services' container maintains the set of VPN services managed within the service provider network. The 'vpn-service' is the data structure that abstracts a VPN service (Section 7.3).

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           ...
           +--rw vpn-nodes
              +--rw vpn-node* [vpn-node-id]
                 ...
                 +--rw vpn-network-accesses
                    +--rw vpn-network-access* [id]
                       ...
```

Figure 3: Overall L3NM Tree Structure

Some of the data nodes are keyed by the address family. For the sake of data representation compactness, it is RECOMMENDED to use the dual-stack address family for data nodes that have the same value for both IPv4 and IPv6. If, for some reason, a data node is present for both dual-stack and IPv4 (or IPv6), the value that is indicated under dual-stack takes precedence over the value that is indicated under IPv4 (or IPv6).

7.2. VPN Profiles

The 'vpn-profiles' container (Figure 4) allows the VPN service provider to define and maintain a set of VPN profiles [RFC9181] that apply to one or several VPN services.

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  +--rw valid-provider-identifiers
     |     +--rw external-connectivity-identifier* [id]
     |     |       {external-connectivity}?
     |     |  +--rw id    string
     |     +--rw encryption-profile-identifier* [id]
     |     |  +--rw id    string
     |     +--rw qos-profile-identifier* [id]
     |     |  +--rw id    string
     |     +--rw bfd-profile-identifier* [id]
     |     |  +--rw id    string
     |     +--rw forwarding-profile-identifier* [id]
     |     |  +--rw id    string
     |     +--rw routing-profile-identifier* [id]
     |        +--rw id    string
     +--rw vpn-services
        ...
```

Figure 4: VPN Profiles Subtree Structure

This document does not make any assumption about the exact definition of these profiles. The exact definition of the profiles is local to each VPN service provider. The model only includes an identifier for these profiles in order to facilitate identifying and binding local policies when building a VPN service. As shown in Figure 4, the following identifiers can be included:

'external-connectivity-identifier': This identifier refers to a profile that defines the external connectivity provided to a VPN service (or a subset of VPN sites). External connectivity may be access to the Internet or restricted connectivity, such as access to a public/private cloud.

'encryption-profile-identifier': An encryption profile refers to a set of policies related to the encryption schemes and setup that can be applied when building and offering a VPN service.

'qos-profile-identifier': A Quality of Service (QoS) profile refers to a set of policies, such as classification, marking, and actions

(e.g., [RFC3644]).

'bfd-profile-identifier': A Bidirectional Forwarding Detection (BFD) profile refers to a set of BFD policies [RFC5880] that can be invoked when building a VPN service.

'forwarding-profile-identifier': A forwarding profile refers to the policies that apply to the forwarding of packets conveyed within a VPN. Such policies may consist, for example, of applying Access Control Lists (ACLs).

'routing-profile-identifier': A routing profile refers to a set of routing policies that will be invoked (e.g., BGP policies) when delivering the VPN service.

7.3. VPN Services

The 'vpn-service' is the data structure that abstracts a VPN service in the service provider network. Each 'vpn-service' is uniquely identified by an identifier: 'vpn-id'. Such a 'vpn-id' is only meaningful locally (e.g., the network controller). The subtree of the 'vpn-services' is shown in Figure 5.

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           +--rw vpn-id                    vpn-common:vpn-id
           +--rw vpn-name?                 string
           +--rw vpn-description?          string
           +--rw customer-name?            string
           +--rw parent-service-id?        vpn-common:vpn-id
           +--rw vpn-type?                 identityref
           +--rw vpn-service-topology?     identityref
           +--rw status
           |  +--rw admin-status
           |  |  +--rw status?          identityref
           |  |  +--rw last-change?   yang:date-and-time
           |  +--ro oper-status
           |     +--ro status?          identityref
           |     +--ro last-change?   yang:date-and-time
           +--rw vpn-instance-profiles
           |  ...
           +--rw underlay-transport
           |  +-- (type)?
           |     +--:(abstract)
           |     |  +--rw transport-instance-id?   string
           |     |  +--rw instance-type?           identityref
           |     +--:(protocol)
           |        +--rw protocol*                identityref
           +--rw external-connectivity
           |                    {vpn-common:external-connectivity}?
           |  +--rw (profile)?
           |     +--:(profile)
           |        +--rw profile-name?            leafref
           +--rw vpn-nodes
              ...
```

Figure 5: VPN Services Subtree Structure

The descriptions of the VPN service data nodes that are depicted in Figure 5 are as follows:

'vpn-id': An identifier that is used to uniquely identify the L3VPN service within the L3NM scope.

'vpn-name': Associates a name with the service in order to facilitate the identification of the service.

'vpn-description': Includes a textual description of the service.

The internal structure of a VPN description is local to each VPN service provider.

'customer-name': Indicates the name of the customer who ordered the service.

'parent-service-id': Refers to an identifier of the parent service (e.g., L3SM, IETF network slice, VPN+) that triggered the creation of the VPN service. This identifier is used to easily correlate the (network) service as built in the network with a service order. A controller can use that correlation to enrich or populate some fields (e.g., description fields) as a function of local deployments.

'vpn-type': Indicates the VPN type. The values are taken from [RFC9181]. For the L3NM, this is typically set to "BGP/MPLS L3VPN", but other values may be defined to support specific Layer 3 VPN capabilities (e.g., [RFC9136]).

'vpn-service-topology': Indicates the network topology for the service: 'hub-spoke', 'any-to-any', or 'custom'. The network implementation of this attribute is defined by the correct usage of import and export targets (Section 4.3.5 of [RFC4364]).

'status': Used to track the service status of a given VPN service. Both operational status and administrative status are maintained together with a timestamp. For example, a service can be created but not put into effect.

Administrative status and operational status can be used as a trigger to detect service anomalies. For example, a service that is declared active at the service layer but is still inactive at the network layer may be an indication that network provision actions are needed to align the observed service status with the expected service status.

'vpn-instance-profiles': Defines reusable parameters for the same 'vpn-service'.

More details are provided in Section 7.4.

'underlay-transport': Describes the preference for the transport technology to carry the traffic of the VPN service. This preference is especially useful in networks with multiple domains and Network-to-Network Interface (NNI) types. The underlay transport can be expressed as an abstract transport instance (e.g., an identifier of a VPN+ instance, a virtual network identifier, or a network slice name) or as an ordered list of the actual protocols to be enabled in the network.

A rich set of protocol identifiers that can be used to refer to an underlay transport are defined in [RFC9181].

'external-connectivity': Indicates whether/how external connectivity is provided to the VPN service. For example, a service provider may provide external connectivity to a VPN customer (e.g., to a public cloud). Such a service may involve tweaking both filtering and NAT rules (e.g., binding a Virtual Routing and Forwarding (VRF) interface with a NAT instance as discussed in Section 2.10 of [RFC8512]). These value-added features may be bound to all, or a subset of, network accesses. Some of these value-added features may be implemented in a PE or in nodes other than PEs (e.g., a P node or even a dedicated node that hosts the NAT function).

Only a pointer to a local profile that defines the external-connectivity feature is supported in this document.

'vpn-node': An abstraction that represents a set of policies applied to a network node and belonging to a single 'vpn-service'. A VPN service is typically built by adding instances of 'vpn-node' to

the 'vpn-nodes' container.

A 'vpn-node' contains 'vpn-network-accesses', which are the interfaces attached to the VPN by which the customer traffic is received. Therefore, the customer sites are connected to the 'vpn-network-accesses'.

Note that because this is a network data model, information about customers' sites is not required in the model. Rather, such information is relevant in the L3SM. Whether that information is included in the L3NM, e.g., to populate the various 'description' data nodes, is implementation specific.

More details are provided in Section 7.5.

7.4. VPN Instance Profiles

VPN instance profiles are meant to factorize data nodes that are used at many levels of the model. Generic VPN instance profiles are defined at the VPN service level and then called at the VPN node and VPN network access levels. Each VPN instance profile is identified by 'profile-id'. This identifier is then referenced for one or multiple VPN nodes (Section 7.5) so that the controller can identify generic resources (e.g., RTs and RDs) to be configured for a given VRF instance.

The subtree of the 'vpn-instance-profiles' is shown in Figure 6.

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           +--rw vpn-id                    vpn-common:vpn-id
           ...
           +--rw vpn-instance-profiles
           |  +--rw vpn-instance-profile* [profile-id]
           |     +--rw profile-id                 string
           |     +--rw role?                      identityref
           |     +--rw local-as?                  inet:as-number
           |     |      {vpn-common:rtg-bgp}?
           |     +--rw (rd-choice)?
           |     |  +--:(directly-assigned)
           |     |  |  +--rw rd?
           |     |  |         rt-types:route-distinguisher
           |     |  +--:(directly-assigned-suffix)
           |     |  |  +--rw rd-suffix?           uint16
           |     |  +--:(auto-assigned)
           |     |  |  +--rw rd-auto
           |     |  |     +--rw (auto-mode)?
           |     |  |     |  +--:(from-pool)
           |     |  |     |  |  +--rw rd-pool-name?   string
           |     |  |     |  +--:(full-auto)
           |     |  |     |     +--rw auto?           empty
           |     |  |     +--ro auto-assigned-rd?
           |     |  |           rt-types:route-distinguisher
           |     |  +--:(auto-assigned-suffix)
           |     |  |  +--rw rd-auto-suffix
           |     |  |     +--rw (auto-mode)?
           |     |  |     |  +--:(from-pool)
           |     |  |     |  |  +--rw rd-pool-name?        string
           |     |  |     |  +--:(full-auto)
           |     |  |     |     +--rw auto?                empty
           |     |  |     +--ro auto-assigned-rd-suffix?   uint16
           |     |  +--:(no-rd)
           |     |     +--rw no-rd?               empty
           |     +--rw address-family* [address-family]
           |     |  +--rw address-family          identityref
           |     |  +--rw vpn-targets
           |     |  |  +--rw vpn-target* [id]
           |     |  |  |  +--rw id                  uint8
```

```
|      |  |  |  +--rw route-targets* [route-target]
|      |  |  |  |  +--rw route-target
|      |  |  |  |       rt-types:route-target
|      |  |  |  +--rw route-target-type
|      |  |  |          rt-types:route-target-type
|      |  |  +--rw vpn-policies
|      |  |     +--rw import-policy?   string
|      |  |     +--rw export-policy?   string
|      |  +--rw maximum-routes* [protocol]
|      |     +--rw protocol          identityref
|      |     +--rw maximum-routes?   uint32
|      +--rw multicast {vpn-common:multicast}?
|         ...
```

Figure 6: Subtree Structure of VPN Instance Profiles

The descriptions of the listed data nodes are as follows:

'profile-id': Used to uniquely identify a VPN instance profile.

'role': Indicates the role of the VPN instance profile in the VPN. Role values are defined in [RFC9181] (e.g., 'any-to-any-role', 'spoke-role', 'hub-role').

'local-as': Indicates the Autonomous System Number (ASN) that is configured for the VPN node.

'rd': As defined in [RFC9181], the following RD assignment modes are supported: direct assignment, full automatic assignment, automatic assignment from a given pool, and no assignment. For illustration purposes, the following modes can be used in the deployment cases:

'directly-assigned': The VPN service provider (service orchestrator) assigns RDs explicitly. This case will fit with a brownfield scenario where some existing services need to be updated by the VPN service provider.

'full-auto': The network controller auto-assigns RDs. This can apply for the deployment of new services.

'no-rd': The VPN service provider (service orchestrator) explicitly wants no RD to be assigned. This case can be used for CE testing within the network or for troubleshooting proposes.

Also, the module accommodates deployments where only the Assigned Number subfield of RDs (Section 4.2 of [RFC4364]) is assigned from a pool while the Administrator subfield is set to, for example, the Router ID that is assigned to a VPN node. The module supports these modes for managing the Assigned Number subfield: explicit assignment, auto-assignment from a pool, and full auto-assignment.

'address-family': Includes a set of data nodes per address family:

'address-family': Identifies the address family. It can be set to 'ipv4', 'ipv6', or 'dual-stack'.

'vpn-targets': Specifies RT import/export rules for the VPN service (Section 4.3 of [RFC4364]).

'maximum-routes': Indicates the maximum number of prefixes that the VPN node can accept for a given routing protocol. If 'protocol' is set to 'any', this means that the maximum value applies to each active routing protocol.

'multicast': Enables multicast traffic in the VPN service. Refer to Section 7.7.

7.5. VPN Nodes

The 'vpn-node' is an abstraction that represents a set of common

policies applied on a given network node (typically, a PE) and belonging to one L3VPN service. The 'vpn-node' includes a parameter to indicate the network node on which it is applied. In the case that the 'ne-id' points to a specific PE, the 'vpn-node' will likely be mapped to a VRF instance in the node. However, the model also allows pointing to an abstract node. In this case, the network controller will decide how to split the 'vpn-node' into VRF instances.

The VPN node subtree structure is shown in Figure 7.

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           ...
           +--rw vpn-nodes
              +--rw vpn-node* [vpn-node-id]
                 +--rw vpn-node-id                vpn-common:vpn-id
                 +--rw description?               string
                 +--rw ne-id?                     string
                 +--rw local-as?                  inet:as-number
                 |       {vpn-common:rtg-bgp}?
                 +--rw router-id?                 rt-types:router-id
                 +--rw active-vpn-instance-profiles
                 |  +--rw vpn-instance-profile* [profile-id]
                 |     +--rw profile-id                 leafref
                 |     +--rw router-id* [address-family]
                 |     |  +--rw address-family    identityref
                 |     |  +--rw router-id?        inet:ip-address
                 |     +--rw local-as?            inet:as-number
                 |     |     {vpn-common:rtg-bgp}?
                 |     +--rw (rd-choice)?
                 |     |  ....
                 |     +--rw address-family* [address-family]
                 |     |  +--rw address-family          identityref
                 |     |  |  ...
                 |     |  +--rw vpn-targets
                 |     |  |  ...
                 |     |  +--rw maximum-routes* [protocol]
                 |     |     ...
                 |     +--rw multicast {vpn-common:multicast}?
                 |        ...
                 +--rw msdp {msdp}?
                 |  +--rw peer?            inet:ipv4-address
                 |  +--rw local-address?   inet:ipv4-address
                 |  +--rw status
                 |     +--rw admin-status
                 |     |  +--rw status?         identityref
                 |     |  +--rw last-change?   yang:date-and-time
                 |     +--ro oper-status
                 |        +--ro status?         identityref
                 |        +--ro last-change?   yang:date-and-time
                 +--rw groups
                 |  +--rw group* [group-id]
                 |     +--rw group-id    string
                 +--rw status
                 |  +--rw admin-status
                 |  |  +--rw status?         identityref
                 |  |  +--rw last-change?   yang:date-and-time
                 |  +--ro oper-status
                 |     +--ro status?         identityref
                 |     +--ro last-change?   yang:date-and-time
                 +--rw vpn-network-accesses
                    ...
```

Figure 7: VPN Node Subtree Structure

The descriptions of the 'vpn-node' data nodes (Figure 7) are as follows:

'vpn-node-id': An identifier that uniquely identifies a node that enables a VPN network access.

'description': Provides a textual description of the VPN node.

'ne-id': Includes a unique identifier of the network element where the VPN node is deployed.

'local-as': Indicates the ASN that is configured for the VPN node.

'router-id': Indicates a 32-bit number that is used to uniquely identify a router within an AS.

'active-vpn-instance-profiles': Lists the set of active VPN instance profiles for this VPN node. Concretely, one or more VPN instance profiles that are defined at the VPN service level can be enabled at the VPN node level; each of these profiles is uniquely identified by means of 'profile-id'. The structure of 'active-vpn-instance-profiles' is the same as the structure discussed in Section 7.4, except that the structure of 'active-vpn-instance-profiles' includes 'router-id' but does not include the 'role' leaf. The value of 'router-id' indicated under 'active-vpn-instance-profiles' takes precedence over the 'router-id' under the 'vpn-node' for the indicated address family. For example, Router IDs can be configured per address family. This capability can be used, for example, to configure an IPv6 address as a Router ID when such a capability is supported by involved routers.

Values defined in 'active-vpn-instance-profiles' override the values defined at the VPN service level. An example is shown in Appendix A.3.

'msdp': For redundancy purposes, the Multicast Source Discovery Protocol (MSDP) [RFC3618] may be enabled and used to share state information about sources between multiple Rendezvous Points (RPs). The purpose of MSDP in this context is to enhance the robustness of the multicast service. MSDP may be configured on non-RP routers; this is useful in a domain that does not support multicast sources but does support multicast transit.

'groups': Lists the groups to which a VPN node belongs [RFC9181]. For example, the 'group-id' is used to associate redundancy or protection constraints with VPN nodes.

'status': Tracks the status of a node involved in a VPN service. Both operational status and administrative status are maintained. A mismatch between the administrative status vs. the operational status can be used as a trigger to detect anomalies.

'vpn-network-accesses': Represents the point to which sites are connected.

Note that unlike the L3SM, the L3NM does not need to model the customer site -- only the points that receive traffic from the site (i.e., the PE side of Provider Edge to Customer Edge (PE-CE) connections). Hence, the VPN network access contains the connectivity information between the provider's network and the customer premises. The VPN profiles ('vpn-profiles') have a set of routing policies that can be applied during the service creation.

See Section 7.6 for more details.

7.6. VPN Network Accesses

The 'vpn-network-access' includes a set of data nodes that describe the access information for the traffic that belongs to a particular L3VPN (Figure 8).

...

