OPSAWG S. Barguil
Internet-Draft O. Gonzalez de Dios, Ed.
Intended status: Standards Track Telefonica
Expires: August 26, 2021 M. Boucadair, Ed.
Orange
L. Munoz
Vodafone
A. Aguado
Nokia
February 22, 2021

A Layer 3 VPN Network YANG Model
draft-ietf-opsawg-l3sm-l3nm-06

Abstract

This document defines a L3VPN Network YANG Model (L3NM) that can be used for the provisioning of Layer 3 Virtual Private Network (VPN) services within a service provider network. The model provides a network-centric view of L3VPN services.

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 the communication between a service orchestrator and a network controller/orchestrator.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

- "This version of this YANG module is part of RFC XXXX;"
- "RFC XXXX: Layer 3 VPN Network Model";
- reference: RFC XXXX

Also, please update the "revision" date of the YANG module.

Table of Contents

1. Introduction

[RFC8299] defines a Layer 3 Virtual Private Network Service YANG data Model (L3SM) that can be used for communication between customers and network operators. Such model is focused 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 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 (or a network operator) and the network controller/orchestrator by allowing for more network-centric information to be included. It enables further capabilities, such as resource management or to serve 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 [I-D.ietf-opsawg-vpn-common].

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

The L3NM YANG module is initially built with a prune and extend approach, taking as a starting points the YANG module described in

[RFC8299]. Nevertheless, the L3NM is not defined as an augment to L3SM because a specific structure is required to meet network-oriented L3 needs.

Some of the information captured in the L3SM can be passed by the Orchestrator in the L3NM (e.g., customer) or be used to fed some of the L3NM attributes (e.g., actual forwarding policies). Some of the information captured in L3SM may be maintained locally within the Orchestrator; which is in charge of maintaining the correspondence between a customer view and its network instantiation. Likewise, some of the 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 the 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 an end-to-end VPN connectivity are addressed by complementary YANG modules, e.g., [I-D.evenwu-opsawg-yang-composed-vpn].

L3NM focuses on BGP Provider Edge (PE) based Layer 3 VPNs as described in [RFC4026][RFC4110][RFC4364] and Multicast VPNs as described in [RFC6037][RFC6513][RFC7988].

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" defined in Section 2.1 of [RFC8969].

The meaning of the symbols in the tree diagrams is defined in [RFC8340].

This document makes use of the following terms:

Layer 3 VPN Customer Service Model (L3SM): A YANG module that describes the service requirements of a 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 Service Network Model (L3NM): A YANG module that describes a VPN service in the service provider network. It contains information of 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 YANG module can be consumed by a service orchestrator to request a VPN service to a Network Controller.

Service orchestrator: A functional entity that interacts with the customer of a L3VPN. The service orchestrator interacts with the customer using the L3SM. The service orchestrator is responsible of the Customer Edge (CE) - Provider Edge (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 nontrollers. A network orchestrator can manage one or several network nontrollers.

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 that belong to a single VPN service. A VPN service involves one or more VPN nodes. As it is an abstraction, the network controller will take on how to implement a VPN node. For example, typically, in a BGP-based VPN, a VPN node could be mapped into a Virtual Routing and Forwarding (VRF).

VPN network access: An abstraction that represents the network interfaces that are associated to 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 L3SM and L3NM.

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.

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

3. Acronyms

The following acronyms are used in the document:

ACL	Access Control List
AS	Autonomous System
ASM	Any-Source Multicast
ASN	AS Number
BSR	Bootstrap Router
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
CE	Customer Edge
IGMP	nternet Group Management Protocol
L3VPN	Layer 3 Virtual Private Network
L3SM	L3VPN Service Model
L3NM	L3VPN Network Model
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	Rendez-vous 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 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 module that reflect network-related information (that is hidden to customers).

The intelligence for translating customer-facing information into network-centric one is implementation specific.

The terminology from [RFC8309] is introduced 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".

```
                                     +---------------+
                                     |   Customer    |
                                     +---------------+
                     Customer Service Model  |
                         e.g., l3vpn-svc     |
                                     +---------------+
                                     |    Service    |
                                     | Orchestration |
                                     +---------------+
                        Network Model        |
                          l3vpn-ntw          |
                                     +---------------+
                                     |   Network     |
                                     | Orchestration |
                                     +---------------+
               Network Configuration Model   |
                                 +-----------+-----------+
                                 |                       |
                        +---------------+      +---------------+
                        |    Domain     |      |     Domain    |
                        | Orchestration |      | Orchestration |
                        +---------------+      +---------------+
             Device         |        |                   |
             Configuration  |        |                   |
             Model          |        |                   |
                       +---------+   |                   |
                       | Config  |   |                   |
                       | Manager |   |                   |
                       +---------+   |                   |
                            |        |                   |
                            | NETCONF/CLI..................
                            |        |                   |
                     +-------------------------------------------------+
                                         Network
```

Figure 1: L3NM Reference Architecture

The customer may use a variety of means to request a service that may trigger the instantiation of a L3NM. The customer may use the L3SM or may rely upon 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) [RFC7297], an enhanced VPN (VPN+) service [I-D.ietf-teas-enhanced-vpn], an IETF network slice [I-D.ietf-teas-ietf-network-slice-definition], or Abstraction and Control of TE Networks (ACTN) [RFC8453].

Note also that both the L3SM and the L3NM may be used in the context of the ACTN architecture. Figure 2 shows the Customer Network

Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC).

```
                       +----------------------------------+
                       | Customer                         |
                       | +-----------------------------+  |
                       | |             CNC             |  |
                       | +-----------------------------+  |
                       +----:-----------------------:-----+
                            :                       :
                            : L3SM                  : L3SM
                            :                       :
                  +---------:---------+   +-------------------+
                  | MDSC    :         |   |       MDSC        |
                  | +---------------+ |   |     (parent)      |
                  | |    Service    | |   +-------------------+
                  | | Orchestration | |             :
                  | +---------------+ |             : L3NM
                  |         :         |             :
                  |         : L3NM    |   +-------------------+
                  |         :         |   |       MDSC        |
                  | +---------------+ |   |      (child)      |
                  | |    Network    | |   +-------------------+
                  | | Orchestration | |             :
                  | +---------------+ |             :
                  +---------:---------+             :
                            :                       :
                            : Network Configuration :
                            :                       :
               +------------:-------+     +---------:------------+
               | Domain      :       |     |         : Domain     |
               | Controller :       |     |         : Controller |
               |       +---------+  |     |    +---------+       |
               |       |   PNC   |  |     |    |   PNC   |       |
               |       +---------+  |     |    +---------+       |
               +------------:-------+     +---------:------------+
                            :                       :
                            : Device Configuration  :
                            :                       :
                       +--------+              +--------+
                       | Device |              | Device |
                       +--------+              +--------+
```

Figure 2: L3SM and L3NM in the Context of ACTN

5. Relation with other YANG Models

The "ietf-vpn-common" module [I-D.ietf-opsawg-vpn-common] 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 L3NM depends on 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 view is only visible within the service provider and is not exposed outside (to customers, for example). The following discusses how L3NM interfaces with other YANG modules:

L3SM: L3NM is not a customer service model.

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

The L3NM can be fed with inputs that are requested by customers, typically, relying upon a L3SM template. Concretely, some parts of the L3SM module can be directly mapped into 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 L3SM.

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

Network Topology Modules: A L3VPN involves nodes that are part of a topology managed by the service provider network. Such topology can be represented as using the network topology module in [RFC8345].

Device Modules: L3NM is not a device model.

Once a global VPN service is captured by means of 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 [I-D.ietf-idr-bgp-model].

* PIM [I-D.ietf-pim-yang].

* NAT management [RFC8512].

* QoS management [I-D.ietf-rtgwg-qos-model].

* ACLs [RFC8519].

How 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 to 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, and therefore, L3NM can be useful to automate 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) [RFC7149][RFC7426] layer and little manual intervention will be still required.

Also common addition and/or removal of sites of an existing customer VPN can benefit of using L3NM by creation of workflows that either prune or add nodes as required from the network data mode.

6.2. Multi-Domain Resource Management

The implementation of L3VPN services which span across 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, there are two resources that must be orchestrated and manage to avoid asymmetric (non-functional) configuration, or the usage of unavailable resources.