```
+--rw vpn-nodes
   +--rw vpn-node* [vpn-node-id]
      ...
      +--rw vpn-network-accesses
         +--rw vpn-network-access* [id]
            +--rw id                          vpn-common:vpn-id
            +--rw interface-id?               string
            +--rw description?                string
            +--rw vpn-network-access-type?    identityref
            +--rw vpn-instance-profile?       leafref
            +--rw status
            |  +--rw admin-status
            |  |  +--rw status?          identityref
            |  |  +--rw last-change?    yang:date-and-time
            |  +--ro oper-status
            |     +--ro status?          identityref
            |     +--ro last-change?    yang:date-and-time
            +--rw connection
            |  ...
            +--rw ip-connection
            |  ...
            +--rw routing-protocols
            |  ...
            +--rw oam
            |  ...
            +--rw security
            |  ...
            +--rw service
               ...
```

Figure 8: VPN Network Access Subtree Structure

A 'vpn-network-access' (Figure 8) includes the following data nodes:

'id': An identifier of the VPN network access.

'interface-id': Indicates the physical or logical interface on which the VPN network access is bound.

'description': Includes a textual description of the VPN network access.

'vpn-network-access-type': Used to select the type of network interface to be deployed in the devices. The available defined values are as follows:

- 'point-to-point': Represents a direct connection between the endpoints. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.
- 'multipoint': Represents a multipoint connection between the customer site and the PEs. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.
- 'irb': Represents a connection coming from an L2VPN service. An identifier of such a service ('l2vpn-id') may be included in the 'connection' container, as depicted in Figure 9 (Section 7.6.1). The controller must keep the relationship between the logical tunnels or bridges on the devices with the 'id' of the 'vpn-network-access'.
- 'loopback': Represents the creation of a logical interface on a device. An example that illustrates how a loopback interface can be used in the L3NM is provided in Appendix A.2.

'vpn-instance-profile': Provides a pointer to an active VPN instance profile at the VPN node level. Referencing an active VPN instance profile implies that all associated data nodes will be inherited by the VPN network access. However, some inherited data nodes

(e.g., multicast) can be overridden at the VPN network access level. In such a case, adjusted values take precedence over inherited values.

'status': Indicates both operational status and administrative status of a VPN network access.

'connection': Represents and groups the set of Layer 2 connectivity from where the traffic of the L3VPN in a particular VPN network access is coming. See Section 7.6.1.

'ip-connection': Contains Layer 3 connectivity information on a VPN network access (e.g., IP addressing). See Section 7.6.2.

'routing-protocols': Includes the CE-PE routing configuration information. See Section 7.6.3.

'oam': Specifies the Operations, Administration, and Maintenance (OAM) mechanisms used for a VPN network access. See Section 7.6.4.

'security': Specifies the authentication and the encryption to be applied for a given VPN network access. See Section 7.6.5.

'service': Specifies the service parameters (e.g., QoS, multicast) to apply for a given VPN network access. See Section 7.6.6.

7.6.1. Connection

The 'connection' container represents the Layer 2 connectivity to the L3VPN for a particular VPN network access. As shown in the tree depicted in Figure 9, the 'connection' container defines protocols and parameters to enable such connectivity at Layer 2.

The traffic can enter the VPN with or without encapsulation (e.g., VLAN, QinQ). The 'encapsulation' container specifies the Layer 2 encapsulation to use (if any) and allows the configuration of the relevant tags.

The interface that is attached to the L3VPN is identified by the 'interface-id' at the 'vpn-network-access' level. From a network model perspective, it is expected that the 'interface-id' is sufficient to identify the interface. However, specific Layer 2 sub-interfaces may be required to be configured in some implementations/ deployments. Such a Layer-2-specific interface can be included in 'l2-termination-point'.

If a Layer 2 tunnel is needed to terminate the service in the CE-PE connection, the 'l2-tunnel-service' container is used to specify the required parameters to set such a tunneling service (e.g., a Virtual Private LAN Service (VPLS) or a Virtual eXtensible Local Area Network (VXLAN)). An identity called 'l2-tunnel-type' is defined for Layer 2 tunnel selection. The container can also identify the pseudowire (Section 6.1 of [RFC8077]).

As discussed in Section 7.6, 'l2vpn-id' is used to identify the L2VPN service that is associated with an Integrated Routing and Bridging (IRB) interface.

To accommodate implementations that require internal bridging, a local bridge reference can be specified in 'local-bridge-reference'. Such a reference may be a local bridge domain.

A site, as per [RFC4176], represents a VPN customer's location that is connected to the service provider network via a CE-PE link, which can access at least one VPN. The connection from the site to the service provider network is the bearer. Every site is associated with a list of bearers. A bearer is the Layer 2 connection with the site. In the L3NM, it is assumed that the bearer has been allocated by the service provider at the service orchestration stage. The bearer is associated with a network element and a port. Hence, a

bearer is just a 'bearer-reference' to allow the association between a service request (e.g., the L3SM) and the L3NM.

The L3NM can be used to create a Link Aggregation Group (LAG) interface for a given L3VPN service ('lag-interface') [IEEE802.1AX]. Such a LAG interface can be referenced under 'interface-id' (Section 7.6).

```
...
+--rw connection
|  +--rw encapsulation
|  |  +--rw type?              identityref
|  |  +--rw dot1q
|  |  |  +--rw tag-type?   identityref
|  |  |  +--rw cvlan-id?   uint16
|  |  +--rw priority-tagged
|  |  |  +--rw tag-type?   identityref
|  |  +--rw qinq
|  |     +--rw tag-type?   identityref
|  |     +--rw svlan-id    uint16
|  |     +--rw cvlan-id    uint16
|  +--rw (l2-service)?
|  |  +--:(l2-tunnel-service)
|  |  |  +--rw l2-tunnel-service
|  |  |     +--rw type?         identityref
|  |  |     +--rw pseudowire
|  |  |     |  +--rw vcid?      uint32
|  |  |     |  +--rw far-end?   union
|  |  |     +--rw vpls
|  |  |     |  +--rw vcid?      uint32
|  |  |     |  +--rw far-end*   union
|  |  |     +--rw vxlan
|  |  |        +--rw vni-id             uint32
|  |  |        +--rw peer-mode?         identityref
|  |  |        +--rw peer-ip-address*   inet:ip-address
|  |  +--:(l2vpn)
|  |     +--rw l2vpn-id?            vpn-common:vpn-id
|  +--rw l2-termination-point?      string
|  +--rw local-bridge-reference?    string
|  +--rw bearer-reference?          string
|  |       {vpn-common:bearer-reference}?
|  +--rw lag-interface {vpn-common:lag-interface}?
|     +--rw lag-interface-id?   string
|     +--rw member-link-list
|        +--rw member-link* [name]
|           +--rw name    string
...
```

Figure 9: Connection Subtree Structure

7.6.2. IP Connection

This container is used to group Layer 3 connectivity information, particularly the IP addressing information, of a VPN network access. The allocated address represents the PE interface address configuration. Note that a distinct Layer 3 interface other than the interface indicated under the 'connection' container may be needed to terminate the Layer 3 service. The identifier of such an interface is included in 'l3-termination-point'. For example, this data node can be used to carry the identifier of a bridge domain interface.

As shown in Figure 10, the 'ip-connection' container can include IPv4, IPv6, or both if dual-stack is enabled.

```
...
+--rw vpn-network-accesses
   +--rw vpn-network-access* [id]
      ...
      +--rw ip-connection
      |  +--rw l3-termination-point?     string
      |  +--rw ipv4 {vpn-common:ipv4}?
```

```
|  |  ...
|  +--rw ipv6 {vpn-common:ipv6}?
|     ...
...
```

Figure 10: IP Connection Subtree Structure

For both IPv4 and IPv6, the IP connection supports three IP address assignment modes for customer addresses: provider DHCP, DHCP relay, and static addressing. Note that for the IPv6 case, Stateless Address Autoconfiguration (SLAAC) [RFC4862] can be used. For both IPv4 and IPv6, 'address-allocation-type' is used to indicate the IP address allocation mode to activate for a given VPN network access.

When 'address-allocation-type' is set to 'provider-dhcp', DHCP assignments can be made locally or by an external DHCP server. Such behavior is controlled by setting 'dhcp-service-type'.

Figure 11 shows the structure of the dynamic IPv4 address assignment (i.e., by means of DHCP).

```
...
+--rw ip-connection
|  +--rw l3-termination-point?     string
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  +--rw local-address?             inet:ipv4-address
|  |  +--rw prefix-length?             uint8
|  |  +--rw address-allocation-type?   identityref
|  |  +--rw (allocation-type)?
|  |     +--:(provider-dhcp)
|  |     |  +--rw dhcp-service-type?   enumeration
|  |     |  +--rw (service-type)?
|  |     |     +--:(relay)
|  |     |     |  +--rw server-ip-address*
|  |     |     |           inet:ipv4-address
|  |     |     +--:(server)
|  |     |        +--rw (address-assign)?
|  |     |           +--:(number)
|  |     |           |  +--rw number-of-dynamic-address?
|  |     |           |           uint16
|  |     |           +--:(explicit)
|  |     |              +--rw customer-addresses
|  |     |                 +--rw address-pool* [pool-id]
|  |     |                    +--rw pool-id          string
|  |     |                    +--rw start-address
|  |     |                    |           inet:ipv4-address
|  |     |                    +--rw end-address?
|  |     |                                inet:ipv4-address
|  |     +--:(dhcp-relay)
|  |     |  +--rw customer-dhcp-servers
|  |     |     +--rw server-ip-address*   inet:ipv4-address
|  |     +--:(static-addresses)
|  |        ...
...
```

Figure 11: IP Connection Subtree Structure (IPv4)

Figure 12 shows the structure of the dynamic IPv6 address assignment (i.e., DHCPv6 and/or SLAAC). Note that if 'address-allocation-type' is set to 'slaac', the Prefix Information option of Router Advertisements that will be issued for SLAAC purposes will carry the IPv6 prefix that is determined by 'local-address' and 'prefix-length'. For example, if 'local-address' is set to '2001:db8:0:1::1' and 'prefix-length' is set to '64', the IPv6 prefix that will be used is '2001:db8:0:1::/64'.

```
...
+--rw ip-connection
|  +--rw l3-termination-point?     string
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  ...
```

```
|  +--rw ipv6 {vpn-common:ipv6}?
|     +--rw local-address?                 inet:ipv6-address
|     +--rw prefix-length?                 uint8
|     +--rw address-allocation-type?       identityref
|     +--rw (allocation-type)?
|        +--:(provider-dhcp)
|        |  +--rw provider-dhcp
|        |     +--rw dhcp-service-type?
|        |     |       enumeration
|        |     +--rw (service-type)?
|        |        +--:(relay)
|        |        |  +--rw server-ip-address*
|        |        |          inet:ipv6-address
|        |        +--:(server)
|        |           +--rw (address-assign)?
|        |              +--:(number)
|        |              |  +--rw number-of-dynamic-address?
|        |              |          uint16
|        |              +--:(explicit)
|        |                 +--rw customer-addresses
|        |                    +--rw address-pool*  [pool-id]
|        |                       +--rw pool-id      string
|        |                       +--rw start-address
|        |                       |       inet:ipv6-address
|        |                       +--rw end-address?
|        |                               inet:ipv6-address
|        +--:(dhcp-relay)
|        |  +--rw customer-dhcp-servers
|        |     +--rw server-ip-address*
|        |             inet:ipv6-address
|        +--:(static-addresses)
|           ...
```

Figure 12: IP Connection Subtree Structure (IPv6)

In the case of static addressing (Figure 13), the model supports the assignment of several IP addresses in the same 'vpn-network-access'. To identify which of the addresses is the primary address of a connection, the 'primary-address' reference MUST be set with the corresponding 'address-id'.

```
...
+--rw ip-connection
|  +--rw l3-termination-point?     string
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  +--rw address-allocation-type?         identityref
|  |  +--rw (allocation-type)?
|  |     ...
|  |     +--:(static-addresses)
|  |        +--rw primary-address?        -> ../address/address-id
|  |        +--rw address* [address-id]
|  |           +--rw address-id          string
|  |           +--rw customer-address?   inet:ipv4-address
|  +--rw ipv6 {vpn-common:ipv6}?
|     +--rw address-allocation-type?         identityref
|     +--rw (allocation-type)?
|        ...
|        +--:(static-addresses)
|           +--rw primary-address?     -> ../address/address-id
|           +--rw address* [address-id]
|              +--rw address-id          string
|              +--rw customer-address?   inet:ipv6-address
...
```

Figure 13: IP Connection Subtree Structure (Static Mode)

7.6.3. CE-PE Routing Protocols

A VPN service provider can configure one or more routing protocols associated with a particular 'vpn-network-access'. Such routing protocols are enabled between the PE and the CE. Each instance is

uniquely identified to accommodate scenarios where multiple instances of the same routing protocol have to be configured on the same link.

The subtree of the 'routing-protocols' is shown in Figure 14.

```
  ...
  +--rw vpn-network-accesses
     +--rw vpn-network-access* [id]
        ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id   string
        |     +--rw type?               identityref
        |     +--rw routing-profiles* [id]
        |     |  +--rw id      leafref
        |     |  +--rw type?   identityref
        |     +--rw static
        |     |  ...
        |     +--rw bgp
        |     |  ...
        |     +--rw ospf
        |     |  ...
        |     +--rw isis
        |     |  ...
        |     +--rw rip
        |     |  ...
        |     +--rw vrrp
        |        ...
        +--rw security
            ...
```

Figure 14: Routing Subtree Structure

Multiple routing instances can be defined, each uniquely identified by an 'id'. The type of routing instance is indicated in 'type'. The values of these attributes are those defined in [RFC9181] (the 'routing-protocol-type' identity).

Configuring multiple instances of the same routing protocol does not automatically imply that, from a device configuration perspective, there will be parallel instances (e.g., multiple processes) running on the PE-CE link. It is up to each implementation (typically, network orchestration, as shown in Figure 1) to decide on the appropriate configuration as a function of underlying capabilities and service provider operational guidelines. As an example, when multiple BGP peers need to be implemented, multiple instances of BGP must be configured as part of this model. However, from a device configuration point of view, this could be implemented as:

* Multiple BGP processes with a single neighbor running in each process.

* A single BGP process with multiple neighbors running.

* A combination thereof.

Routing configuration does not include low-level policies. Such policies are handled at the device configuration level. Local policies of a service provider (e.g., filtering) are implemented as part of the device configuration; these are not captured in the L3NM, but the model allows local profiles to be associated with routing instances ('routing-profiles'). Note that these routing profiles can be scoped to capture parameters that are globally applied to all L3VPN services within a service provider network, while customized L3VPN parameters are captured by means of the L3NM. The provisioning of an L3VPN service will thus rely upon the instantiation of these global routing profiles and the customized L3NM.

7.6.3.1. Static Routing

The L3NM supports the configuration of one or more IPv4/IPv6 static

routes. Since the same structure is used for both IPv4 and IPv6, using one single container to group both static entries independently of their address family was considered at one time, but that design was abandoned to ease the mapping, using the structure provided in [RFC8299].

The static routing subtree structure is shown in Figure 15.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw static
|     |  +--rw cascaded-lan-prefixes
|     |     +--rw ipv4-lan-prefixes*
|     |     |        [lan next-hop]
|     |     |        {vpn-common:ipv4}?
|     |     |  +--rw lan          inet:ipv4-prefix
|     |     |  +--rw lan-tag?       string
|     |     |  +--rw next-hop       union
|     |     |  +--rw bfd-enable?    boolean
|     |     |  +--rw metric?        uint32
|     |     |  +--rw preference?    uint32
|     |     |  +--rw status
|     |     |     +--rw admin-status
|     |     |     |  +--rw status?          identityref
|     |     |     |  +--rw last-change?   yang:date-and-time
|     |     |     +--ro oper-status
|     |     |        +--ro status?          identityref
|     |     |        +--ro last-change?   yang:date-and-time
|     |     +--rw ipv6-lan-prefixes*
|     |              [lan next-hop]
|     |              {vpn-common:ipv6}?
|     |        +--rw lan          inet:ipv6-prefix
|     |        +--rw lan-tag?       string
|     |        +--rw next-hop       union
|     |        +--rw bfd-enable?    boolean
|     |        +--rw metric?        uint32
|     |        +--rw preference?    uint32
|     |        +--rw status
|     |           +--rw admin-status
|     |           |  +--rw status?          identityref
|     |           |  +--rw last-change?   yang:date-and-time
|     |           +--ro oper-status
|     |              +--ro status?          identityref
|     |              +--ro last-change?   yang:date-and-time
...
```

Figure 15: Static Routing Subtree Structure

As depicted in Figure 15, the following data nodes can be defined for a given IP prefix:

'lan-tag': Indicates a local tag (e.g., "myfavorite-lan") that is used to enforce local policies.

'next-hop': Indicates the next hop to be used for the static route. It can be identified by an IP address, a predefined next-hop type (e.g., 'discard' or 'local-link'), etc.

'bfd-enable': Indicates whether BFD is enabled or disabled for this static route entry.

'metric': Indicates the metric associated with the static route entry. This metric is used when the route is exported into an IGP.

'preference': Indicates the preference associated with the static route entry. This preference is used to select a preferred route among routes to the same destination prefix.

'status': Used to convey the status of a static route entry. This data node can also be used to control the (de)activation of individual static route entries.

7.6.3.2. BGP

The L3NM allows the configuration of a BGP neighbor, including a set of parameters that are pertinent to be tweaked at the network level for service customization purposes. The 'bgp' container does not aim to include every BGP parameter; a comprehensive set of parameters belongs more to the BGP device model.

The BGP routing subtree structure is shown in Figure 16.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw bgp
|     |  +--rw description?              string
|     |  +--rw local-as?                 inet:as-number
|     |  +--rw peer-as                   inet:as-number
|     |  +--rw address-family?           identityref
|     |  +--rw local-address?            union
|     |  +--rw neighbor*                 inet:ip-address
|     |  +--rw multihop?                 uint8
|     |  +--rw as-override?              boolean
|     |  +--rw allow-own-as?             uint8
|     |  +--rw prepend-global-as?        boolean
|     |  +--rw send-default-route?       boolean
|     |  +--rw site-of-origin?           rt-types:route-origin
|     |  +--rw ipv6-site-of-origin?      rt-types:ipv6-route-origin
|     |  +--rw redistribute-connected* [address-family]
|     |  |  +--rw address-family    identityref
|     |  |  +--rw enable?           boolean
|     |  +--rw bgp-max-prefix
|     |  |  +--rw max-prefix?          uint32
|     |  |  +--rw warning-threshold?   decimal64
|     |  |  +--rw violate-action?      enumeration
|     |  |  +--rw restart-timer?       uint32
|     |  +--rw bgp-timers
|     |  |  +--rw keepalive?   uint16
|     |  |  +--rw hold-time?   uint16
|     |  +--rw authentication
|     |  |  +--rw enable?            boolean
|     |  |  +--rw keying-material
|     |  |     +--rw (option)?
|     |  |        +--:(ao)
|     |  |        |  +--rw enable-ao?         boolean
|     |  |        |  +--rw ao-keychain?       key-chain:key-chain-ref
|     |  |        +--:(md5)
|     |  |        |  +--rw md5-keychain?      key-chain:key-chain-ref
|     |  |        +--:(explicit)
|     |  |        |  +--rw key-id?            uint32
|     |  |        |  +--rw key?               string
|     |  |        |  +--rw crypto-algorithm?  identityref
|     |  |        +--:(ipsec)
|     |  |           +--rw sa?             string
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?         identityref
|     |     |  +--rw last-change?    yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?         identityref
|     |        +--ro last-change?    yang:date-and-time
...
```

Figure 16: BGP Routing Subtree Structure

The following data nodes are captured in Figure 16. It is up to the implementation (e.g., network orchestrator) to derive the

corresponding BGP device configuration:

'description': Includes a description of the BGP session.

'local-as': Indicates a local AS Number (ASN), if a distinct ASN is required other than the ASN configured at the VPN node level.

'peer-as': Conveys the customer's ASN.

'address-family': Indicates the address family of the peer. It can be set to 'ipv4', 'ipv6', or 'dual-stack'.

This address family will be used together with the 'vpn-type' to derive the appropriate Address Family Identifiers (AFIs) / Subsequent Address Family Identifiers (SAFIs) that will be part of the derived device configurations (e.g., unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128) as defined in Section 4.3.4 of [RFC4364]).

'local-address': Specifies an address or a reference to an interface to use when establishing the BGP transport session.

'neighbor': Can indicate two neighbors (each for a given address family) or one neighbor (if the 'address-family' attribute is set to 'dual-stack'). A list of IP address(es) of the BGP neighbor(s) can then be conveyed in this data node.

'multihop': Indicates the number of allowed IP hops between a PE and its BGP peer.

'as-override': If set, this parameter indicates whether ASN override is enabled, i.e., replacing the ASN of the customer specified in the AS_PATH BGP attribute with the ASN identified in the 'local-as' attribute.

'allow-own-as': Used in some topologies (e.g., hub-and-spoke) to allow the provider's ASN to be included in the AS_PATH BGP attribute received from a CE. Loops are prevented by setting 'allow-own-as' to a maximum number of the provider's ASN occurrences. By default, this parameter is set to '0' (that is, reject any AS_PATH attribute that includes the provider's ASN).

'prepend-global-as': When distinct ASNs are configured at the VPN node and network access levels, this parameter controls whether the ASN provided at the VPN node level is prepended to the AS_PATH attribute.

'send-default-route': Controls whether default routes can be advertised to the peer.

'site-of-origin': Meant to uniquely identify the set of routes learned from a site via a particular CE-PE connection. It is used to prevent routing loops (Section 7 of [RFC4364]). The Site of Origin attribute is encoded as a Route Origin Extended Community.

'ipv6-site-of-origin': Carries an IPv6 Address Specific BGP Extended Community that is used to indicate the Site of Origin for VRF information [RFC5701]. It is used to prevent routing loops.

'redistribute-connected': Controls whether the PE-CE link is advertised to other PEs.

'bgp-max-prefix': Controls the behavior when a prefix maximum is reached.

'max-prefix': Indicates the maximum number of BGP prefixes allowed in the BGP session. If the limit is reached, the action indicated in 'violate-action' will be followed.

'warning-threshold': A warning notification is triggered when this limit is reached.

'violate-action': Indicates which action to execute when the maximum number of BGP prefixes is reached. Examples of such actions include sending a warning message, discarding extra paths from the peer, or restarting the session.

'restart-timer': Indicates, in seconds, the time interval after which the BGP session will be reestablished.

'bgp-timers': Two timers can be captured in this container: (1) 'hold-time', which is the time interval that will be used for the Hold Timer (Section 4.2 of [RFC4271]) when establishing a BGP session and (2) 'keepalive', which is the time interval for the KeepaliveTimer between a PE and a BGP peer (Section 4.4 of [RFC4271]). Both timers are expressed in seconds.

'authentication': The module adheres to the recommendations in Section 13.2 of [RFC4364], as it allows enabling the TCP Authentication Option (TCP-AO) [RFC5925] and accommodates the installed base that makes use of MD5. In addition, the module includes a provision for using IPsec.

This version of the L3NM assumes that parameters specific to the TCP-AO are preconfigured as part of the key chain that is referenced in the L3NM. No assumption is made about how such a key chain is preconfigured. However, the structure of the key chain should cover data nodes beyond those in [RFC8177], mainly SendID and RecvID (Section 3.1 of [RFC5925]).

'status': Indicates the status of the BGP routing instance.

7.6.3.3. OSPF

OSPF can be configured to run as a routing protocol on the 'vpn-network-access'.

The OSPF routing subtree structure is shown in Figure 17.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw ospf
|        +--rw address-family?   identityref
|        +--rw area-id           yang:dotted-quad
|        +--rw metric?           uint16
|        +--rw sham-links  {vpn-common:rtg-ospf-sham-link}?
|        |  +--rw sham-link* [target-site]
|        |     +--rw target-site    string
|        |     +--rw metric?        uint16
|        +--rw max-lsa?          uint32
|        +--rw authentication
|        |  +--rw enable?            boolean
|        |  +--rw keying-material
|        |     +--rw (option)?
|        |        +--:(auth-key-chain)
|        |        |  +--rw key-chain?
|        |        |          key-chain:key-chain-ref
|        |        +--:(auth-key-explicit)
|        |        |  +--rw key-id?      uint32
|        |        |  +--rw key?         string
|        |        |  +--rw crypto-algorithm?
|        |        |          identityref
|        |        +--:(ipsec)
|        |           +--rw sa?    string
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--rw last-change?  yang:date-and-time
|           +--ro oper-status
|              +--ro status?        identityref
|              +--ro last-change?  yang:date-and-time
```

```
...
```

Figure 17: OSPF Routing Subtree Structure

The following data nodes are captured in Figure 17:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.