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 across multiple management systems, the task of allocating RTs has to be aligned across the domains, therefore, the service model must provide a way to specify RTs. In addition, route distinguishers (RDs) must also be synchronized to avoid collisions in RD allocation 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 L3VPN can be implemented using dual PIM MVPNs (also known as, Draft Rosen model) [RFC4364] or Multiprotocol BGP (MP-BGP)-based MVPNs [RFC6513][RFC6514]. Both methods are supported and equally effective, but the main difference is that MBGP-based MVPN does not require multicast configuration on the service provider network. MBGP MVPNs employ the intra-autonomous system BGP control plane and PIM sparse mode 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, [RFC4364] has limitations such as reduced options for transport, control plane scalability, availability, operational inconsistency, and the need of maintaining state in the backbone. Because of these limitations, MBGP MVPN is the architectural model that has been taken as the base for implementing multicast service in L3VPNs. In this scenario, BGP auto discovery is used to discover 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 P2MP 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-services' and 'vpn-profiles' (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. '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

7.2. VPN Profiles

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

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 to these profiles in order to ease 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). An external connectivity may be an access to the Internet or a 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 scheme(s) and setup that can be applied when building and offering a VPN service.

'qos-profile-identifier': A Quality of Service (QoS) profile refers to as 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 [RFC5880] policies 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, at 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.

```
+--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

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 'vpn-id' is only

meaningful locally within 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-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-updated?   yang:date-and-time
         |  +--ro oper-status
         |     +--ro status?         identityref
         |     +--ro last-updated?   yang:date-and-time
         +--rw ie-profiles
         |  ...
         +--rw underlay-transport
         |  +-- (type)?
         |     +--:(abstract)
         |     |  +-- transport-instance-id?   string
         |     +--:(protocol)
         |       +-- protocol*           identityref
         +--rw external-connectivity
         |                   {external-connectivity}
         |  +--rw (profile)?
         |     +--:(profile)
         |        +--rw profile-name?   leafref
         +--rw vpn-nodes
            ...
```

Figure 5: VPN Services Subtree Structure

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

'vpn-id': Is an identifier that is used to uniquely identify the L3VPN service within 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 [I-D.ietf-opsawg-vpn-common]. For the L3NM, this is typically set to BGP/MPLS L3VPN.

'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 profiles (Section 4.3.5 of [RFC4364]).

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

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

'ie-profiles': Defines reusable import/export policies 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 [I-D.ietf-opsawg-vpn-common].

'external-connectivity': Indicates whether/how external connectivity is provided to the VPN service. For example, a service provider may provide an external connectivity to a VPN customer (e.g., to a public cloud). Such service may involve tweaking both filtering and NAT rules (e.g., bind a Virtual Routing and Forwarding (VRF) interface with a NAT instance as discussed in Section 2.10 of [RFC8512]). These added value features may be bound to all or a subset of network accesses. Some of these added value features may be implemented in a PE or in other nodes than PEs (e.g., a P node or event 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': Is an abstraction that represents a set of policies applied to a network node and that belong 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, as this is a network data model, the information about customers sites is not required in the model. Such information is rather relevant in the L3SM. Whether that information is included in the L3NM, e.g., to populate the various 'description' data node is implementation specific.

More details are provided in Section 7.5.

7.4. Import/Export Profiles

The import and export profiles construct contains a list with information related with route targets and distinguishers (RTs and RDs), grouped and identified by 'ie-profile-id'. The identifier is then referenced in one or multiple 'vpn-nodes' (Section 7.5) so that the controller can identify RTs and RDs to be configured for a given VRF. The subtree of 'ie-profiles' is shown in Figure 6.

The following modes are supported in:

'full-autoasigned': The network controller auto-assigns logical resources (RTs, RDs). This can apply for the deployment of new services.

'rd-from-pool': A variant of the previous one is to indicate a pool from where the RD values can be auto-assigned.

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

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

```
     +--rw l3vpn-ntw
        +--rw vpn-profiles
        |  ...
        +--rw vpn-services
           +--rw vpn-service* [vpn-id]
              +--rw vpn-id                  vpn-common:vpn-id
              +  ...
              +--rw ie-profiles
              |  +--rw ie-profile* [ie-profile-id]
              |     +--rw ie-profile-id         string
              |     +--rw (rd-choice)?
              |     |  +--:(directly-assigned)
              |     |  |  +--rw rd?
              |     |  |          rt-types:route-distinguisher
              |     |  +--:(pool-assigned)
              |     |  |  +--rw rd-pool-name?   string
              |     |  |  +--ro rd-from-pool?
              |     |  |          rt-types:route-distinguisher
              |     |  +--:(full-autoasigned)
              |     |  |  +--rw auto?           empty
              |     |  |  +--ro rd-auto?
              |     |  |          rt-types:route-distinguisher
              |     |  +--:(no-rd)
              |     |     +--rw no-rd?          empty
              |     +--rw vpn-targets
              |        +--rw vpn-target* [id]
              |        |  +--rw id                   int8
              |        |  +--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 vpn-nodes
                 +--rw vpn-node* [ne-id]
                    +--rw ne-id                      string
                    ...
                    +--rw node-ie-profile?           leafref
                    ...
```

Figure 6: Subtree Structure of Import/Export Profiles

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 belong 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 into a VRF in the node. However, the model also allows to point to an abstract node. In this case, the network controller will decide how to split the 'vpn-node' into VRFs.

```
+--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                union
               +--rw description?               string
               +--rw ne-id?                     string
               +--rw node-role?                 identityref
               +--rw local-autonomous-system?   inet:as-number
               |       {vpn-common:rtg-bgp}?
               +--rw address-family?            identityref
               +--rw router-id?                 inet:ip-address
               +--rw (rd-choice)?
               |  +--:(directly-assigned)
               |  |  +--rw rd?
               |  |          rt-types:route-distinguisher
               |  +--:(pool-assigned)
               |  |  +--rw rd-pool-name?        string
               |  |  +--ro rd-from-pool?
               |  |          rt-types:route-distinguisher
               |  +--:(full-autoasigned)
               |  |  +--rw auto?                empty
               |  |  +--ro rd-auto?
               |  |          rt-types:route-distinguisher
               |  +--:(no-rd)
               |     +--rw no-rd?               empty
               +--rw vpn-targets
               |  +--rw vpn-target* [id]
               |  |  +--rw id                   int8
               |  |  +--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 node-ie-profile?           leafref
```

```
               +--rw maximum-routes
               |  +--rw selector* [address-family protocol]
               |     +--rw address-family    identityref
               |     +--rw protocol          identityref
               |     +--rw maximum-routes?   uint32
               +--rw groups
               |  +--rw group* [group-id]
               |     +--rw group-id    string
               +--rw multicast {vpn-common:multicast}?
               |  ...
               +--rw status
               |  +--rw admin-status
               |  |  +--rw status?         identityref
               |  |  +--rw last-updated?   yang:date-and-time
               |  +--ro oper-status
               |     +--ro status?         identityref
               |     +--ro last-updated?   yang:date-and-time
               +--rw vpn-network-accesses
                  ...
```

Figure 7: VPN Node Subtree Structure

In reference to the subtree shown in Figure 7, the description of VPN node data nodes is as follows:

'vpn-node-id': Is an identifier that uniquely identifies a node that enable a VPN network access.

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

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

'node-role': Indicates the role of the VPN node in the VPN. Roles values are defines defined in [I-D.ietf-opsawg-vpn-common] (e.g., any-to-any-role, spoke-role, hub-role).

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

'address-family': Is used to identify the address family used for the Router ID. It can be set to IPv4 or IPv6.

'router-id': Indicates a unique Router ID information. It can be an IPv4 or IPv6 address as a function of the enclosed address-family.

'rd': If the logical resources are managed outside the network controller, the model allows to explicitly indicate the logical resources such as RTs and RDs.

As defined in [I-D.ietf-opsawg-vpn-common] and recalled in Section 7.4, RDs can be explicitly configured or automatically assigned. RD auto- assignment can also constrained by indicating an RD pool name ('rd- pool-name').

'vpn-targets': Specifies RT import/export rules for the VPN service.

'node-ie-profile': Refer to Section 7.4.

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

'groups': Lists the groups to which a VPN node belongs to [I-D.ietf-opsawg-vpn-common]. The 'group-id' is used to associate, e.g., redundancy or protection constraints with VPN nodes.