When the IPv4 or dual-stack address family is requested, it is up to the implementation (e.g., network orchestrator) to decide whether OSPFv2 [RFC4577] or OSPFv3 [RFC6565] is used to announce IPv4 routes. Such a decision will typically be reflected in the device configurations that will be derived to implement the L3VPN.

'area-id': Indicates the OSPF Area ID.

'metric': Associates a metric with OSPF routes.

'sham-links': Used to create OSPF sham links between two VPN network accesses sharing the same area and having a backdoor link (Section 4.2.7 of [RFC4577] and Section 5 of [RFC6565]).

'max-lsa': Sets the maximum number of Link State Advertisements (LSAs) that the OSPF instance will accept.

'authentication': Controls the authentication schemes to be enabled for the OSPF instance. The following options are supported: IPsec for OSPFv3 authentication [RFC4552], and the Authentication Trailer for OSPFv2 [RFC5709] [RFC7474] and OSPFv3 [RFC7166].

'status': Indicates the status of the OSPF routing instance.

7.6.3.4. IS-IS

The model allows the user to configure IS-IS [ISO10589] [RFC1195] [RFC5308] to run on the 'vpn-network-access' interface. See Figure 18.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw isis
|        |  +--rw address-family?   identityref
|        |  +--rw area-address      area-address
|        |  +--rw level?            identityref
|        |  +--rw metric?           uint16
|        |  +--rw mode?             enumeration
|        |  +--rw authentication
|        |  |  +--rw enable?            boolean
|        |  |  +--rw keying-material
|        |  |     +--rw (option)?
|        |  |        +--:(auth-key-chain)
|        |  |        |  +--rw key-chain?
|        |  |        |          key-chain:key-chain-ref
|        |  |        +--:(auth-key-explicit)
|        |  |           +--rw key-id?             uint32
|        |  |           +--rw key?                string
|        |  |           +--rw crypto-algorithm?   identityref
|        |  +--rw status
|        |     +--rw admin-status
|        |     |  +--rw status?        identityref
|        |     |  +--rw last-change?  yang:date-and-time
|        |     +--ro oper-status
|        |        +--ro status?        identityref
|        |        +--ro last-change?  yang:date-and-time
...
```

Figure 18: IS-IS Routing Subtree Structure

The following IS-IS data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.

'area-address': Indicates the IS-IS area address.

'level': Indicates the IS-IS level: Level 1, Level 2, or both.

'metric': Associates a metric with IS-IS routes.

'mode': Indicates the IS-IS interface mode type. It can be set to 'active' (that is, send or receive IS-IS protocol control packets) or 'passive' (that is, suppress the sending of IS-IS updates through the interface).

'authentication': Controls the authentication schemes to be enabled for the IS-IS instance. Both the specification of a key chain [RFC8177] and the direct specification of key and authentication algorithms are supported.

'status': Indicates the status of the IS-IS routing instance.

7.6.3.5. RIP

The model allows the user to configure RIP to run on the 'vpn-network-access' interface. See Figure 19.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw rip
|        +--rw address-family?   identityref
|        +--rw timers
|        |  +--rw update-interval?     uint16
|        |  +--rw invalid-interval?    uint16
|        |  +--rw holddown-interval?   uint16
|        |  +--rw flush-interval?      uint16
|        +--rw default-metric?   uint8
|        +--rw authentication
|        |  +--rw enable?            boolean
|        |  +--rw keying-material
|        |     +--rw (option)?
|        |        +--:(auth-key-chain)
|        |        |  +--rw key-chain?
|        |        |           key-chain:key-chain-ref
|        |        +--:(auth-key-explicit)
|        |           +--rw key?                string
|        |           +--rw crypto-algorithm?   identityref
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--rw last-change?  yang:date-and-time
|           +--ro oper-status
|              +--ro status?        identityref
|              +--ro last-change?  yang:date-and-time
...
```

Figure 19: RIP Subtree Structure

As shown in Figure 19, the following RIP data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. This parameter is used to determine whether RIPv2 [RFC2453], RIP Next Generation (RIPng), or both are to be enabled [RFC2080].

'timers': Indicates the following timers:

'update-interval': The interval at which RIP updates are sent.

'invalid-interval': The interval before a RIP route is declared invalid.

'holddown-interval': The interval before better RIP routes are released.

'flush-interval': The interval before a route is removed from the routing table.

These timers are expressed in seconds.

'default-metric': Sets the default RIP metric.

'authentication': Controls the authentication schemes to be enabled for the RIP instance.

'status': Indicates the status of the RIP routing instance.

7.6.3.6. VRRP

The model allows enabling the Virtual Router Redundancy Protocol (VRRP) on the 'vpn-network-access' interface. See Figure 20.

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw vrrp
|        +--rw address-family*   identityref
|        +--rw vrrp-group?       uint8
|        +--rw backup-peer?      inet:ip-address
|        +--rw virtual-ip-address*   inet:ip-address
|        +--rw priority?         uint8
|        +--rw ping-reply?       boolean
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--rw last-change?  yang:date-and-time
|           +--ro oper-status
|              +--ro status?        identityref
|              +--ro last-change?  yang:date-and-time
...
```

Figure 20: VRRP Subtree Structure

The following data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. Note that VRRP version 3 [RFC5798] supports both IPv4 and IPv6.

'vrrp-group': Used to identify the VRRP group.

'backup-peer': Carries the IP address of the peer.

'virtual-ip-address': Includes virtual IP addresses for a single VRRP group.

'priority': Assigns the VRRP election priority for the backup virtual router.

'ping-reply': Controls whether the VRRP speaker should reply to ping requests.

'status': Indicates the status of the VRRP instance.

Note that no authentication data node is included for VRRP, as there isn't any type of VRRP authentication at this time (see Section 9 of [RFC5798]).

7.6.4. OAM

This container (Figure 21) defines the Operations, Administration, and Maintenance (OAM) mechanisms used for a VPN network access. In the current version of the L3NM, only BFD is supported.

```
...
+--rw oam
|  +--rw bfd {vpn-common:bfd}?
|     +--rw session-type?               identityref
|     +--rw desired-min-tx-interval?    uint32
|     +--rw required-min-rx-interval?   uint32
|     +--rw local-multiplier?           uint8
|     +--rw holdtime?                   uint32
|     +--rw profile?                    leafref
|     +--rw authentication!
|     |  +--rw key-chain?    key-chain:key-chain-ref
|     |  +--rw meticulous?   boolean
|     +--rw status
|        +--rw admin-status
|           |  +--rw status?         identityref
|           |  +--rw last-change?   yang:date-and-time
|           +--ro oper-status
|              +--ro status?         identityref
|              +--ro last-change?   yang:date-and-time
...
```

Figure 21: IP Connection Subtree Structure (OAM)

The following OAM data nodes can be specified:

'session-type': Indicates which BFD flavor is used to set up the session (e.g., classic BFD [RFC5880], Seamless BFD [RFC7880]). By default, it is assumed that the BFD session will follow the behavior specified in [RFC5880].

'desired-min-tx-interval': The minimum interval, in microseconds, that a PE would like to use when transmitting BFD Control packets, less any jitter applied.

'required-min-rx-interval': The minimum interval, in microseconds, between received BFD Control packets that a PE is capable of supporting, less any jitter applied by the sender.

'local-multiplier': The negotiated transmit interval, multiplied by this value, provides the detection time for the peer.

'holdtime': Used to indicate the expected BFD holddown time, in milliseconds. This value may be inherited from the service request (see Section 6.3.2.2.2 of [RFC8299]).

'profile': Refers to a BFD profile (Section 7.2). Such a profile can be set by the provider or inherited from the service request (see Section 6.3.2.2.2 of [RFC8299]).

'authentication': Includes the required information to enable the BFD authentication modes discussed in Section 6.7 of [RFC5880]. In particular, 'meticulous' controls the activation of meticulous mode as discussed in Sections 6.7.3 and 6.7.4 of [RFC5880].

'status': Indicates the status of BFD.

7.6.5. Security

The 'security' container specifies the authentication and the encryption to be applied to traffic for a given VPN network access. As depicted in the subtree shown in Figure 22, the L3NM can be used to directly control the encryption to be applied (e.g., Layer 2 or Layer 3 encryption) or invoke a local encryption profile.

```
     ...
```

```
      +--rw vpn-services
         +--rw vpn-service* [vpn-id]
            ...
            +--rw vpn-nodes
               +--rw vpn-node* [vpn-node-id]
                  ...
                  +--rw vpn-network-accesses
                     +--rw vpn-network-access* [id]
                        ...
                        +--rw security
                        |  +--rw encryption {vpn-common:encryption}?
                        |  |  +--rw enabled?   boolean
                        |  |  +--rw layer?     enumeration
                        |  +--rw encryption-profile
                        |     +--rw (profile)?
                        |        +--:(provider-profile)
                        |        |  +--rw profile-name?        leafref
                        |        +--:(customer-profile)
                        |           +--rw customer-key-chain?
                        |                    key-chain:key-chain-ref
                        +--rw service
                            ...
```

Figure 22: Security Subtree Structure

7.6.6. Services

7.6.6.1. Overview

The 'service' container specifies the service parameters to apply for a given VPN network access (Figure 23).

```
...
+--rw vpn-network-accesses
   +--rw vpn-network-access* [id]
      ...
      +--rw service
         +--rw pe-to-ce-bandwidth?   uint64 {vpn-common:inbound-bw}?
         +--rw ce-to-pe-bandwidth?   uint64 {vpn-common:outbound-bw}?
         +--rw mtu?                  uint32
         +--rw qos {vpn-common:qos}?
         |  ...
         +--rw carriers-carrier
         |       {vpn-common:carriers-carrier}?
         |  +--rw signaling-type?   enumeration
         +--rw ntp
         |  +--rw broadcast?        enumeration
         |  +--rw auth-profile
         |  |  +--rw profile-id?    string
         |  +--rw status
         |     +--rw admin-status
         |     |  +--rw status?         identityref
         |     |  +--rw last-change?   yang:date-and-time
         |     +--ro oper-status
         |        +--ro status?         identityref
         |        +--ro last-change?   yang:date-and-time
         +--rw multicast {vpn-common:multicast}?
            ...
```

Figure 23: Services Subtree Structure

The following data nodes are defined:

'pe-to-ce-bandwidth': Indicates, in bits per second (bps), the inbound bandwidth of the connection (i.e., the download bandwidth from the service provider to the site).

'ce-to-pe-bandwidth': Indicates, in bps, the outbound bandwidth of the connection (i.e., the upload bandwidth from the site to the service provider).

'mtu': Indicates the MTU at the service level.

'qos': Used to define a set of QoS policies to apply on a given connection (refer to Section 7.6.6.2 for more details).

'carriers-carrier': Groups a set of parameters that are used when Carriers' Carriers (CsC) is enabled, such as using BGP for signaling purposes [RFC8277].

'ntp': Time synchronization may be needed in some VPNs, such as infrastructure and management VPNs. This container is used to enable the NTP service [RFC5905].

'multicast': Specifies the multicast mode and other data nodes, such as the address family. Refer to Section 7.7.

7.6.6.2. QoS

The 'qos' container is used to define a set of QoS policies to apply on a given connection (Figure 24). A QoS policy may be a classification or an action policy. For example, a QoS action can be defined to rate-limit inbound/outbound traffic of a given class of service.

```
...
+--rw qos {vpn-common:qos}?
|  +--rw qos-classification-policy
|  |  +--rw rule* [id]
|  |     +--rw id              string
|  |     +--rw (match-type)?
|  |     |  +--:(match-flow)
|  |     |  |  +--rw (l3)?
|  |     |  |  |  +--:(ipv4)
|  |     |  |  |  |  ...
|  |     |  |  |  +--:(ipv6)
|  |     |  |  |     ...
|  |     |  |  +--rw (l4)?
|  |     |  |     +--:(tcp)
|  |     |  |     |  ...
|  |     |  |     +--:(udp)
|  |     |  |        ...
|  |     |  +--:(match-application)
|  |     |     +--rw match-application?
|  |     |              identityref
|  |     +--rw target-class-id?    string
|  +--rw qos-action
|  |  +--rw rule* [id]
|  |     +--rw id                      string
|  |     +--rw target-class-id?        string
|  |     +--rw inbound-rate-limit?     decimal64
|  |     +--rw outbound-rate-limit?    decimal64
|  +--rw qos-profile
|     +--rw qos-profile* [profile]
|        +--rw profile      leafref
|        +--rw direction?   identityref
...
```

Figure 24: Overall QoS Subtree Structure

QoS classification can be based on many criteria, such as the following:

Layer 3: As shown in Figure 25, classification can be based on any IP header field or a combination thereof. Both IPv4 and IPv6 are supported.

```
+--rw qos {vpn-common:qos}?
|  +--rw qos-classification-policy
|  |  +--rw rule* [id]
|  |     +--rw id           string
|  |     +--rw (match-type)?
```

```
|  |  |  +--:(match-flow)
|  |  |  |  +--rw (l3)?
|  |  |  |  |  +--:(ipv4)
|  |  |  |  |  |  +--rw ipv4
|  |  |  |  |  |     +--rw dscp?              inet:dscp
|  |  |  |  |  |     +--rw ecn?               uint8
|  |  |  |  |  |     +--rw length?            uint16
|  |  |  |  |  |     +--rw ttl?               uint8
|  |  |  |  |  |     +--rw protocol?          uint8
|  |  |  |  |  |     +--rw ihl?               uint8
|  |  |  |  |  |     +--rw flags?             bits
|  |  |  |  |  |     +--rw offset?            uint16
|  |  |  |  |  |     +--rw identification?    uint16
|  |  |  |  |  |     +--rw (destination-network)?
|  |  |  |  |  |     |  +--:(destination-ipv4-network)
|  |  |  |  |  |     |     +--rw destination-ipv4-network?
|  |  |  |  |  |     |             inet:ipv4-prefix
|  |  |  |  |  |     +--rw (source-network)?
|  |  |  |  |  |        +--:(source-ipv4-network)
|  |  |  |  |  |           +--rw source-ipv4-network?
|  |  |  |  |  |  inet:ipv4-prefix
|  |  |  |  |  +--:(ipv6)
|  |  |  |  |     +--rw ipv6
|  |  |  |  |        +--rw dscp?              inet:dscp
|  |  |  |  |        +--rw ecn?               uint8
|  |  |  |  |        +--rw length?            uint16
|  |  |  |  |        +--rw ttl?               uint8
|  |  |  |  |        +--rw protocol?          uint8
|  |  |  |  |        +--rw (destination-network)?
|  |  |  |  |        |  +--:(destination-ipv6-network)
|  |  |  |  |        |     +--rw destination-ipv6-network?
|  |  |  |  |        |             inet:ipv6-prefix
|  |  |  |  |        +--rw (source-network)?
|  |  |  |  |        |  +--:(source-ipv6-network)
|  |  |  |  |        |     +--rw source-ipv6-network?
|  |  |  |  |        |             inet:ipv6-prefix
|  |  |  |  |        +--rw flow-label?
|  |  |  |  |                   inet:ipv6-flow-label
...
```

Figure 25: QoS Subtree Structure (L3)

Layer 4: As discussed in [RFC9181], any Layer 4 protocol can be indicated in the 'protocol' data node under 'l3' (Figure 25), but only TCP- and UDP-specific match criteria are elaborated in this version, as these protocols are widely used in the context of VPN services. Augmentations can be considered in the future to add other Layer-4-specific data nodes, if needed.

TCP- or UDP-related match criteria can be specified in the L3NM, as shown in Figure 26.

As discussed in [RFC9181], some transport protocols use existing protocols (e.g., TCP or UDP) as the substrate. The match criteria for such protocols may rely upon the 'protocol' setting under 'l3', TCP/UDP match criteria as shown in Figure 26, part of the TCP/UDP payload, or a combination thereof. This version of the module does not support such advanced match criteria. Future revisions of the VPN common module or augmentations to the L3NM may consider adding match criteria based on the transport protocol payload (e.g., by means of a bitmask match).