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

'status': Tracks the status of a node involved in a VPN service. Both operational 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 in L3SM, the L3NM does not need to model the customer site, only the points where the traffic from the site are received (i.e., the PE side of 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 Access

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 port-id?                    vpn-common:vpn-id
            +--rw description?                string
            +--rw vpn-network-access-type?   identityref
            +--rw status
            |  +--rw admin-status
            |  |  +--rw status?         identityref
            |  |  +--rw last-updated?   yang:date-and-time
            |  +--ro oper-status
            |     +--ro status?         identityref
            |     +--ro last-updated?   yang:date-and-time
            +--rw connection
            |  ...
            +--rw ip-connection
            |  ...
            +--rw oam
            |  ...
            +--rw security
            |  ...
            +--rw routing-protocols
            |  ...
            +--rw service
               ...
```

Figure 8: VPN Network Access Subtree Structure

In reference to the subtree depicted in Figure 8, a 'vpn-network-access' includes the following data nodes:

'id': Is an identifier of the VPN network access.

'port-id': Indicates the physical port on which the VPN network access is bound.

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

'vpn-network-access-type': Is used to select the type of network interface to be deployed in the devices. The available options are:

Point-to-Point: Represents a direct connection between the end-points. It implies that 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 broadcast connection between the end-points. It implies that the controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.

Pseudowire: Represents a connection coming from an L2VPN service. It implies that 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 to illustrate how loopback interfaces can be created is provided in Figure 35.

'status': Indicates both operational 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 the IP addressing information of a VPN network access. See Section 7.6.2.

'routing-protocols': 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 7.6.3.

'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 definition of a L3VPN is commonly specified not only at the IP layer, but also requires to provide parameters at the Ethernet layer, such as specifying an encapsulation type (e.g., VLAN, QinQ, QinAny,

VxLAN, etc.). The L3NM uses the 'connection' container to specify such parameters.

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 two connections 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 to a network element and a port. Hence, a bearer is just a bearer-reference to allow the translation between a service request (e.g., L3SM) and L3NM.

As shown in Figure 9, the 'connection' container defines protocols and parameters to enable connectivity at Layer 2.

```
             ...
             +--rw connection
             |  +--rw encapsulation-type?   identityref
             |  +--rw logical-interface
             |  |  +--rw peer-reference?   uint32
             |  +--rw tagged-interface
             |  |  +--rw type?                identityref
             |  |  +--rw dot1q-vlan-tagged {vpn-common:dot1q}?
             |  |  |  +--rw tag-type?   identityref
             |  |  |  +--rw cvlan-id?   uint16
             |  |  +--rw priority-tagged
             |  |  |  +--rw tag-type?   identityref
             |  |  +--rw qinq {vpn-common:qinq}?
             |  |  |  +--rw tag-type?   identityref
             |  |  |  +--rw svlan-id    uint16
             |  |  |  +--rw cvlan-id    uint16
             |  |  +--rw qinany {vpn-common:qinany}?
             |  |  |  +--rw tag-type?   identityref
             |  |  |  +--rw svlan-id    uint16
             |  |  +--rw vxlan {vpn-common:vxlan}?
             |  |     +--rw vni-id       uint32
             |  |     +--rw peer-mode?   identityref
             |  |     +--rw peer-list* [peer-ip]
             |  |        +--rw peer-ip    inet:ip-address
             |  +--rw bearer
             |     +--rw bearer-reference?   string
             |     |       {vpn-common:bearer-reference}?
             |     +--rw pseudowire
             |     |  +--rw vcid?      uint32
             |     |  +--rw far-end?   union
             |     +--rw vpls
             |        +--rw vcid?      union
             |        +--rw far-end?   union
             ...
```

Figure 9: Connection Subtree Structure

7.6.2. IP Connections

This container is used to group the IP addressing information of a VPN network access. The allocated address represents the PE interface address configuration. As shown in Figure 10, this container can include IPv4, IPv6, or both information if dual-stack is enabled.

```
                 ...
                 +--rw vpn-network-accesses
                    +--rw vpn-network-access* [id]
                       ...
                       +--rw ip-connection
                       |  +--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. Only one mode is enabled for a given service. Note that for the IPv6 cases, SLAAC [RFC7527] can be used.