```
+--rw qos {vpn-common:qos}?
|  +--rw qos-classification-policy
|  |  +--rw rule* [id]
|  |     +--rw id           string
|  |     +--rw (match-type)?
|  |     |  +--:(match-flow)
|  |     |  |  +--rw (l3)?
|  |     |  |  |  ...
|  |     |  |  +--rw (l4)?
```

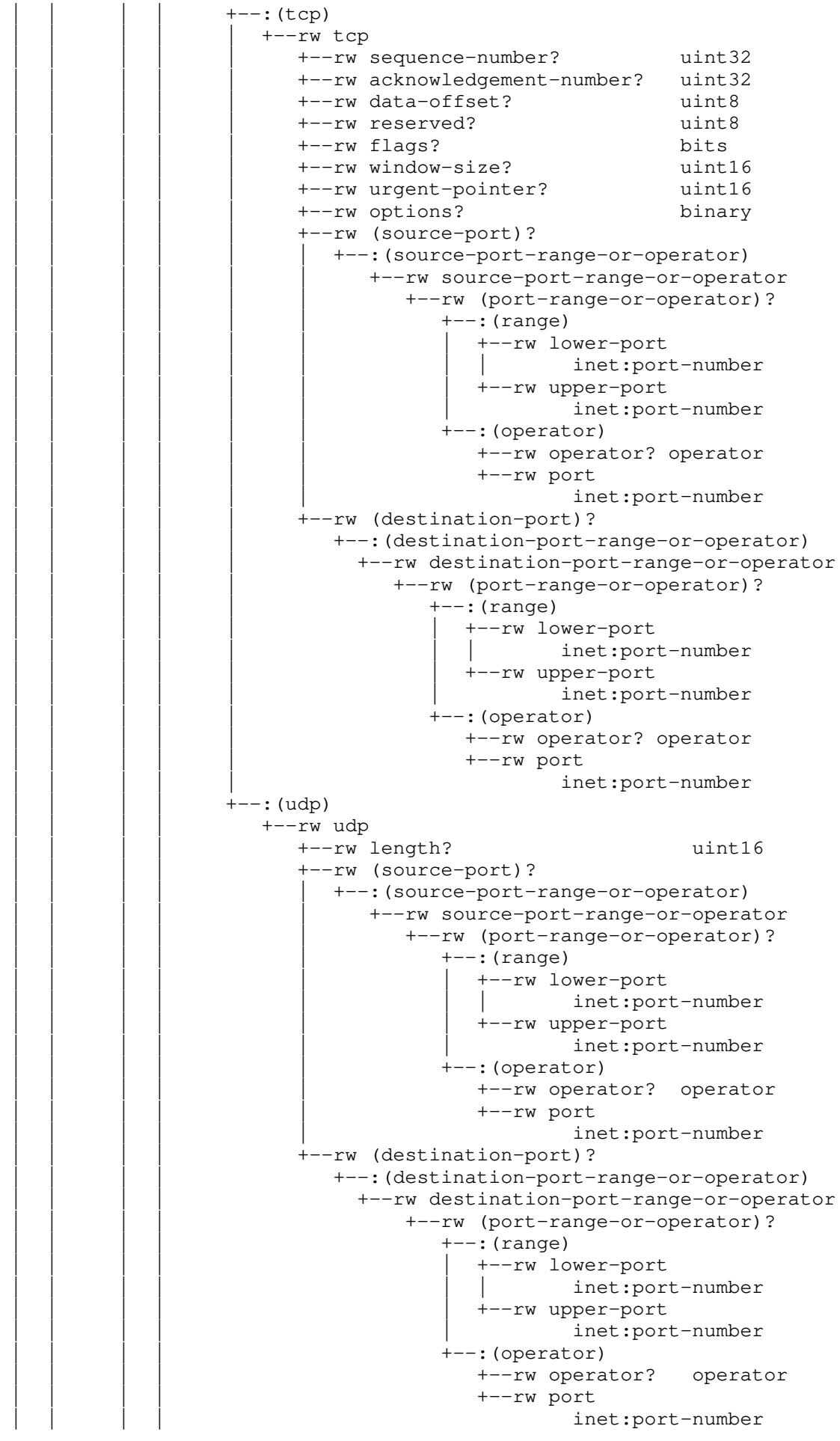


Figure 26: QoS Subtree Structure (L4)

Application match: Relies upon application-specific classification (Figure 24).

7.7. Multicast

Multicast may be enabled for a particular VPN at the VPN node and VPN network access levels (see Figure 27). Some data nodes (e.g., max-groups (Figure 28)) can be controlled at various levels: VPN service, VPN node level, or VPN network access.

```
     ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           ...
           +--rw vpn-instance-profiles
           |  +--rw vpn-instance-profile* [profile-id]
           |     ....
           |     +--rw multicast {vpn-common:multicast}?
           |        ...
           +--rw vpn-nodes
              +--rw vpn-node* [vpn-node-id]
                 ...
                 +--rw active-vpn-instance-profiles
                 |  +--rw vpn-instance-profile* [profile-id]
                 |     ...
                 |     +--rw multicast {vpn-common:multicast}?
                 |        ...
                 +--rw vpn-network-accesses
                    +--rw vpn-network-access* [id]
                       ...
                       +--rw service
                          ...
                          +--rw multicast {vpn-common:multicast}?
                             ...
```

Figure 27: Overall Multicast Subtree Structure

Multicast-related data nodes at the VPN instance profile level have the structure shown in Figure 28.

```
...
+--rw vpn-services
   +--rw vpn-service* [vpn-id]
      ...
      +--rw vpn-instance-profiles
      |  +--rw vpn-instance-profile* [profile-id]
      |     ....
      |     +--rw multicast {vpn-common:multicast}?
      |        +--rw tree-flavor?   identityref
      |        +--rw rp
      |           |  +--rw rp-group-mappings
      |           |  |  +--rw rp-group-mapping* [id]
      |           |  |     +--rw id                    uint16
      |           |  |     +--rw provider-managed
      |           |  |     |  +--rw enabled?                    boolean
      |           |  |     |  +--rw rp-redundancy?              boolean
      |           |  |     |  +--rw optimal-traffic-delivery?   boolean
      |           |  |     |  +--rw anycast
      |           |  |     |     +--rw local-address?    inet:ip-address
      |           |  |     |     +--rw rp-set-address*   inet:ip-address
      |           |  |     +--rw rp-address          inet:ip-address
      |           |  |     +--rw groups
      |           |  |        +--rw group* [id]
      |           |  |           +--rw id                      uint16
      |           |  |           +--rw (group-format)
      |           |  |              +--:(group-prefix)
      |           |  |              |  +--rw group-address?
      |           |  |              |           inet:ip-prefix
      |           |  |              +--:(startend)
      |           |  |                 +--rw group-start?
      |           |  |                 |       inet:ip-address
      |           |  |                 +--rw group-end?
      |           |  |                 |       inet:ip-address
```

```
|          |  +--rw rp-discovery
|          |     +--rw rp-discovery-type?   identityref
|          |     +--rw bsr-candidates
|          |        +--rw bsr-candidate-address*
|          |        |       inet:ip-address
|          +--rw igmp {vpn-common:igmp and vpn-common:ipv4}?
|          |  +--rw static-group* [group-addr]
|          |  |  +--rw group-addr
|          |  |  |       rt-types:ipv4-multicast-group-address
|          |  |  +--rw source-addr?
|          |  |          rt-types:ipv4-multicast-source-address
|          |  +--rw max-groups?     uint32
|          |  +--rw max-entries?    uint32
|          |  +--rw version?        identityref
|          +--rw mld {vpn-common:mld and vpn-common:ipv6}?
|          |  +--rw static-group* [group-addr]
|          |  |  +--rw group-addr
|          |  |  |       rt-types:ipv6-multicast-group-address
|          |  |  +--rw source-addr?
|          |  |          rt-types:ipv6-multicast-source-address
|          |  +--rw max-groups?     uint32
|          |  +--rw max-entries?    uint32
|          |  +--rw version?        identityref
|          +--rw pim {vpn-common:pim}?
|             +--rw hello-interval?
|             |       rt-types:timer-value-seconds16
|             +--rw dr-priority?      uint32
|        ...
```

Figure 28: Multicast Subtree Structure (VPN Instance Profile Level)

The model supports a single type of tree per VPN access ('tree-flavor'): Any-Source Multicast (ASM), Source-Specific Multicast (SSM), or bidirectional.

When ASM is used, the model supports the configuration of Rendezvous Points (RPs). RP discovery may be 'static', 'bsr-rp', or 'auto-rp'. When set to 'static', RP-to-multicast-group mappings MUST be configured as part of the 'rp-group-mappings' container. The RP MAY be a provider node or a customer node. When the RP is a customer node, the RP address must be configured using the 'rp-address' leaf.

The model supports RP redundancy through the 'rp-redundancy' leaf. How the redundancy is achieved is out of scope.

When a particular VPN using ASM requires traffic delivery that is more optimal (e.g., requested per the guidance in [RFC8299]), 'optimal-traffic-delivery' can be set. When set to 'true', the implementation must use any mechanism to provide traffic delivery that is more optimal for the customer. For example, anycast is one of the mechanisms for enhancing RP redundancy, providing resilience against failures, and recovering from failures quickly.

When configuring multicast-related parameters at the VPN node level (Figure 29), the same structure as the structure depicted in Figure 30 is used. When defined at the VPN node level, Internet Group Management Protocol (IGMP) parameters [RFC1112] [RFC2236] [RFC3376], Multicast Listener Discovery (MLD) parameters [RFC2710] [RFC3810], and Protocol Independent Multicast (PIM) parameters [RFC7761] are applicable to all VPN network accesses of that VPN node unless corresponding nodes are overridden at the VPN network access level.

```
...
+--rw vpn-nodes
   +--rw vpn-node* [vpn-node-id]
      ...
      +--rw active-vpn-instance-profiles
      |  +--rw vpn-instance-profile* [profile-id]
      |     ...
      |     +--rw multicast {vpn-common:multicast}?
```

```
|        +--rw tree-flavor*   identityref
|        +--rw rp
|        |  ...
|        +--rw igmp {vpn-common:igmp and vpn-common:ipv4}?
|        |  ...
|        +--rw mld {vpn-common:mld and vpn-common:ipv6}?
|        |  ...
|        +--rw pim {vpn-common:pim}?
|           ...
```

Figure 29: Multicast Subtree Structure (VPN Node Level)

Multicast-related data nodes at the VPN network access level are shown in Figure 30. The values configured at the VPN network access level override the values configured for the corresponding data nodes at other levels.

```
...
+--rw vpn-network-accesses
   +--rw vpn-network-access* [id]
      ...
      +--rw service
         ...
         +--rw multicast {vpn-common:multicast}?
            +--rw access-type?      enumeration
            +--rw address-family?   identityref
            +--rw protocol-type?    enumeration
            +--rw remote-source?    boolean
            +--rw igmp  {vpn-common:igmp}?
            |  +--rw static-group* [group-addr]
            |  |  +--rw group-addr
            |  |           rt-types:ipv4-multicast-group-address
            |  |  +--rw source-addr?
            |  |           rt-types:ipv4-multicast-source-address
            |  +--rw max-groups?          uint32
            |  +--rw max-entries?         uint32
            |  +--rw max-group-sources?   uint32
            |  +--rw version?             identityref
            |  +--rw status
            |     +--rw admin-status
            |     |  +--rw status?         identityref
            |     |  +--rw last-change?   yang:date-and-time
            |     +--ro oper-status
            |        +--ro status?         identityref
            |        +--ro last-change?   yang:date-and-time
            +--rw mld {vpn-common:mld}?
            |  +--rw static-group* [group-addr]
            |  |  +--rw group-addr
            |  |           rt-types:ipv6-multicast-group-address
            |  |  +--rw source-addr?
            |  |           rt-types:ipv6-multicast-source-address
            |  +--rw max-groups?          uint32
            |  +--rw max-entries?         uint32
            |  +--rw max-group-sources?   uint32
            |  +--rw version?             identityref
            |  +--rw status
            |     +--rw admin-status
            |     |  +--rw status?         identityref
            |     |  +--rw last-change?   yang:date-and-time
            |     +--ro oper-status
            |        +--ro status?         identityref
            |        +--ro last-change?   yang:date-and-time
            +--rw pim {vpn-common:pim}?
               +--rw hello-interval?   rt-types:timer-value-seconds16
               +--rw dr-priority?      uint32
               +--rw status
                  +--rw admin-status
                  |  +--rw status?         identityref
                  |  +--rw last-change?   yang:date-and-time
                  +--ro oper-status
                     +--ro status?         identityref
```

```
                      +--ro last-change?   yang:date-and-time
```

Figure 30: Multicast Subtree Structure (VPN Network Access Level)

8. L3NM YANG Module

This module uses types defined in [RFC6991], [RFC8343], and [RFC9181]. It also uses groupings defined in [RFC8519], [RFC8177], and [RFC8294].

```
<CODE BEGINS> file "ietf-l3vpn-ntw@2022-02-14.yang"
module ietf-l3vpn-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw";
  prefix l3nm;

  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types, Section 4";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types, Section 3";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>

     Author:   Samier Barguil
               <mailto:samier.barguilgiraldo.ext@telefonica.com>
     Editor:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Luis Angel Munoz
               <mailto:luis-angel.munoz@vodafone.com>
     Author:   Alejandro Aguado
               <mailto:alejandro.aguado_martin@nokia.com>";
  description
    "This YANG module defines a generic network-oriented model
     for the configuration of Layer 3 Virtual Private Networks.

     Copyright (c) 2022 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
```

```
   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Revised BSD License
   set forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC 9182; see the
   RFC itself for full legal notices.";

revision 2022-02-14 {
  description
    "Initial revision.";
  reference
    "RFC 9182: A YANG Network Data Model for Layer 3 VPNs";
}

/* Features */

feature msdp {
  description
    "This feature indicates that Multicast Source Discovery
     Protocol (MSDP) capabilities are supported by the VPN.";
  reference
    "RFC 3618: Multicast Source Discovery Protocol (MSDP)";
}

/* Identities */

identity address-allocation-type {
  description
    "Base identity for address allocation type in the
     Provider Edge to Customer Edge (PE-CE) link.";
}

identity provider-dhcp {
  base address-allocation-type;
  description
    "The provider's network provides a DHCP service to the
     customer.";
}

identity provider-dhcp-relay {
  base address-allocation-type;
  description
    "The provider's network provides a DHCP relay service to the
     customer.";
}

identity provider-dhcp-slaac {
  if-feature "vpn-common:ipv6";
  base address-allocation-type;
  description
    "The provider's network provides a DHCP service to the
     customer as well as IPv6 Stateless Address
     Autoconfiguration (SLAAC).";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

identity static-address {
  base address-allocation-type;
  description
    "The provider's network provides static IP addressing to the
     customer.";
}

identity slaac {
  if-feature "vpn-common:ipv6";
  base address-allocation-type;
  description
```

```
    "The provider's network uses IPv6 SLAAC to provide
     addressing to the customer.";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

identity local-defined-next-hop {
  description
    "Base identity of local defined next hops.";
}

identity discard {
  base local-defined-next-hop;
  description
    "Indicates an action to discard traffic for the
     corresponding destination.
     For example, this can be used to black-hole traffic.";
}

identity local-link {
  base local-defined-next-hop;
  description
    "Treat traffic towards addresses within the specified
     next-hop prefix as though they are connected to a local
     link.";
}

identity l2-tunnel-type {
  description
    "Base identity for Layer 2 tunnel selection under the VPN
     network access.";
}

identity pseudowire {
  base l2-tunnel-type;
  description
    "Pseudowire tunnel termination in the VPN network access.";
}

identity vpls {
  base l2-tunnel-type;
  description
    "Virtual Private LAN Service (VPLS) tunnel termination in
     the VPN network access.";
}

identity vxlan {
  base l2-tunnel-type;
  description
    "Virtual eXtensible Local Area Network (VXLAN) tunnel
     termination in the VPN network access.";
}

/* Typedefs */

typedef predefined-next-hop {
  type identityref {
    base local-defined-next-hop;
  }
  description
    "Predefined next-hop designation for locally generated
     routes.";
}

typedef area-address {
  type string {
    pattern '[0-9A-Fa-f]{2}(\.[0-9A-Fa-f]{4}){0,6}';
  }
  description
    "This type defines the area address format.";
}
```

```
/* Groupings */

grouping vpn-instance-profile {
  description
    "Grouping for data nodes that may be factorized
     among many levels of the model.  The grouping can
     be used to define generic profiles at the VPN service
     level and then referenced at the VPN node and VPN
     network access levels.";
  leaf local-as {
    if-feature "vpn-common:rtg-bgp";
    type inet:as-number;
    description
      "Provider's Autonomous System (AS) number.  Used if the
       customer requests BGP routing.";
  }
  uses vpn-common:route-distinguisher;
  list address-family {
    key "address-family";
    description
      "Set of parameters per address family.";
    leaf address-family {
      type identityref {
        base vpn-common:address-family;
      }
      description
        "Indicates the address family (IPv4 and/or IPv6).";
    }
    container vpn-targets {
      description
        "Set of route targets to match for import and export
         routes to/from VRF.";
      uses vpn-common:vpn-route-targets;
    }
    list maximum-routes {
      key "protocol";
      description
        "Defines the maximum number of routes for VRF.";
      leaf protocol {
        type identityref {
          base vpn-common:routing-protocol-type;
        }
        description
          "Indicates the routing protocol.  A value of 'any'
           can be used to identify a limit that will apply for
           each active routing protocol.";
      }
      leaf maximum-routes {
        type uint32;
        description
          "Indicates the maximum number of prefixes that VRF can
           accept for this address family and protocol.";
      }
    }
  }
  container multicast {
    if-feature "vpn-common:multicast";
    description
      "Global multicast parameters.";
    leaf tree-flavor {
      type identityref {
        base vpn-common:multicast-tree-type;
      }
      description
        "Type of the multicast tree to be used.";
    }
    container rp {
      description
        "Rendezvous Point (RP) parameters.";
      container rp-group-mappings {
```

```
description
  "RP-to-group mapping parameters.";
list rp-group-mapping {
  key "id";
  description
    "List of RP-to-group mappings.";
  leaf id {
    type uint16;
    description
      "Unique identifier for the mapping.";
  }
  container provider-managed {
    description
      "Parameters for a provider-managed RP.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "Set to 'true' if the RP must be a
         provider-managed node.  Set to 'false' if it is
         a customer-managed node.";
    }
    leaf rp-redundancy {
      type boolean;
      default "false";
      description
        "If set to 'true', it indicates that a
         redundancy mechanism for the RP is required.";
    }
    leaf optimal-traffic-delivery {
      type boolean;
      default "false";
      description
        "If set to 'true', the service provider (SP)
         must ensure that the traffic uses an optimal
         path.  An SP may use Anycast RP or
         RP-tree-to-SPT ('SPT' is 'shortest path tree')
         switchover architectures.";
    }
    container anycast {
      when "../rp-redundancy = 'true' and
            ../optimal-traffic-delivery = 'true'" {
        description
          "Only applicable if both RP redundancy and
           delivery through an optimal path are
           activated.";
      }
      description
        "PIM Anycast-RP parameters.";
      leaf local-address {
        type inet:ip-address;
        description
          "IP local address for the PIM RP.  Usually
           corresponds to the Router ID or the
           primary address.";
      }
      leaf-list rp-set-address {
        type inet:ip-address;
        description
          "Specifies the IP address of other RP routers
           that share the same RP IP address.";
      }
    }
  }
  leaf rp-address {
    when "../provider-managed/enabled = 'false'" {
      description
        "Relevant when the RP is not managed by the
         provider.";
    }
    type inet:ip-address;
```

```
        mandatory true;
        description
          "Defines the address of the RP.
           Used if the RP is managed by the customer.";
      }
      container groups {
        description
          "Multicast groups associated with the RP.";
        list group {
          key "id";
          description
            "List of multicast groups.";
          leaf id {
            type uint16;
            description
              "Identifier for the group.";
          }
          choice group-format {
            mandatory true;
            description
              "Choice for multicast group format.";
            case group-prefix {
              leaf group-address {
                type inet:ip-prefix;
                description
                  "A single multicast group prefix.";
              }
            }
            case startend {
              leaf group-start {
                type inet:ip-address;
                description
                  "The first multicast group address in
                   the multicast group address range.";
              }
              leaf group-end {
                type inet:ip-address;
                description
                  "The last multicast group address in
                   the multicast group address range.";
              }
            }
          }
        }
      }
    }
  }
  container rp-discovery {
    description
      "RP discovery parameters.";
    leaf rp-discovery-type {
      type identityref {
        base vpn-common:multicast-rp-discovery-type;
      }
      default "vpn-common:static-rp";
      description
        "Type of RP discovery used.";
    }
    container bsr-candidates {
      when "derived-from-or-self(../rp-discovery-type, "
         + "'vpn-common:bsr-rp')" {
        description
          "Only applicable if the discovery type
           is 'bsr-rp'.";
      }
      description
        "Container for the customer Bootstrap Router (BSR)
         candidate's addresses.";
      leaf-list bsr-candidate-address {
        type inet:ip-address;
        description
```

```
            "Specifies the address of the candidate BSR.";
        }
      }
    }
  }
  container igmp {
    if-feature "vpn-common:igmp and vpn-common:ipv4";
    description
      "Includes IGMP-related parameters.";
    list static-group {
      key "group-addr";
      description
        "Multicast static source/group associated with the
         IGMP session.";
      leaf group-addr {
        type rt-types:ipv4-multicast-group-address;
        description
          "Multicast group IPv4 address.";
      }
      leaf source-addr {
        type rt-types:ipv4-multicast-source-address;
        description
          "Multicast source IPv4 address.";
      }
    }
    leaf max-groups {
      type uint32;
      description
        "Indicates the maximum number of groups.";
    }
    leaf max-entries {
      type uint32;
      description
        "Indicates the maximum number of IGMP entries.";
    }
    leaf version {
      type identityref {
        base vpn-common:igmp-version;
      }
      default "vpn-common:igmpv2";
      description
        "Indicates the IGMP version.";
      reference
        "RFC 1112: Host Extensions for IP Multicasting
         RFC 2236: Internet Group Management Protocol,
                   Version 2
         RFC 3376: Internet Group Management Protocol,
                   Version 3";
    }
  }
  container mld {
    if-feature "vpn-common:mld and vpn-common:ipv6";
    description
      "Includes MLD-related parameters.";
    list static-group {
      key "group-addr";
      description
        "Multicast static source/group associated with the
         MLD session.";
      leaf group-addr {
        type rt-types:ipv6-multicast-group-address;
        description
          "Multicast group IPv6 address.";
      }
      leaf source-addr {
        type rt-types:ipv6-multicast-source-address;
        description
          "Multicast source IPv6 address.";
      }
    }
    leaf max-groups {
```

```
        type uint32;
        description
          "Indicates the maximum number of groups.";
      }
      leaf max-entries {
        type uint32;
        description
          "Indicates the maximum number of MLD entries.";
      }
      leaf version {
        type identityref {
          base vpn-common:mld-version;
        }
        default "vpn-common:mldv2";
        description
          "Indicates the MLD protocol version.";
        reference
          "RFC 2710: Multicast Listener Discovery (MLD) for IPv6
           RFC 3810: Multicast Listener Discovery Version 2
                     (MLDv2) for IPv6";
      }
    }
    container pim {
      if-feature "vpn-common:pim";
      description
        "Only applies when the protocol type is 'pim'.";
      leaf hello-interval {
        type rt-types:timer-value-seconds16;
        default "30";
        description
          "Interval between PIM Hello messages.  If set to
           'infinity' or 'not-set', no periodic Hello messages
           are sent.";
        reference
          "RFC 7761: Protocol Independent Multicast - Sparse
                     Mode (PIM-SM): Protocol Specification
                     (Revised), Section 4.11
           RFC 8294: Common YANG Data Types for the Routing
                     Area";
      }
      leaf dr-priority {
        type uint32;
        default "1";
        description
          "Indicates the preference associated with the
           Designated Router (DR) election process.  A larger
           value has a higher priority over a smaller value.";
        reference
          "RFC 7761: Protocol Independent Multicast - Sparse
                     Mode (PIM-SM): Protocol Specification
                     (Revised), Section 4.3.2";
      }
    }
  }
}

/* Main Blocks */
/* Main l3vpn-ntw */

container l3vpn-ntw {
  description
    "Main container for management of Layer 3 Virtual Private
     Network (L3VPN) services.";
  container vpn-profiles {
    description
      "Contains a set of valid VPN profiles to reference
       in the VPN service.";
    uses vpn-common:vpn-profile-cfg;
  }
  container vpn-services {
    description
```

```
  "Container for the VPN services.";
list vpn-service {
  key "vpn-id";
  description
    "List of VPN services.";
  uses vpn-common:vpn-description;
  leaf parent-service-id {
    type vpn-common:vpn-id;
    description
      "Pointer to the parent service, if any.
       A parent service can be an L3SM, a slice request,
       a VPN+ service, etc.";
  }
  leaf vpn-type {
    type identityref {
      base vpn-common:service-type;
    }
    description
      "Indicates the service type.";
  }
  leaf vpn-service-topology {
    type identityref {
      base vpn-common:vpn-topology;
    }
    default "vpn-common:any-to-any";
    description
      "VPN service topology.";
  }
  uses vpn-common:service-status;
  container vpn-instance-profiles {
    description
      "Container for a list of VPN instance profiles.";
    list vpn-instance-profile {
      key "profile-id";
      description
        "List of VPN instance profiles.";
      leaf profile-id {
        type string;
        description
          "VPN instance profile identifier.";
      }
      leaf role {
        type identityref {
          base vpn-common:role;
        }
        default "vpn-common:any-to-any-role";
        description
          "Role of the VPN node in the VPN.";
      }
      uses vpn-instance-profile;
    }
  }
  container underlay-transport {
    description
      "Container for the underlay transport.";
    uses vpn-common:underlay-transport;
  }
  container external-connectivity {
    if-feature "vpn-common:external-connectivity";
    description
      "Container for external connectivity.";
    choice profile {
      description
        "Choice for the external connectivity profile.";
      case profile {
        leaf profile-name {
          type leafref {
            path "/l3vpn-ntw/vpn-profiles"
               + "/valid-provider-identifiers"
               + "/external-connectivity-identifier/id";
          }
```

```
        description
          "Name of the service provider's profile to be
           applied at the VPN service level.";
      }
    }
  }
}
container vpn-nodes {
  description
    "Container for VPN nodes.";
  list vpn-node {
    key "vpn-node-id";
    description
      "Includes a list of VPN nodes.";
    leaf vpn-node-id {
      type vpn-common:vpn-id;
      description
        "An identifier of the VPN node.";
    }
    leaf description {
      type string;
      description
        "Textual description of the VPN node.";
    }
    leaf ne-id {
      type string;
      description
        "Unique identifier of the network element where
         the VPN node is deployed.";
    }
    leaf local-as {
      if-feature "vpn-common:rtg-bgp";
      type inet:as-number;
      description
        "Provider's AS number.  Used if the customer
         requests BGP routing.";
    }
    leaf router-id {
      type rt-types:router-id;
      description
        "A 32-bit number in the dotted-quad format that is
         used to uniquely identify a node within an AS.
         This identifier is used for both IPv4 and IPv6.";
    }
    container active-vpn-instance-profiles {
      description
        "Container for active VPN instance profiles.";
      list vpn-instance-profile {
        key "profile-id";
        description
          "Includes a list of active VPN instance
           profiles.";
        leaf profile-id {
          type leafref {
            path "/l3vpn-ntw/vpn-services/vpn-service"
               + "/vpn-instance-profiles"
               + "/vpn-instance-profile/profile-id";
          }
          description
            "Node's active VPN instance profile.";
        }
        list router-id {
          key "address-family";
          description
            "Router ID per address family.";
          leaf address-family {
            type identityref {
              base vpn-common:address-family;
            }
            description
              "Indicates the address family for which the
```

```
            Router ID applies.";
        }
        leaf router-id {
          type inet:ip-address;
          description
            "The 'router-id' information can be an IPv4
             or IPv6 address.  This can be used,
             for example, to configure an IPv6 address
             as a Router ID when such a capability is
             supported by underlay routers.  In such a
             case, the configured value overrides the
             generic value defined at the VPN node
             level.";
        }
      }
      uses vpn-instance-profile;
    }
  }
  container msdp {
    if-feature "msdp";
    description
      "Includes MSDP-related parameters.";
    leaf peer {
      type inet:ipv4-address;
      description
        "Indicates the IPv4 address of the MSDP peer.";
    }
    leaf local-address {
      type inet:ipv4-address;
      description
        "Indicates the IPv4 address of the local end.
         This local address must be configured on
         the node.";
    }
    uses vpn-common:service-status;
  }
  uses vpn-common:vpn-components-group;
  uses vpn-common:service-status;
  container vpn-network-accesses {
    description
      "List of network accesses.";
    list vpn-network-access {
      key "id";
      description
        "List of network accesses.";
      leaf id {
        type vpn-common:vpn-id;
        description
          "Identifier for the network access.";
      }
      leaf interface-id {
        type string;
        description
          "Identifier for the physical or logical
           interface.
           The identification of the sub-interface
           is provided at the connection level and/or
           the IP connection level.";
      }
      leaf description {
        type string;
        description
          "Textual description of the network access.";
      }
      leaf vpn-network-access-type {
        type identityref {
          base vpn-common:site-network-access-type;
        }
        default "vpn-common:point-to-point";
        description
          "Describes the type of connection, e.g.,
```

```
     point to point.";
}
leaf vpn-instance-profile {
  type leafref {
    path "/l3vpn-ntw/vpn-services/vpn-service"
       + "/vpn-nodes/vpn-node"
       + "/active-vpn-instance-profiles"
       + "/vpn-instance-profile/profile-id";
  }
  description
    "An identifier of an active VPN instance
     profile.";
}
uses vpn-common:service-status;
container connection {
  description
    "Defines Layer 2 protocols and parameters that
     are required to enable connectivity between
     the PE and the CE.";
  container encapsulation {
    description
      "Container for Layer 2 encapsulation.";
    leaf type {
      type identityref {
        base vpn-common:encapsulation-type;
      }
      default "vpn-common:priority-tagged";
      description
        "Encapsulation type.  By default, the type
         of the tagged interface is
         'priority-tagged'.";
    }
    container dot1q {
      when "derived-from-or-self(../type, "
         + "'vpn-common:dot1q')" {
        description
          "Only applies when the type of the
           tagged interface is 'dot1q'.";
      }
      description
        "Tagged interface.";
      leaf tag-type {
        type identityref {
          base vpn-common:tag-type;
        }
        default "vpn-common:c-vlan";
        description
          "Tag type.  By default, the tag type is
           'c-vlan'.";
      }
      leaf cvlan-id {
        type uint16 {
          range "1..4094";
        }
        description
          "VLAN identifier.";
      }
    }
    container priority-tagged {
      when "derived-from-or-self(../type, "
         + "'vpn-common:priority-tagged')" {
        description
          "Only applies when the type of
           the tagged interface is
           'priority-tagged'.";
      }
      description
        "Priority tagged.";
      leaf tag-type {
        type identityref {
          base vpn-common:tag-type;
```

```
      }
      default "vpn-common:c-vlan";
      description
        "Tag type.  By default, the tag type is
         'c-vlan'.";
    }
  }
  container qinq {
    when "derived-from-or-self(../type, "
       + "'vpn-common:qinq')" {
      description
        "Only applies when the type of the
         tagged interface is 'qinq'.";
    }
    description
      "Includes QinQ parameters.";
    leaf tag-type {
      type identityref {
        base vpn-common:tag-type;
      }
      default "vpn-common:s-c-vlan";
      description
        "Tag type.";
    }
    leaf svlan-id {
      type uint16;
      mandatory true;
      description
        "Service VLAN (S-VLAN) identifier.";
    }
    leaf cvlan-id {
      type uint16;
      mandatory true;
      description
        "Customer VLAN (C-VLAN) identifier.";
    }
  }
}
choice l2-service {
  description
    "The Layer 2 connectivity service can be
     provided by indicating a pointer to an
     L2VPN or by specifying a Layer 2 tunnel
     service.";
  container l2-tunnel-service {
    description
      "Defines a Layer 2 tunnel termination.
       It is only applicable when a tunnel is
       required.  The supported values are
       'pseudowire', 'vpls', and 'vxlan'.  Other
       values may be defined, if needed.";
    leaf type {
      type identityref {
        base l2-tunnel-type;
      }
      description
        "Selects the tunnel termination option
         for each VPN network access.";
    }
    container pseudowire {
      when "derived-from-or-self(../type, "
         + "'pseudowire')" {
        description
          "Only applies when the Layer 2 service
           type is 'pseudowire'.";
      }
      description
        "Includes pseudowire termination
         parameters.";
      leaf vcid {
        type uint32;
```

```
      description
        "Indicates a pseudowire (PW) or
         virtual circuit (VC) identifier.";
    }
    leaf far-end {
      type union {
        type uint32;
        type inet:ip-address;
      }
      description
        "Neighbor reference.";
      reference
        "RFC 8077: Pseudowire Setup and
                   Maintenance Using the Label
                   Distribution Protocol
                   (LDP), Section 6.1";
    }
  }
  container vpls {
    when "derived-from-or-self(../type, "
       + "'vpls')" {
      description
        "Only applies when the Layer 2 service
         type is 'vpls'.";
    }
    description
      "VPLS termination parameters.";
    leaf vcid {
      type uint32;
      description
        "VC identifier.";
    }
    leaf-list far-end {
      type union {
        type uint32;
        type inet:ip-address;
      }
      description
        "Neighbor reference.";
    }
  }
  container vxlan {
    when "derived-from-or-self(../type, "
       + "'vxlan')" {
      description
        "Only applies when the Layer 2 service
         type is 'vxlan'.";
    }
    description
      "VXLAN termination parameters.";
    leaf vni-id {
      type uint32;
      mandatory true;
      description
        "VXLAN Network Identifier (VNI).";
    }
    leaf peer-mode {
      type identityref {
        base vpn-common:vxlan-peer-mode;
      }
      default "vpn-common:static-mode";
      description
        "Specifies the VXLAN access mode.  By
         default, the peer mode is set to
         'static-mode'.";
    }
    leaf-list peer-ip-address {
      type inet:ip-address;
      description
        "List of a peer's IP addresses.";
    }
```

```
        }
      }
      case l2vpn {
        leaf l2vpn-id {
          type vpn-common:vpn-id;
          description
            "Indicates the L2VPN service associated
             with an Integrated Routing and Bridging
             (IRB) interface.";
        }
      }
    }
    leaf l2-termination-point {
      type string;
      description
        "Specifies a reference to a local Layer 2
         termination point, such as a Layer 2
         sub-interface.";
    }
    leaf local-bridge-reference {
      type string;
      description
        "Specifies a local bridge reference to
         accommodate, for example, implementations
         that require internal bridging.
         A reference may be a local bridge domain.";
    }
    leaf bearer-reference {
      if-feature "vpn-common:bearer-reference";
      type string;
      description
        "This is an internal reference for the
         service provider to identify the bearer
         associated with this VPN.";
    }
    container lag-interface {
      if-feature "vpn-common:lag-interface";
      description
        "Container for configuration of Link
         Aggregation Group (LAG) interface
         attributes.";
      leaf lag-interface-id {
        type string;
        description
          "LAG interface identifier.";
      }
      container member-link-list {
        description
          "Container for the member link list.";
        list member-link {
          key "name";
          description
            "Member link.";
          leaf name {
            type string;
            description
              "Member link name.";
          }
        }
      }
    }
  }
  container ip-connection {
    description
      "Defines IP connection parameters.";
    leaf l3-termination-point {
      type string;
      description
        "Specifies a reference to a local Layer 3
         termination point, such as a bridge domain
         interface.";
```

```
}
container ipv4 {
  if-feature "vpn-common:ipv4";
  description
    "IPv4-specific parameters.";
  leaf local-address {
    type inet:ipv4-address;
    description
      "The IP address used at the provider's
       interface.";
  }
  leaf prefix-length {
    type uint8 {
      range "0..32";
    }
    description
      "Subnet prefix length expressed in bits.
       It is applied to both local and customer
       addresses.";
  }
  leaf address-allocation-type {
    type identityref {
      base address-allocation-type;
    }
    must "not(derived-from-or-self(current(), "
       + "'slaac') or "
       + "derived-from-or-self(current(), "
       + "'provider-dhcp-slaac'))" {
      error-message "SLAAC is only applicable "
                  + "to IPv6.";
    }
    description
      "Defines how addresses are allocated to
       the peer site.

       If there is no value for the address
       allocation type, then IPv4 addressing
       is not enabled.";
  }
  choice allocation-type {
    description
      "Choice of the IPv4 address allocation.";
    case provider-dhcp {
      description
        "Parameters related to DHCP-allocated
         addresses.  IP addresses are allocated
         by DHCP, which is provided by the
         operator.";
      leaf dhcp-service-type {
        type enumeration {
          enum server {
            description
              "Local DHCP server.";
          }
          enum relay {
            description
              "Local DHCP relay.  DHCP requests
               are relayed to a provider's
               server.";
          }
        }
        description
          "Indicates the type of DHCP service to
           be enabled on this access.";
      }
      choice service-type {
        description
          "Choice based on the DHCP service
           type.";
        case relay {
          description
```

```
        "Container for a list of the
         provider's DHCP servers (i.e.,
         'dhcp-service-type' is set to
         'relay').";
      leaf-list server-ip-address {
        type inet:ipv4-address;
        description
          "IPv4 addresses of the provider's
           DHCP server, for use by the local
           DHCP relay.";
      }
    }
    case server {
      description
        "A choice for how addresses are
         assigned when a local DHCP server
         is enabled.";
      choice address-assign {
        default "number";
        description
          "A choice for how IPv4 addresses
           are assigned.";
        case number {
          leaf number-of-dynamic-address {
            type uint16;
            default "1";
            description
              "Specifies the number of IP
               addresses to be assigned to
               the customer on this
               access.";
          }
        }
        case explicit {
          container customer-addresses {
            description
              "Container for customer
               addresses to be allocated
               using DHCP.";
            list address-pool {
              key "pool-id";
              description
                "Describes IP addresses to
                 be allocated by DHCP.

                 When only 'start-address'
                 is present, it represents a
                 single address.

                 When both 'start-address'
                 and 'end-address' are
                 specified, it implies a
                 range inclusive of both
                 addresses.";
              leaf pool-id {
                type string;
                description
                  "A pool identifier for the
                   address range from
                   'start-address' to
                   'end-address'.";
              }
              leaf start-address {
                type inet:ipv4-address;
                mandatory true;
                description
                  "Indicates the first
                   address in the pool.";
              }
              leaf end-address {
                type inet:ipv4-address;
```

```
                    description
                      "Indicates the last
                       address in the pool.";
                  }
                }
              }
            }
          }
        }
      }
      case dhcp-relay {
        description
          "The DHCP relay is provided by the
           operator.";
        container customer-dhcp-servers {
          description
            "Container for a list of the
             customer's DHCP servers.";
          leaf-list server-ip-address {
            type inet:ipv4-address;
            description
              "IPv4 addresses of the customer's
               DHCP server.";
          }
        }
      }
      case static-addresses {
        description
          "Lists the IPv4 addresses that are
           used.";
        leaf primary-address {
          type leafref {
            path "../address/address-id";
          }
          description
            "Primary address of the connection.";
        }
        list address {
          key "address-id";
          description
            "Lists the IPv4 addresses that are
             used.";
          leaf address-id {
            type string;
            description
              "An identifier of the static IPv4
               address.";
          }
          leaf customer-address {
            type inet:ipv4-address;
            description
              "IPv4 address of the customer
               side.";
          }
        }
      }
    }
  }
  container ipv6 {
    if-feature "vpn-common:ipv6";
    description
      "IPv6-specific parameters.";
    leaf local-address {
      type inet:ipv6-address;
      description
        "IPv6 address of the provider side.";
    }
    leaf prefix-length {
      type uint8 {
        range "0..128";
```

```
    }
    description
      "Subnet prefix length expressed in bits.
       It is applied to both local and customer
       addresses.";
  }
  leaf address-allocation-type {
    type identityref {
      base address-allocation-type;
    }
    description
      "Defines how addresses are allocated.
       If there is no value for the address
       allocation type, then IPv6 addressing is
       disabled.";
  }
  choice allocation-type {
    description
      "A choice based on the IPv6 allocation
       type.";
    container provider-dhcp {
      when "derived-from-or-self(../address-allo"
         + "cation-type, 'provider-dhcp') or "
         + "derived-from-or-self(../address-allo"
         + "cation-type, 'provider-dhcp-slaac')" {
        description
          "Only applies when addresses are
           allocated by DHCPv6 as provided by
           the operator.";
      }
      description
        "Parameters related to DHCP-allocated
         addresses.";
      leaf dhcp-service-type {
        type enumeration {
          enum server {
            description
              "Local DHCPv6 server.";
          }
          enum relay {
            description
              "DHCPv6 relay.";
          }
        }
        description
          "Indicates the type of the DHCPv6
           service to be enabled on this
           access.";
      }
      choice service-type {
        description
          "Choice based on the DHCPv6 service
           type.";
        case relay {
          leaf-list server-ip-address {
            type inet:ipv6-address;
            description
              "IPv6 addresses of the provider's
               DHCPv6 server.";
          }
        }
        case server {
          choice address-assign {
            default "number";
            description
              "Choice for how IPv6 prefixes are
               assigned by the DHCPv6 server.";
            case number {
              leaf number-of-dynamic-address {
                type uint16;
                default "1";
```

```
            description
              "Describes the number of IPv6
               prefixes that are allocated
               to the customer on this
               access.";
          }
        }
        case explicit {
          container customer-addresses {
            description
              "Container for customer IPv6
               addresses allocated by
               DHCPv6.";
            list address-pool {
              key "pool-id";
              description
                "Describes IPv6 addresses
                 allocated by DHCPv6.