Figure 11 shows the structure of the dynamic IPv4 address assignment.

```
...
+--rw ip-connection
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  +--rw local-address?                          inet:ipv4-prefix
|  |  +--rw address-allocation-type?                identityref
|  |  +--rw (allocation-type)?
|  |     +--:(provider-dhcp)
|  |     |  +--rw dhcp-server-enable?               boolean
|  |     |  +--rw (address-assign)?
|  |     |     +--:(number)
|  |     |     |  +--rw number-of-dynamic-address?   uint16
|  |     |     +--:(explicit)
|  |     |        +--rw customer-addresses
|  |     |           +--rw address-group* [group-id]
|  |     |              +--rw group-id         string
|  |     |              +--rw start-address?   inet:ipv4-address
|  |     |              +--rw end-address?     inet:ipv4-address
|  |     +--:(dhcp-relay)
|  |     |  +--rw dhcp-relay-enable?                boolean
|  |     |  +--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.

```
...
+--rw ip-connection
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  ...
|  +--rw ipv6 {vpn-common:ipv6}?
|     +--rw local-address?                          inet:ipv6-prefix
|     +--rw address-allocation-type?                identityref
|     +--rw (allocation-type)?
|        +--:(provider-dhcp)
|        |  +--rw dhcp-server-enable?               boolean
|        |  +--rw (address-assign)?
|        |     +--:(number)
|        |     |  +--rw number-of-dynamic-address?   uint16
|        |     +--:(explicit)
|        |        +--rw customer-addresses
|        |           +--rw address-group* [group-id]
|        |              +--rw group-id         string
|        |              +--rw start-address?   inet:ipv6-address
|        |              +--rw end-address?     inet:ipv6-address
|        +--:(dhcp-relay)
|        |  +--rw dhcp-relay-enable?                boolean
|        |  +--rw customer-dhcp-servers
|        |     +--rw server-ip-address*   inet:ipv6-address
|        +--:(static-addresses)
|           ...
...
```

Figure 12: IP Connection Subtree Structure (IPv6)

In the case of the 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 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/prefix-id
   |           +--rw address* [address-id]
   |              +--rw prefix-id              string
   |              +--rw customer-prefix?     inet:ipv6-prefix
   ...
```

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 protocol is 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 {vpn-common:rtg-bgp}?
                    |     |  ...
                    |     +--rw ospf {vpn-common:rtg-ospf}?
                    |     |  ...
                    |     +--rw isis {vpn-common:rtg-isis}?
                    |     |  ...
                    |     +--rw rip {vpn-common:rtg-rip}?
                    |     |  ...
                    |     +--rw vrrp {vpn-common:rtg-vrrp}?
                    |        ...
                    +--rw security
                        ...
```

Figure 14: Routing Subtree Structure

Multiple routing instances can be defined; each uniquely identified by an 'id'. The type of a routing instance is indicated in 'type'. The values of this attributes are those defined in [I-D.ietf-opsawg-vpn-common] ('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 to decide about 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:

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

- o A single BGP process with multiple neighbors running.

o A combination thereof.

Routing configuration does not include low-level policies. Such policies are handed at the device configuration level. Local policies of a service provider (e.g., filtering) will be implemented as part of the device configuration; these are not captured in the L3NM, but the model allows to associate local profiles with routing instances ('routing-profiles').

The L3NM supports the configuration of one or more IPv4/IPv6 static routes. Since the same structure is used for both IPv4 and IPv6, it was considered to have one single container to group both static entries independently of their address family, but that design was abandoned to ease the mapping with the structure in [RFC8299]. 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., "myfavourite-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, an interface, 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.

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

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