                 When only 'start-address'
                 is present, it represents a
                 single address.

                 When both 'start-address'
                 and 'end-address' are
                 specified, it implies a
                 range inclusive of both
                 addresses.";
              leaf pool-id {
                type string;
                description
                  "A pool identifier for the
                   address range from
                   'start-address' to
                   'end-address'.";
              }
              leaf start-address {
                type inet:ipv6-address;
                mandatory true;
                description
                  "Indicates the first
                   address.";
              }
              leaf end-address {
                type inet:ipv6-address;
                description
                  "Indicates the last
                   address.";
              }
            }
          }
        }
      }
    }
  }
}
case dhcp-relay {
  description
    "DHCPv6 relay provided by the
     operator.";
  container customer-dhcp-servers {
    description
      "Container for a list of the
       customer's DHCP servers.";
    leaf-list server-ip-address {
      type inet:ipv6-address;
      description
        "Contains the IP addresses of the
         customer's DHCPv6 server.";
    }
  }
```

```
        }
        case static-addresses {
          description
            "IPv6-specific parameters for static
             allocation.";
          leaf primary-address {
            type leafref {
              path "../address/address-id";
            }
            description
              "Principal address of the
               connection.";
          }
          list address {
            key "address-id";
            description
              "Describes IPv6 addresses that are
               used.";
            leaf address-id {
              type string;
              description
                "An identifier of an IPv6 address.";
            }
            leaf customer-address {
              type inet:ipv6-address;
              description
                "An IPv6 address of the customer
                 side.";
            }
          }
        }
      }
    }
  }
  container routing-protocols {
    description
      "Defines routing protocols.";
    list routing-protocol {
      key "id";
      description
        "List of routing protocols used on the
         CE-PE link.  This list can be augmented.";
      leaf id {
        type string;
        description
          "Unique identifier for the routing
           protocol.";
      }
      leaf type {
        type identityref {
          base vpn-common:routing-protocol-type;
        }
        description
          "Type of routing protocol.";
      }
      list routing-profiles {
        key "id";
        description
          "Routing profiles.";
        leaf id {
          type leafref {
            path "/l3vpn-ntw/vpn-profiles"
               + "/valid-provider-identifiers"
               + "/routing-profile-identifier/id";
          }
          description
            "Routing profile to be used.";
        }
        leaf type {
          type identityref {
            base vpn-common:ie-type;
```

```
    }
    description
      "Import, export, or both.";
  }
}
container static {
  when "derived-from-or-self(../type, "
     + "'vpn-common:static-routing')" {
    description
      "Only applies when the protocol is a
       static routing protocol.";
  }
  description
    "Configuration specific to static
     routing.";
  container cascaded-lan-prefixes {
    description
      "LAN prefixes from the customer.";
    list ipv4-lan-prefixes {
      if-feature "vpn-common:ipv4";
      key "lan next-hop";
      description
        "List of LAN prefixes for the site.";
      leaf lan {
        type inet:ipv4-prefix;
        description
          "LAN prefixes.";
      }
      leaf lan-tag {
        type string;
        description
          "Internal tag to be used in VPN
           policies.";
      }
      leaf next-hop {
        type union {
          type inet:ip-address;
          type predefined-next-hop;
        }
        description
          "The next hop that is to be used
           for the static route.  This may be
           specified as an IP address or a
           predefined next-hop type (e.g.,
           'discard' or 'local-link').";
      }
      leaf bfd-enable {
        if-feature "vpn-common:bfd";
        type boolean;
        description
          "Enables Bidirectional Forwarding
           Detection (BFD).";
      }
      leaf metric {
        type uint32;
        description
          "Indicates the metric associated
           with the static route.";
      }
      leaf preference {
        type uint32;
        description
          "Indicates the preference associated
           with the static route.";
      }
      uses vpn-common:service-status;
    }
    list ipv6-lan-prefixes {
      if-feature "vpn-common:ipv6";
      key "lan next-hop";
      description
```

```
          "List of LAN prefixes for the site.";
        leaf lan {
          type inet:ipv6-prefix;
          description
            "LAN prefixes.";
        }
        leaf lan-tag {
          type string;
          description
            "Internal tag to be used in VPN
             policies.";
        }
        leaf next-hop {
          type union {
            type inet:ip-address;
            type predefined-next-hop;
          }
          description
            "The next hop that is to be used for
             the static route.  This may be
             specified as an IP address or a
             predefined next-hop type (e.g.,
             'discard' or 'local-link').";
        }
        leaf bfd-enable {
          if-feature "vpn-common:bfd";
          type boolean;
          description
            "Enables BFD.";
        }
        leaf metric {
          type uint32;
          description
            "Indicates the metric associated
             with the static route.";
        }
        leaf preference {
          type uint32;
          description
            "Indicates the preference associated
             with the static route.";
        }
        uses vpn-common:service-status;
      }
    }
  }
  container bgp {
    when "derived-from-or-self(../type, "
       + "'vpn-common:bgp-routing')" {
      description
        "Only applies when the protocol is
         BGP.";
    }
    description
      "Configuration specific to BGP.";
    leaf description {
      type string;
      description
        "Includes a description of the BGP
         session.

         This description is meant to be used
         for diagnostic purposes.  The semantic
         of the description is local to an
         implementation.";
    }
    leaf local-as {
      type inet:as-number;
      description
        "Indicates a local AS Number (ASN), if
         an ASN distinct from the ASN configured
```

```
     at the VPN node level is needed.";
}
leaf peer-as {
  type inet:as-number;
  mandatory true;
  description
    "Indicates the customer's ASN when
     the customer requests BGP routing.";
}
leaf address-family {
  type identityref {
    base vpn-common:address-family;
  }
  description
    "This node contains the address families
     to be activated.  'dual-stack' means
     that both IPv4 and IPv6 will be
     activated.";
}
leaf local-address {
  type union {
    type inet:ip-address;
    type if:interface-ref;
  }
  description
    "Sets the local IP address to use for
     the BGP transport session.  This may be
     expressed as either an IP address or a
     reference to an interface.";
}
leaf-list neighbor {
  type inet:ip-address;
  description
    "IP address(es) of the BGP neighbor.
     IPv4 and IPv6 neighbors may be
     indicated if two sessions will be used
     for IPv4 and IPv6.";
}
leaf multihop {
  type uint8;
  description
    "Describes the number of IP hops allowed
     between a given BGP neighbor and
     the PE.";
}
leaf as-override {
  type boolean;
  default "false";
  description
    "Defines whether ASN override is
     enabled, i.e., replacing the ASN of
     the customer specified in the AS_PATH
     attribute with the local ASN.";
}
leaf allow-own-as {
  type uint8;
  default "0";
  description
    "If set, specifies the maximum number of
     occurrences of the provider's ASN that
     are permitted within the AS_PATH
     before it is rejected.";
}
leaf prepend-global-as {
  type boolean;
  default "false";
  description
    "In some situations, the ASN that is
     provided at the VPN node level may be
     distinct from the ASN configured at the
     VPN network access level.  When such
```

```
        ASNs are provided, they are both
        prepended to the BGP route updates
        for this access.  To disable that
        behavior, 'prepend-global-as'
        must be set to 'false'.  In such a
        case, the ASN that is provided at
        the VPN node level is not prepended
        to the BGP route updates for
        this access.";
    }
    leaf send-default-route {
      type boolean;
      default "false";
      description
        "Defines whether default routes can be
         advertised to a peer.  If set, the
         default routes are advertised to a
         peer.";
    }
    leaf site-of-origin {
      when "../address-family = 'vpn-common:ipv4' "
         + "or 'vpn-common:dual-stack'" {
        description
          "Only applies if IPv4 is activated.";
      }
      type rt-types:route-origin;
      description
        "The Site of Origin attribute is encoded
         as a Route Origin Extended Community.
         It is meant to uniquely identify the
         set of routes learned from a site via a
         particular CE-PE connection and is used
         to prevent routing loops.";
      reference
        "RFC 4364: BGP/MPLS IP Virtual Private
                   Networks (VPNs), Section 7";
    }
    leaf ipv6-site-of-origin {
      when "../address-family = 'vpn-common:ipv6' "
         + "or 'vpn-common:dual-stack'" {
        description
          "Only applies if IPv6 is activated.";
      }
      type rt-types:ipv6-route-origin;
      description
        "The IPv6 Site of Origin attribute is
         encoded as an IPv6 Route Origin
         Extended Community.  It is meant to
         uniquely identify the set of routes
         learned from a site via VRF
         information.";
      reference
        "RFC 5701: IPv6 Address Specific BGP
                   Extended Community
                   Attribute";
    }
    list redistribute-connected {
      key "address-family";
      description
        "Indicates, per address family, the
         policy to follow for connected
         routes.";
      leaf address-family {
        type identityref {
          base vpn-common:address-family;
        }
        description
          "Indicates the address family.";
      }
      leaf enable {
        type boolean;
```

```
    description
      "Enables the redistribution of
       connected routes.";
  }
}
container bgp-max-prefix {
  description
    "Controls the behavior when a prefix
     maximum is reached.";
  leaf max-prefix {
    type uint32;
    default "5000";
    description
      "Indicates the maximum number of BGP
       prefixes allowed in the BGP session.

       It allows control of how many
       prefixes can be received from a
       neighbor.

       If the limit is exceeded, the action
       indicated in 'violate-action' will be
       followed.";
    reference
      "RFC 4271: A Border Gateway Protocol 4
                 (BGP-4), Section 8.2.2";
  }
  leaf warning-threshold {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    units "percent";
    default "75";
    description
      "When this value is reached, a warning
       notification will be triggered.";
  }
  leaf violate-action {
    type enumeration {
      enum warning {
        description
          "Only a warning message is sent to
           the peer when the limit is
           exceeded.";
      }
      enum discard-extra-paths {
        description
          "Discards extra paths when the
           limit is exceeded.";
      }
      enum restart {
        description
          "The BGP session restarts after
           the indicated time interval.";
      }
    }
    description
      "If the BGP neighbor 'max-prefix'
       limit is reached, the action
       indicated in 'violate-action'
       will be followed.";
  }
  leaf restart-timer {
    type uint32;
    units "seconds";
    description
      "Time interval after which the BGP
       session will be reestablished.";
  }
}
```

```
container bgp-timers {
  description
    "Includes two BGP timers that can be
     customized when building a VPN service
     with BGP used as the CE-PE routing
     protocol.";
  leaf keepalive {
    type uint16 {
      range "0..21845";
    }
    units "seconds";
    default "30";
    description
      "This timer indicates the KEEPALIVE
       messages' frequency between a PE
       and a BGP peer.

       If set to '0', it indicates that
       KEEPALIVE messages are disabled.

       It is suggested that the maximum
       time between KEEPALIVE messages be
       one-third of the Hold Time
       interval.";
    reference
      "RFC 4271: A Border Gateway Protocol 4
                 (BGP-4), Section 4.4";
  }
  leaf hold-time {
    type uint16 {
      range "0 | 3..65535";
    }
    units "seconds";
    default "90";
    description
      "Indicates the maximum number of
       seconds that may elapse between the
       receipt of successive KEEPALIVE
       and/or UPDATE messages from the peer.

       The Hold Time must be either zero or
       at least three seconds.";
    reference
      "RFC 4271: A Border Gateway Protocol 4
                 (BGP-4), Section 4.2";
  }
}
container authentication {
  description
    "Container for BGP authentication
     parameters between a PE and a CE.";
  leaf enable {
    type boolean;
    default "false";
    description
      "Enables or disables authentication.";
  }
  container keying-material {
    when "../enable = 'true'";
    description
      "Container for describing how a BGP
       routing session is to be secured
       between a PE and a CE.";
    choice option {
      description
        "Choice of authentication options.";
      case ao {
        description
          "Uses the TCP Authentication
           Option (TCP-AO).";
        reference
```

```
        "RFC 5925: The TCP Authentication
                   Option";
      leaf enable-ao {
        type boolean;
        description
          "Enables the TCP-AO.";
      }
      leaf ao-keychain {
        type key-chain:key-chain-ref;
        description
          "Reference to the TCP-AO key
           chain.";
        reference
          "RFC 8177: YANG Data Model for
                     Key Chains";
      }
    }
    case md5 {
      description
        "Uses MD5 to secure the session.";
      reference
        "RFC 4364: BGP/MPLS IP Virtual
                   Private Networks
                   (VPNs), Section 13.2";
      leaf md5-keychain {
        type key-chain:key-chain-ref;
        description
          "Reference to the MD5 key
           chain.";
        reference
          "RFC 8177: YANG Data Model for
                     Key Chains";
      }
    }
    case explicit {
      leaf key-id {
        type uint32;
        description
          "Key identifier.";
      }
      leaf key {
        type string;
        description
          "BGP authentication key.
           This model only supports the
           subset of keys that are
           representable as ASCII
           strings.";
      }
      leaf crypto-algorithm {
        type identityref {
          base key-chain:crypto-algorithm;
        }
        description
          "Indicates the cryptographic
           algorithm associated with the
           key.";
      }
    }
    case ipsec {
      description
        "Specifies a reference to an
         Internet Key Exchange Protocol
         (IKE) Security Association
         (SA).";
      leaf sa {
        type string;
        description
          "Indicates the
           administrator-assigned name
           of the SA.";
```

```
          }
        }
      }
    }
  }
  uses vpn-common:service-status;
}
container ospf {
  when "derived-from-or-self(../type, "
     + "'vpn-common:ospf-routing')" {
    description
      "Only applies when the protocol is
       OSPF.";
  }
  description
    "Configuration specific to OSPF.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or
       both are to be activated.";
  }
  leaf area-id {
    type yang:dotted-quad;
    mandatory true;
    description
      "Area ID.";
    reference
      "RFC 4577: OSPF as the Provider/Customer
                 Edge Protocol for BGP/MPLS IP
                 Virtual Private Networks
                 (VPNs), Section 4.2.3
       RFC 6565: OSPFv3 as a Provider Edge to
                 Customer Edge (PE-CE) Routing
                 Protocol, Section 4.2";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the PE-CE link.  It is used
       in the routing state calculation and
       path selection.";
  }
  container sham-links {
    if-feature "vpn-common:rtg-ospf-sham-link";
    description
      "List of sham links.";
    reference
      "RFC 4577: OSPF as the Provider/Customer
                 Edge Protocol for BGP/MPLS IP
                 Virtual Private Networks
                 (VPNs), Section 4.2.7
       RFC 6565: OSPFv3 as a Provider Edge to
                 Customer Edge (PE-CE) Routing
                 Protocol, Section 5";
    list sham-link {
      key "target-site";
      description
        "Creates a sham link with another
         site.";
      leaf target-site {
        type string;
        description
          "Target site for the sham link
           connection.  The site is referred
           to by its identifier.";
      }
      leaf metric {
```

```
        type uint16;
        default "1";
        description
          "Metric of the sham link.  It is
           used in the routing state
           calculation and path selection.
           The default value is set to '1'.";
        reference
          "RFC 4577: OSPF as the
                     Provider/Customer Edge
                     Protocol for BGP/MPLS IP
                     Virtual Private Networks
                     (VPNs), Section 4.2.7.3
           RFC 6565: OSPFv3 as a Provider Edge
                     to Customer Edge (PE-CE)
                     Routing Protocol,
                     Section 5.2";
      }
    }
  }
  leaf max-lsa {
    type uint32 {
      range "1..4294967294";
    }
    description
      "Maximum number of allowed Link State
       Advertisements (LSAs) that the OSPF
       instance will accept.";
  }
  container authentication {
    description
      "Authentication configuration.";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enables or disables authentication.";
    }
    container keying-material {
      when "../enable = 'true'";
      description
        "Container for describing how an OSPF
         session is to be secured between a CE
         and a PE.";
      choice option {
        description
          "Options for OSPF authentication.";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Name of the key chain.";
          }
        }
        case auth-key-explicit {
          leaf key-id {
            type uint32;
            description
              "Key identifier.";
          }
          leaf key {
            type string;
            description
              "OSPF authentication key.
               This model only supports the
               subset of keys that are
               representable as ASCII
               strings.";
          }
          leaf crypto-algorithm {
            type identityref {
```

```
              base key-chain:crypto-algorithm;
            }
            description
              "Indicates the cryptographic
               algorithm associated with the
               key.";
          }
        }
        case ipsec {
          leaf sa {
            type string;
            description
              "Indicates the
               administrator-assigned name
               of the SA.";
            reference
              "RFC 4552: Authentication/
                         Confidentiality for
                         OSPFv3";
          }
        }
      }
    }
  }
  uses vpn-common:service-status;
}
container isis {
  when "derived-from-or-self(../type, "
     + "'vpn-common:isis-routing')" {
    description
      "Only applies when the protocol is
       IS-IS.";
  }
  description
    "Configuration specific to IS-IS.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both
       are to be activated.";
  }
  leaf area-address {
    type area-address;
    mandatory true;
    description
      "Area address.";
  }
  leaf level {
    type identityref {
      base vpn-common:isis-level;
    }
    description
      "Can be 'level-1', 'level-2', or
       'level-1-2'.";
    reference
      "RFC 9181: A Common YANG Data Model for
                 Layer 2 and Layer 3 VPNs";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the PE-CE link.  It is used
       in the routing state calculation and
       path selection.";
  }
  leaf mode {
    type enumeration {
      enum active {
```

```
          description
            "The interface sends or receives
             IS-IS protocol control packets.";
        }
        enum passive {
          description
            "Suppresses the sending of IS-IS
             updates through the specified
             interface.";
        }
      }
      default "active";
      description
        "IS-IS interface mode type.";
    }
    container authentication {
      description
        "Authentication configuration.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Enables or disables authentication.";
      }
      container keying-material {
        when "../enable = 'true'";
        description
          "Container for describing how an IS-IS
           session is to be secured between a CE
           and a PE.";
        choice option {
          description
            "Options for IS-IS authentication.";
          case auth-key-chain {
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Name of the key chain.";
            }
          }
          case auth-key-explicit {
            leaf key-id {
              type uint32;
              description
                "Key identifier.";
            }
            leaf key {
              type string;
              description
                "IS-IS authentication key.
                 This model only supports the
                 subset of keys that are
                 representable as ASCII
                 strings.";
            }
            leaf crypto-algorithm {
              type identityref {
                base key-chain:crypto-algorithm;
              }
              description
                "Indicates the cryptographic
                 algorithm associated with the
                 key.";
            }
          }
        }
      }
    }
    uses vpn-common:service-status;
  }
  container rip {
```

```
when "derived-from-or-self(../type, "
   + "'vpn-common:rip-routing')" {
  description
    "Only applies when the protocol is RIP.
     For IPv4, the model assumes that RIP
     version 2 is used.";
}
description
  "Configuration specific to RIP routing.";
leaf address-family {
  type identityref {
    base vpn-common:address-family;
  }
  description
    "Indicates whether IPv4, IPv6, or both
     address families are to be activated.";
}
container timers {
  description
    "Indicates the RIP timers.";
  reference
    "RFC 2453: RIP Version 2";
  leaf update-interval {
    type uint16 {
      range "1..32767";
    }
    units "seconds";
    default "30";
    description
      "Indicates the RIP update time, i.e.,
       the amount of time for which RIP
       updates are sent.";
  }
  leaf invalid-interval {
    type uint16 {
      range "1..32767";
    }
    units "seconds";
    default "180";
    description
      "The interval before a route is
       declared invalid after no updates are
       received.  This value is at least
       three times the value for the
       'update-interval' argument.";
  }
  leaf holddown-interval {
    type uint16 {
      range "1..32767";
    }
    units "seconds";
    default "180";
    description
      "Specifies the interval before better
       routes are released.";
  }
  leaf flush-interval {
    type uint16 {
      range "1..32767";
    }
    units "seconds";
    default "240";
    description
      "Indicates the RIP flush timer, i.e.,
       the amount of time that must elapse
       before a route is removed from the
       routing table.";
  }
}
leaf default-metric {
  type uint8 {
```

```
        range "0..16";
      }
      default "1";
      description
        "Sets the default metric.";
    }
    container authentication {
      description
        "Authentication configuration.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Enables or disables authentication.";
      }
      container keying-material {
        when "../enable = 'true'";
        description
          "Container for describing how a RIP
           session is to be secured between a CE
           and a PE.";
        choice option {
          description
            "Specifies the authentication
             scheme.";
          case auth-key-chain {
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Name of the key chain.";
            }
          }
          case auth-key-explicit {
            leaf key {
              type string;
              description
                "RIP authentication key.
                 This model only supports the
                 subset of keys that are
                 representable as ASCII
                 strings.";
            }
            leaf crypto-algorithm {
              type identityref {
                base key-chain:crypto-algorithm;
              }
              description
                "Indicates the cryptographic
                 algorithm associated with the
                 key.";
            }
          }
        }
      }
    }
    uses vpn-common:service-status;
  }
  container vrrp {
    when "derived-from-or-self(../type, "
       + "'vpn-common:vrrp-routing')" {
      description
        "Only applies when the protocol is the
         Virtual Router Redundancy Protocol
         (VRRP).";
    }
    description
      "Configuration specific to VRRP.";
    reference
      "RFC 5798: Virtual Router Redundancy
                 Protocol (VRRP) Version 3 for
                 IPv4 and IPv6";
```

```
        leaf address-family {
          type identityref {
            base vpn-common:address-family;
          }
          description
            "Indicates whether IPv4, IPv6, or both
             address families are to be enabled.";
        }
        leaf vrrp-group {
          type uint8 {
            range "1..255";
          }
          description
            "Includes the VRRP group identifier.";
        }
        leaf backup-peer {
          type inet:ip-address;
          description
            "Indicates the IP address of the peer.";
        }
        leaf-list virtual-ip-address {
          type inet:ip-address;
          description
            "Virtual IP addresses for a single VRRP
             group.";
          reference
            "RFC 5798: Virtual Router Redundancy
                       Protocol (VRRP) Version 3 for
                       IPv4 and IPv6,
                       Sections 1.2 and 1.3";
        }
        leaf priority {
          type uint8 {
            range "1..254";
          }
          default "100";
          description
            "Sets the local priority of the VRRP
             speaker.";
        }
        leaf ping-reply {
          type boolean;
          default "false";
          description
            "Controls whether the VRRP speaker
             should reply to ping requests.";
        }
        uses vpn-common:service-status;
      }
    }
  }
  container oam {
    description
      "Defines the Operations, Administration,
       and Maintenance (OAM) mechanisms used.

       BFD is set as a fault detection mechanism,
       but other mechanisms can be defined in the
       future.";
    container bfd {
      if-feature "vpn-common:bfd";
      description
        "Container for BFD.";
      leaf session-type {
        type identityref {
          base vpn-common:bfd-session-type;
        }
        default "vpn-common:classic-bfd";
        description
          "Specifies the BFD session type.";
      }
```

```
leaf desired-min-tx-interval {
  type uint32;
  units "microseconds";
  default "1000000";
  description
    "The minimum interval between
     transmissions of BFD Control packets, as
     desired by the operator.";
  reference
    "RFC 5880: Bidirectional Forwarding
               Detection (BFD),
               Section 6.8.7";
}
leaf required-min-rx-interval {
  type uint32;
  units "microseconds";
  default "1000000";
  description
    "The minimum interval between received BFD
     Control packets that the PE should
     support.";
  reference
    "RFC 5880: Bidirectional Forwarding
               Detection (BFD),
               Section 6.8.7";
}
leaf local-multiplier {
  type uint8 {
    range "1..255";
  }
  default "3";
  description
    "Specifies the detection multiplier that
     is transmitted to a BFD peer.

     The detection interval for the receiving
     BFD peer is calculated by multiplying the
     value of the negotiated transmission
     interval by the received detection
     multiplier value.";
  reference
    "RFC 5880: Bidirectional Forwarding
               Detection (BFD),
               Section 6.8.7";
}
leaf holdtime {
  type uint32;
  units "milliseconds";
  description
    "Expected BFD holdtime.

     The customer may impose some fixed
     values for the holdtime period if the
     provider allows the customer to use
     this function.

     If the provider doesn't allow the
     customer to use this function,
     fixed values will not be set.";
  reference
    "RFC 5880: Bidirectional Forwarding
               Detection (BFD),
               Section 6.8.18";
}
leaf profile {
  type leafref {
    path "/l3vpn-ntw/vpn-profiles"
       + "/valid-provider-identifiers"
       + "/bfd-profile-identifier/id";
  }
  description
```

```
          "Well-known service provider profile name.

           The provider can propose some profiles
           to the customer, depending on the
           service level the customer wants to
           achieve.";
      }
      container authentication {
        presence "Enables BFD authentication";
        description
          "Parameters for BFD authentication.";
        leaf key-chain {
          type key-chain:key-chain-ref;
          description
            "Name of the key chain.";
        }
        leaf meticulous {
          type boolean;
          description
            "Enables meticulous mode.";
          reference
            "RFC 5880: Bidirectional Forwarding
                       Detection (BFD),
                       Section 6.7";
        }
      }
      uses vpn-common:service-status;
    }
  }
  container security {
    description
      "Site-specific security parameters.";
    container encryption {
      if-feature "vpn-common:encryption";
      description
        "Container for CE-PE security encryption.";
      leaf enabled {
        type boolean;
        default "false";
        description
          "If set to 'true', traffic encryption on
           the connection is required.  Otherwise,
           it is disabled.";
      }
      leaf layer {
        when "../enabled = 'true'" {
          description
            "Included only when encryption
             is enabled.";
        }
        type enumeration {
          enum layer2 {
            description
              "Encryption occurs at Layer 2.";
          }
          enum layer3 {
            description
              "Encryption occurs at Layer 3.
               For example, IPsec may be used when
               a customer requests Layer 3
               encryption.";
          }
        }
        description
          "Indicates the layer on which encryption
           is applied.";
      }
    }
    container encryption-profile {
      when "../encryption/enabled = 'true'" {
        description
```

```
          "Indicates the layer on which encryption
           is enabled.";
      }
      description
        "Container for the encryption profile.";
      choice profile {
        description
          "Choice for the encryption profile.";
        case provider-profile {
          leaf profile-name {
            type leafref {
              path "/l3vpn-ntw/vpn-profiles"
                 + "/valid-provider-identifiers"
                 + "/encryption-profile-identifier/id";
            }
            description
              "Name of the service provider's
               profile to be applied.";
          }
        }
        case customer-profile {
          leaf customer-key-chain {
            type key-chain:key-chain-ref;
            description
              "Customer-supplied key chain.";
          }
        }
      }
    }
  }
  container service {
    description
      "Service parameters of the attachment.";
    leaf pe-to-ce-bandwidth {
      if-feature "vpn-common:inbound-bw";
      type uint64;
      units "bps";
      description
        "From the customer site's perspective, the
         service inbound bandwidth of the connection
         or download bandwidth from the SP to the
         site.  Note that the L3SM uses
         'input-bandwidth' to refer to the same
         concept.";
    }
    leaf ce-to-pe-bandwidth {
      if-feature "vpn-common:outbound-bw";
      type uint64;
      units "bps";
      description
        "From the customer site's perspective,
         the service outbound bandwidth of the
         connection or upload bandwidth from
         the site to the SP.  Note that the L3SM
         uses 'output-bandwidth' to refer to the
         same concept.";
    }
    leaf mtu {
      type uint32;
      units "bytes";
      description
        "MTU at the service level.  If the service
         is IP, it refers to the IP MTU.  If
         Carriers' Carriers (CsC) is enabled, the
         requested MTU will refer to the MPLS
         maximum labeled packet size and not to the
         IP MTU.";
    }
    container qos {
      if-feature "vpn-common:qos";
      description
```

```
    "QoS configuration.";
  container qos-classification-policy {
    description
      "Configuration of the traffic
       classification policy.";
    uses vpn-common:qos-classification-policy;
  }
  container qos-action {
    description
      "List of QoS action policies.";
    list rule {
      key "id";
      description
        "List of QoS actions.";
      leaf id {
        type string;
        description
          "An identifier of the QoS action
           rule.";
      }
      leaf target-class-id {
        type string;
        description
          "Identification of the class of
           service.  This identifier is internal
           to the administration.";
      }
      leaf inbound-rate-limit {
        type decimal64 {
          fraction-digits 5;
          range "0..100";
        }
        units "percent";
        description
          "Specifies whether/how to rate-limit
           the inbound traffic matching this QoS
           policy.  It is expressed as a percent
           of the value that is indicated in
           'input-bandwidth'.";
      }
      leaf outbound-rate-limit {
        type decimal64 {
          fraction-digits 5;
          range "0..100";
        }
        units "percent";
        description
          "Specifies whether/how to rate-limit
           the outbound traffic matching this
           QoS policy.  It is expressed as a
           percent of the value that is
           indicated in 'output-bandwidth'.";
      }
    }
  }
  container qos-profile {
    description
      "QoS profile configuration.";
    list qos-profile {
      key "profile";
      description
        "QoS profile.
         Can be a standard profile or
         a customized profile.";
      leaf profile {
        type leafref {
          path "/l3vpn-ntw/vpn-profiles"
             + "/valid-provider-identifiers"
             + "/qos-profile-identifier/id";
        }
        description
```

```
            "QoS profile to be used.";
        }
        leaf direction {
          type identityref {
            base vpn-common:qos-profile-direction;
          }
          default "vpn-common:both";
          description
            "The direction to which the QoS
             profile is applied.";
        }
      }
    }
  }
  container carriers-carrier {
    if-feature "vpn-common:carriers-carrier";
    description
      "This container is used when the customer
       provides MPLS-based services.  This is
       only used in the case of CsC (i.e., a
       customer builds an MPLS service using an
       IP VPN to carry its traffic).";
    leaf signaling-type {
      type enumeration {
        enum ldp {
          description
            "Uses LDP as the signaling protocol
             between the PE and the CE.  In this
             case, an IGP routing protocol must
             also be configured.";
        }
        enum bgp {
          description
            "Uses BGP as the signaling protocol
             between the PE and the CE.
             In this case, BGP must also be
             configured as the routing protocol.";
          reference
            "RFC 8277: Using BGP to Bind MPLS
                       Labels to Address
                       Prefixes";
        }
      }
      default "bgp";
      description
        "MPLS signaling type.";
    }
  }
  container ntp {
    description
      "Time synchronization may be needed in some
       VPNs, such as infrastructure and management
       VPNs.  This container includes parameters
       to enable the NTP service.";
    reference
      "RFC 5905: Network Time Protocol Version 4:
                 Protocol and Algorithms
                 Specification";
    leaf broadcast {
      type enumeration {
        enum client {
          description
            "The VPN node will listen to NTP
             broadcast messages on this VPN
             network access.";
        }
        enum server {
          description
            "The VPN node will behave as a
             broadcast server.";
        }
```

```
      }
      description
        "Indicates the NTP broadcast mode to use
         for the VPN network access.";
    }
    container auth-profile {
      description
        "Pointer to a local profile.";
      leaf profile-id {
        type string;
        description
          "A pointer to a local authentication
           profile on the VPN node is provided.";
      }
    }
    uses vpn-common:service-status;
  }
  container multicast {
    if-feature "vpn-common:multicast";
    description
      "Multicast parameters for the network
       access.";
    leaf access-type {
      type enumeration {
        enum receiver-only {
          description
            "The peer site only has receivers.";
        }
        enum source-only {
          description
            "The peer site only has sources.";
        }
        enum source-receiver {
          description
            "The peer site has both sources and
             receivers.";
        }
      }
      default "source-receiver";
      description
        "Type of multicast site.";
    }
    leaf address-family {
      type identityref {
        base vpn-common:address-family;
      }
      description
        "Indicates the address family.";
    }
    leaf protocol-type {
      type enumeration {
        enum host {
          description
            "Hosts are directly connected to the
             provider network.

             Host protocols, such as IGMP or MLD,
             are required.";
        }
        enum router {
          description
            "Hosts are behind a customer router.
             PIM will be implemented.";
        }
        enum both {
          description
            "Some hosts are behind a customer
             router, and some others are directly
             connected to the provider network.
             Both host and routing protocols must
             be used.
```

```
          Typically, IGMP and PIM will be
          implemented.";
      }
    }
    default "both";
    description
      "Multicast protocol type to be used with
       the customer site.";
  }
  leaf remote-source {
    type boolean;
    default "false";
    description
      "A remote multicast source is a source
       that is not on the same subnet as the
       VPN network access.  When set to 'true',
       the multicast traffic from a remote
       source is accepted.";
  }
  container igmp {
    when "../protocol-type = 'host' and "
       + "../address-family = 'vpn-common:ipv4' "
       + "or 'vpn-common:dual-stack'";
    if-feature "vpn-common:igmp";
    description
      "Includes IGMP-related parameters.";
    list static-group {
      key "group-addr";
      description
        "Multicast static source/group
         associated with the IGMP session.";
      leaf group-addr {
        type rt-types:ipv4-multicast-group-address;
        description
          "Multicast group IPv4 address.";
      }
      leaf source-addr {
        type
          rt-types:ipv4-multicast-source-address;
        description
          "Multicast source IPv4 address.";
      }
    }
    leaf max-groups {
      type uint32;
      description
        "Indicates the maximum number of
         groups.";
    }
    leaf max-entries {
      type uint32;
      description
        "Indicates the maximum number of IGMP
         entries.";
    }
    leaf max-group-sources {
      type uint32;
      description
        "The maximum number of group sources.";
    }
    leaf version {
      type identityref {
        base vpn-common:igmp-version;
      }
      default "vpn-common:igmpv2";
      description
        "Indicates the IGMP version.";
    }
    uses vpn-common:service-status;
  }
```

```
container mld {
  when "../protocol-type = 'host' and "
     + "../address-family = 'vpn-common:ipv6' "
     + "or 'vpn-common:dual-stack'";
  if-feature "vpn-common:mld";
  description
    "Includes MLD-related parameters.";
  list static-group {
    key "group-addr";
    description
      "Multicast static source/group associated
       with the MLD session.";
    leaf group-addr {
      type rt-types:ipv6-multicast-group-address;
      description
        "Multicast group IPv6 address.";
    }
    leaf source-addr {
      type
        rt-types:ipv6-multicast-source-address;
      description
        "Multicast source IPv6 address.";
    }
  }
  leaf max-groups {
    type uint32;
    description
      "Indicates the maximum number of
       groups.";
  }
  leaf max-entries {
    type uint32;
    description
      "Indicates the maximum number of MLD
       entries.";
  }
  leaf max-group-sources {
    type uint32;
    description
      "The maximum number of group sources.";
  }
  leaf version {
    type identityref {
      base vpn-common:mld-version;
    }
    default "vpn-common:mldv2";
    description
      "Indicates the MLD protocol version.";
  }
  uses vpn-common:service-status;
}
container pim {
  when "../protocol-type = 'router'";
  if-feature "vpn-common:pim";
  description
    "Only applies when the protocol type is
     'pim'.";
  leaf hello-interval {
    type rt-types:timer-value-seconds16;
    default "30";
    description
      "Interval between PIM Hello messages.
       If set to 'infinity' or 'not-set',
       no periodic Hello messages are sent.";
    reference
      "RFC 7761: Protocol Independent
                 Multicast - Sparse Mode
                 (PIM-SM): Protocol
                 Specification (Revised),
                 Section 4.11
       RFC 8294: Common YANG Data Types for
```

```
                                the Routing Area";
                        }
                        leaf dr-priority {
                          type uint32;
                          default "1";
                          description
                            "Indicates the preference associated
                             with the DR election process.  A larger
                             value has a higher priority over a
                             smaller value.";
                          reference
                            "RFC 7761: Protocol Independent
                                       Multicast - Sparse Mode
                                       (PIM-SM): Protocol
                                       Specification (Revised),
                                       Section 4.3.2";
                        }
                        uses vpn-common:service-status;
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

9. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability in the "ietf-l3vpn-ntw" module:

'vpn-profiles': This container includes a set of sensitive data that influence how the L3VPN service is delivered. For example, an attacker who has access to these data nodes may be able to manipulate routing policies, QoS policies, or encryption properties. These data nodes are defined with "nacm:default-deny-write" tagging [RFC9181].

'vpn-services': An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L3VPN service, interrupting all the traffic of a client. In addition, an attacker may modify the attributes of a running service (e.g., QoS, bandwidth, routing protocols, keying material), leading to malfunctioning of the service and therefore to Service Level Agreement (SLA) violations. In addition, an attacker could attempt to create an L3VPN service or add a new network access. In addition to using NACM to prevent unauthorized access, such

activity can be detected by adequately monitoring and tracking network configuration changes.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

'customer-name' and 'ip-connection': An attacker can retrieve privacy-related information, which can be used to track a customer. Disclosing such information may be considered a violation of the customer-provider trust relationship.

'keying-material': An attacker can retrieve the cryptographic keys protecting the underlying VPN service (CE-PE routing, in particular). These keys could be used to inject spoofed routing advertisements.

Several data nodes ('bgp', 'ospf', 'isis', 'rip', and 'bfd') rely upon [RFC8177] for authentication purposes. Therefore, this module inherits the security considerations discussed in Section 5 of [RFC8177]. Also, these data nodes support supplying explicit keys as strings in ASCII format. The use of keys in hexadecimal string format would afford greater key entropy with the same number of key-string octets. However, such a format is not included in this version of the L3NM, because it is not supported by the underlying device modules (e.g., [RFC8695]).

As discussed in Section 7.6.3, the module supports MD5 to basically accommodate the installed BGP base. MD5 suffers from the security weaknesses discussed in Section 2 of [RFC6151] and Section 2.1 of [RFC6952].

[RFC8633] describes best current practices to be considered in VPNs making use of NTP. Moreover, a mechanism to provide cryptographic security for NTP is specified in [RFC8915].

10. IANA Considerations

IANA has registered the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

IANA has registered the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

Name: ietf-l3vpn-ntw
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw
Prefix: l3nm
Reference: RFC 9182

11. References

11.1. Normative References

[ISO10589] ISO, "Information technology - Telecommunications and information exchange between systems - Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)", ISO/IEC 10589:2002, 2002, <https://www.iso.org/standard/30932.html>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <https://www.rfc-editor.org/info/rfc2080>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, DOI 10.17487/RFC2236, November 1997, <https://www.rfc-editor.org/info/rfc2236>.

[RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <https://www.rfc-editor.org/info/rfc2453>.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <https://www.rfc-editor.org/info/rfc2710>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <https://www.rfc-editor.org/info/rfc4552>.

[RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, DOI 10.17487/RFC4577, June 2006, <https://www.rfc-editor.org/info/rfc4577>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <https://www.rfc-editor.org/info/rfc5701>.

[RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, DOI 10.17487/RFC5709, October 2009, <https://www.rfc-editor.org/info/rfc5709>.

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/info/rfc5798>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/ BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <https://www.rfc-editor.org/info/rfc6513>.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <https://www.rfc-editor.org/info/rfc6514>.

[RFC6565] Pillay-Esnault, P., Moyer, P., Doyle, J., Ertekin, E., and M. Lundberg, "OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol", RFC 6565, DOI 10.17487/RFC6565, June 2012, <https://www.rfc-editor.org/info/rfc6565>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7166] Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, DOI 10.17487/RFC7166, March 2014, <https://www.rfc-editor.org/info/rfc7166>.

[RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <https://www.rfc-editor.org/info/rfc7474>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <https://www.rfc-editor.org/info/rfc8466>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[RFC9181] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., and Q. Wu, "A Common YANG Data Model for Layer 2 and Layer 3 VPNs", RFC 9181, DOI 10.17487/RFC9181, February 2022, <https://www.rfc-editor.org/info/rfc9181>.

11.2. Informative References

[BGP-YANG] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-12, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-12>.

[Enhanced-VPN-Framework]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-09, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-enhanced-vpn-09>.

[IEEE802.1AX]
IEEE, "802.1AX-2020 - IEEE Standard for Local and Metropolitan Area Networks--Link Aggregation", IEEE Std 802.1AX-2020, <https://ieeexplore.ieee.org/document/9105034>.

[Network-Slices-Framework]
Farrel, A., Ed., Gray, E., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, LM., and J. Tantsura,

"Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slices-05, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-ietf-network-slices-05>.

[PIM-YANG] Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and F. Hu, "A YANG Data Model for Protocol Independent Multicast (PIM)", Work in Progress, Internet-Draft, draft-ietf-pim-yang-17, 19 May 2018, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-yang-17>.

[PYANG] "pyang", commit 524cf61, December 2021, <https://github.com/mbj4668/pyang>.

[QoS-YANG] Choudhary, A., Jethanandani, M., Aries, E., and I. Chen, "A YANG Data Model for Quality of Service (QoS)", Work in Progress, Internet-Draft, draft-ietf-rtgwg-qos-model-06, 8 November 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-qos-model-06>.

[RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <https://www.rfc-editor.org/info/rfc3618>.

[RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <https://www.rfc-editor.org/info/rfc3644>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <https://www.rfc-editor.org/info/rfc4110>.

[RFC4176] El Mghazli, Y., Ed., Nadeau, T., Boucadair, M., Chan, K., and A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management", RFC 4176, DOI 10.17487/RFC4176, October 2005, <https://www.rfc-editor.org/info/rfc4176>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC6037] Rosen, E., Ed., Cai, Y., Ed., and IJ. Wijnands, "Cisco Systems' Solution for Multicast in BGP/MPLS IP VPNs", RFC 6037, DOI 10.17487/RFC6037, October 2010, <https://www.rfc-editor.org/info/rfc6037>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014,

<https://www.rfc-editor.org/info/rfc7149>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <https://www.rfc-editor.org/info/rfc7297>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <https://www.rfc-editor.org/info/rfc7880>.

[RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <https://www.rfc-editor.org/info/rfc8077>.

[RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <https://www.rfc-editor.org/info/rfc8277>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <https://www.rfc-editor.org/info/rfc8453>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

[RFC8633] Reilly, D., Stenn, H., and D. Sibold, "Network Time Protocol Best Current Practices", BCP 223, RFC 8633, DOI 10.17487/RFC8633, July 2019, <https://www.rfc-editor.org/info/rfc8633>.

[RFC8695] Liu, X., Sarda, P., and V. Choudhary, "A YANG Data Model for the Routing Information Protocol (RIP)", RFC 8695, DOI 10.17487/RFC8695, February 2020, <https://www.rfc-editor.org/info/rfc8695>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/info/rfc8915>.

[RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <https://www.rfc-editor.org/info/rfc8969>.

[RFC9136] Rabadan, J., Ed., Henderickx, W., Drake, J., Lin, W., and A. Sajassi, "IP Prefix Advertisement in Ethernet VPN (EVPN)", RFC 9136, DOI 10.17487/RFC9136, October 2021, <https://www.rfc-editor.org/info/rfc9136>.

[YANG-Composed-VPN] Even, R., Wu, B., Wu, Q., and Y. Cheng, "YANG Data Model for Composed VPN Service Delivery", Work in Progress, Internet-Draft, draft-evenwu-opsawg-yang-composed-vpn-03, 8 March 2019, <https://datatracker.ietf.org/doc/html/draft-evenwu-opsawg-yang-composed-vpn-03>.

[YANG-SAPs] Gonzalez de Dios, O., Barguil, S., Wu, Q., Boucadair, M., and V. Lopez, "A Network YANG Model for Service Attachment Points", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-00, 25 January 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-sap-00>.

Appendix A. L3VPN Examples

A.1. 4G VPN Provisioning Example

L3VPNs are widely used to deploy 3G/4G, fixed, and enterprise services, mainly because several traffic discrimination policies can be applied within the network to deliver to the mobile customers a service that meets the SLA requirements.

Typically, and as shown in Figure 31, an eNodeB (CE) is directly connected to the access routers of the mobile backhaul and their logical interfaces (one or many, according to the service type) are configured in a VPN that transports the packets to the mobile core platforms. In this example, a 'vpn-node' is created with two 'vpn-network-accesses'.

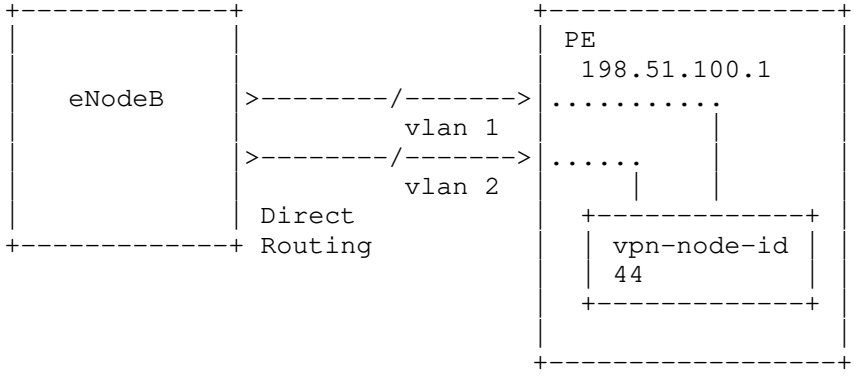


Figure 31: Mobile Backhaul Example

To create an L3VPN service using the L3NM, the following steps can be followed.

First, create the 4G VPN service (Figure 32).

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/vpn-services
Host: example.com
Content-Type: application/yang-data+json

{
 "ietf-l3vpn-ntw:vpn-services": {
   "vpn-service": [
     {
       "vpn-id": "4G",
       "vpn-description": "VPN to deploy 4G services",
       "customer-name": "mycustomer",
       "vpn-service-topology": "custom",
       "vpn-instance-profiles": {
         "vpn-instance-profile": [
           {
             "profile-id": "simple-profile",
             "local-as": 65550,
             "rd": "0:65550:1",
             "address-family": [
               {
                 "address-family": "ietf-vpn-common:dual-stack",
                 "vpn-targets": {
                   "vpn-target": [
                     {
                       "id": 1,
                       "route-targets": [
                         {
                           "route-target": "0:65550:1"
                         }
                       ],
                       "route-target-type": "both"
                     }
                   ]
                 }
               }
             ]
           }
         ]
       }
     }
   ]
 }
}
```

Figure 32: Create VPN Service

Second, create a VPN node, as depicted in Figure 33. In this type of service, the VPN node is equivalent to VRF configured in the physical device ('ne-id'=198.51.100.1). NOTE: '\' line wrapping in Figures 33 and 34 is implemented per [RFC8792].

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-nodes": {
    "vpn-node": [
      {
        "vpn-node-id": "44",
        "ne-id": "198.51.100.1",
        "active-vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
```

```
              "profile-id": "simple-profile"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 33: Create VPN Node

Finally, two VPN network accesses are created using the same physical port ('interface-id'=1/1/1). Each 'vpn-network-access' has a particular VLAN interface (1,2): "SYNC" and "DATA" (Figure 34). These interfaces differentiate the traffic between them.

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G/vpn-nodes/vpn-node=44
content-type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1",
        "description": "Interface SYNC to eNODE-B",
        "vpn-network-access-type": "ietf-vpn-common:point-to-point",
        "vpn-instance-profile": "simple-profile",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        },
        "connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address-allocation-type": "static-address",
            "static-addresses": {
              "primary-address": "1",
              "address": [
                {
                  "address-id": "1",
                  "customer-address": "192.0.2.2"
                }
              ]
            }
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address-allocation-type": "static-address",
            "primary-address": "1",
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
```

```
            "routing-protocol": [
              {
                "id": "1",
                "type": "ietf-vpn-common:direct"
              }
            ]
          }
        },
        {
          "id": "1/1/1.2",
          "interface-id": "1/1/1",
          "description": "Interface DATA to eNODE-B",
          "vpn-network-access-type": "ietf-vpn-common:point-to-point",
          "vpn-instance-profile": "simple-profile",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          },
          "connection": {
            "encapsulation": {
              "type": "ietf-vpn-common:dot1q",
              "dot1q": {
                "cvlan-id": 2
              }
            }
          },
          "ip-connection": {
            "ipv4": {
              "local-address": "192.0.2.1",
              "prefix-length": 30,
              "address-allocation-type": "static-address",
              "static-addresses": {
                "primary-address": "1",
                "address": [
                  {
                    "address-id": "1",
                    "customer-address": "192.0.2.2"
                  }
                ]
              }
            },
            "ipv6": {
              "local-address": "2001:db8::1",
              "prefix-length": 64,
              "address-allocation-type": "static-address",
              "primary-address": "1",
              "address": [
                {
                  "address-id": "1",
                  "customer-address": "2001:db8::2"
                }
              ]
            }
          },
          "routing-protocols": {
            "routing-protocol": [
              {
                "id": "1",
                "type": "ietf-vpn-common:direct"
              }
            ]
          }
        }
      ]
    }
  }
```

Figure 34: Create VPN Network Access

A.2. Loopback Interface

An example of a loopback interface is depicted in Figure 35.


```
{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "vpn-access-loopback",
        "interface-id": "Loopback1",
        "description": "An example of a loopback interface.",
        "vpn-network-access-type": "ietf-vpn-common:loopback",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        },
        "ip-connection": {
          "ipv6": {
            "local-address": "2001:db8::4",
            "prefix-length": 128
          }
        }
      }
    ]
  }
}
```


Figure 35: VPN Network Access with a Loopback Interface (Message Body)

A.3. Overriding VPN Instance Profile Parameters

Figure 36 shows a simplified example to illustrate how some information that is provided at the VPN service level (particularly as part of the 'vpn-instance-profiles') can be overridden by information configured at the VPN node level. In this example, PE3 and PE4 inherit the 'vpn-instance-profiles' parameters that are specified at the VPN service level, but PE1 and PE2 are provided with "maximum-routes" values at the VPN node level that override the values that are specified at the VPN service level.


```
{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "override-example",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "HUB",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 64510,
              "rd-suffix": 1001,
              "address-family": [
                {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "maximum-routes": [
                    {
                      "protocol": "ietf-vpn-common:any",
                      "maximum-routes": 100
                    }
                  ]
                }
              ]
            },
            {
              "profile-id": "SPOKE",
              "role": "ietf-vpn-common:spoke-role",
              "local-as": 64510,
              "address-family": [
```

```
        {
          "address-family": "ietf-vpn-common:dual-stack",
          "maximum-routes": [
            {
              "protocol": "ietf-vpn-common:any",
              "maximum-routes": 1000
            }
          ]
        }
      ]
    }
  ]
},
"vpn-nodes": {
  "vpn-node": [
    {
      "vpn-node-id": "PE1",
      "ne-id": "pe1",
      "router-id": "198.51.100.1",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "HUB",
            "rd": "1:198.51.100.1:1001",
            "address-family": [
              {
                "address-family":
                          "ietf-vpn-common:dual-stack",
                "maximum-routes": [
                  {
                    "protocol": "ietf-vpn-common:any",
                    "maximum-routes": 10
                  }
                ]
              }
            ]
          }
        ]
      }
    },
    {
      "vpn-node-id": "PE2",
      "ne-id": "pe2",
      "router-id": "198.51.100.2",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "SPOKE",
            "address-family": [
              {
                "address-family":
                          "ietf-vpn-common:dual-stack",
                "maximum-routes": [
                  {
                    "protocol": "ietf-vpn-common:any",
                    "maximum-routes": 100
                  }
                ]
              }
            ]
          }
        ]
      }
    },
    {
      "vpn-node-id": "PE3",
      "ne-id": "pe3",
      "router-id": "198.51.100.3",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
```

```
                    "profile-id": "SPOKE"
                  }
                ]
              }
            },
            {
              "vpn-node-id": "PE4",
              "ne-id": "pe4",
              "router-id": "198.51.100.4",
              "active-vpn-instance-profiles": {
                "vpn-instance-profile": [
                  {
                    "profile-id": "SPOKE"
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 36: VPN Instance Profile Example (Message Body)

A.4. Multicast VPN Provisioning Example

IPTV is mainly distributed through multicast over the LANs. In the following example, PIM - Sparse Mode (PIM-SM) is enabled and functional between the PE and the CE. The PE receives multicast traffic from a CE that is directly connected to the multicast source. The signaling between the PE and the CE is achieved using BGP. Also, the RP is statically configured for a multicast group.

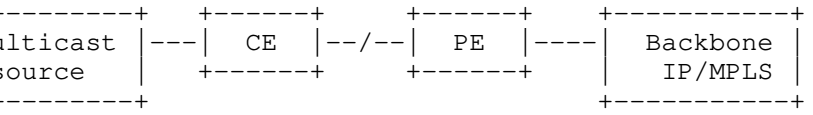


Figure 37: Multicast L3VPN Service Example

Figure 38 illustrates how to configure a multicast L3VPN service using the L3NM.

First, the multicast service is created together with a generic VPN instance profile (see the excerpt of the request message body shown in Figure 38).

```
{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "Multicast-IPTV",
        "vpn-description": "Multicast IPTV VPN service",
        "customer-name": "a-name",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "multicast",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 65536,
              "multicast": {
                "rp": {
                  "rp-group-mappings": {
                    "rp-group-mapping": [
                      {
                        "id": 1,
                        "rp-address": "203.0.113.17",
                        "groups": {
```

```
                          "group": [
                            {
                              "id": 1,
                              "group-address": "239.130.0.0/15"
                            }
                          ]
                        }
                      }
                    ]
                  },
                  "rp-discovery": {
                    "rp-discovery-type": "ietf-vpn-common:static-rp"
                  }
                }
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 38: Create Multicast VPN Service (Excerpt of the Message Request Body)

Then, the VPN nodes are created (see the excerpt of the request message body shown in Figure 39). In this example, the VPN node will represent VRF configured in the physical device.

```
{
  "ietf-l3vpn-ntw:vpn-node": [
    {
      "vpn-node-id": "500003105",
      "description": "VRF-IPTV-MULTICAST",
      "ne-id": "198.51.100.10",
      "router-id": "198.51.100.10",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "multicast",
            "rd": "65536:31050202"
          }
        ]
      }
    }
  ]
}
```

Figure 39: Create Multicast VPN Node (Excerpt of the Message Request Body)

Finally, create the VPN network access with multicast enabled (see the excerpt of the request message body shown in Figure 40).

```
{
  "ietf-l3vpn-ntw:vpn-network-access": {
    "id": "1/1/1",
    "description": "Connected-to-source",
    "vpn-network-access-type": "ietf-vpn-common:point-to-point",
    "vpn-instance-profile": "multicast",
    "status": {
      "admin-status": {
        "status": "ietf-vpn-common:admin-up"
      },
      "ip-connection": {
        "ipv4": {
          "local-address": "203.0.113.1",
          "prefix-length": 30,
          "address-allocation-type": "static-address",
          "static-addresses": {
```

```
              "primary-address": "1",
              "address": [
                {
                  "address-id": "1",
                  "customer-address": "203.0.113.2"
                }
              ]
            }
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "description": "Connected to CE",
                "peer-as": "65537",
                "address-family": "ietf-vpn-common:ipv4",
                "neighbor": "203.0.113.2"
              }
            }
          ]
        },
        "service": {
          "pe-to-ce-bandwidth": "100000000",
          "ce-to-pe-bandwidth": "100000000",
          "mtu": 1500,
          "multicast": {
            "access-type": "source-only",
            "address-family": "ietf-vpn-common:ipv4",
            "protocol-type": "router",
            "pim": {
              "hello-interval": 30,
              "status": {
                "admin-status": {
                  "status": "ietf-vpn-common:admin-up"
                }
              }
            }
          }
        }
      }
    }
  }
```


Figure 40: Create VPN Network Access (Excerpt of the Message Request Body)

Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from (listed alphabetically) Raul Arco, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Greg Mirsky, and Tom Petch. Many thanks to them. Thanks to Philip Eardley for the review of an early draft version of the document.

Daniel King, Daniel Voyer, Luay Jalil, and Stephane Litkowski contributed to early draft versions of this document. Many thanks to Robert Wilton for the AD review. Thanks to Andrew Malis for the routing directorate review, Rifaat Shekh-Yusef for the security directorate review, Qin Wu for the opsdir review, and Pete Resnick for the genart directorate review. Thanks to Michael Scharf for the discussion on the TCP-AO. Thanks to Martin Duke, Lars Eggert, Zaheduzzaman Sarker, Roman Danyliw, Erik Kline, Benjamin Kaduk, Francesca Palombini, and Ã\211ric Vyncke for the IESG review.

This work was supported in part by the European Commission-funded H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727) and Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows

(Teraflow) project (G.A. 101015857).

Contributors

Victor Lopez
Nokia
Madrid
Spain

Email: victor.lopez@nokia.com

Qin Wu
Huawei

Email: bill.wu@huawei.com

Manuel Lopez
Vodafone
Spain

Email: manuel-julian.lopez@vodafone.com

Lucia Oliva Ballega
Telefonica

Email: lucia.olivaballega.ext@telefonica.com

Erez Segev
Ribbon Communications

Email: erez.segev@rbbn.com

Paul Sherratt
Gamma Telecom

Email: paul.sherratt@gamma.co.uk

Authors' Addresses

Samier Barguil
Telefonica
Madrid
Spain

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
Spain

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
35000 Rennes
France

Email: mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone

Spain

Email: luis-angel.munoz@vodafone.com

Alejandro Aguado
Nokia
Madrid
Spain

Email: alejandro.aguado_martin@nokia.com