```
       ...
       +--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-updated?   yang:date-and-time
       |     |     |     +--ro oper-status
       |     |     |        +--ro status?         identityref
       |     |     |        +--ro last-updated?   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-updated?   yang:date-and-time
       |     |           +--ro oper-status
       |     |              +--ro status?         identityref
       |     |              +--ro last-updated?   yang:date-and-time
       ...
```

Figure 15: Static Routing Subtree Structure

In addition, the L3NM supports the following CE-PE routing protocols:

BGP: The L3NM allows to configure a BGP neighbor, including a set for parameters that are pertinent to be tweaked at the network level for service customization purposes.

This container does not aim to include every BGP parameter; a comprehensive set of parameters belongs more to the BGP device model.

The following data nodes are captured in Figure 16. It is up to the implementation to derive the corresponding BGP device configuration:

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

'local-autonomous-system': Is set to the AS Number (ASN) to override a customer ASN if such feature is requested by the customer.

'peer-autonomous-system': Conveys the customer's ASN.

'address-family': Indicates the address-family of the peer. It can be set to IPv4, IPv6, or dual-stack.

'neighbor': Can indicate two neighbors (each for a given address-family) or one neighbor (if 'address-family' attribute is set to dual-stack). A list of IP address(es) of the BGP neighbors can be then 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., replace the ASN of the customer specified in the AS_PATH BGP attribute with the ASN identified in the 'local-autonomous-system' attribute.

'default-route': Controls whether default route(s) can be advertised to the peer.

'site-of-origin': 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 (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 that is used to indicate the Site of Origin for VRF information [RFC5701]. It is used to prevent routing loops.

'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 such limit is reached, the action indicated in 'action-violate' will be followed.

'warning-threshold':a warning
notification will be triggered'
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 are: send a warning message, discard extra paths from the peer, or restart the session.

'bgp-timers': Two timers can be captured in this container: (1) 'hold-time' which is the time interval that will be used for the HoldTimer (Section 4.2 of [RFC4271]) when establishing a BGP session. (2) 'keep-alive' which is the time interval for the KeepAlive timer between a PE and a BGP peer (Section 4.4 of [RFC4271]).

'security': The module adheres to the recommendations in Section 13.2 of [RFC4364] as it allows to enable TCP-AO [RFC5925] and accommodates the installed base that make use of MD5. In addition, the module includes a provision for the use of IPsec.

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

```
   ...
   +--rw routing-protocols
   |  +--rw routing-protocol* [id]
   |     ...
   |     +--rw bgp {vpn-common:rtg-bgp}?
   |     |  +--rw description?              string
   |     |  +--rw local-autonomous-system?   inet:as-number
   |     |  +--rw peer-autonomous-system     inet:as-number
   |     |  +--rw address-family?            identityref
   |     |  +--rw neighbor*                  inet:ip-address
   |     |  +--rw multihop?                  uint8
   |     |  +--rw as-override?               boolean
   |     |  +--rw default-route?             boolean
   |     |  +--rw site-of-origin?            rt-types:route-origin
   |     |  +--rw ipv6-site-of-origin?       rt-types:ipv6-route-origin
   |     |  +--rw bgp-max-prefix
   |     |  |  +--rw max-prefix?          uint32
   |     |  |  +--rw warning-threshold?   decimal64
   |     |  |  +--rw violate-action?      enumeration
   |     |  |  +--rw restart-interval?    uint16
   |     |  +--rw bgp-timers
   |     |  |  +--rw keep-alive?   uint16
   |     |  |  +--rw hold-time?    uint16
   |     |  +--rw security
   |     |  |  +--rw enable?            boolean
   |     |  |  +--rw keying-material
   |     |  |     +--rw (option)?
   |     |  |        +--:(tcp-ao)
   |     |  |        |  +--rw enable-tcp-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-updated?   yang:date-and-time
   |     |     +--ro oper-status
   |     |        +--ro status?         identityref
   |     |        +--ro last-updated?   yang:date-and-time
   ...
```

Figure 16: BGP Routing Subtree Structure

OSPF: OSPF can be configured to run as a routing protocol on the 'vpn-network-access' [RFC4577][RFC6565]. The following data nodes are captured in Figure 17:

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

When only the IPv4 address-family is requested, it will be up to the implementation to decide whether OSPFv2 [RFC2328] or OSPFv3 [RFC5340] is used.

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

'metric': Associates a metric with OSPF routes.

'sham-links': Is 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]).

'max-lsa': Sets the maximum number of LSAs that the OSPF instance will accept.

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

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

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw ospf {vpn-common:rtg-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
|     |  |     |       vpn-common:vpn-id
|     |  |     +--rw metric?        uint16
|     |  +--rw max-lsa?          uint32
|     |  +--rw security
|     |  |  +--rw enable?            boolean
|     |  |  +--rw keying-material
|     |  |     +--rw (option)?
|     |  |        +--:(md5)
|     |  |        |  +--rw md5-keychain?
|     |  |        |          kc:key-chain-ref
|     |  |        +--:(ipsec)
|     |  |           +--rw sa?  string
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-updated?  yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-updated?  yang:date-and-time
...
```

Figure 17: OPSF Routing Subtree Structure

IS-IS: The model (Figure 18) allows the user to configure IS-IS to run on the 'vpn-network-access' interface. 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, Level2, 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).

'security': Controls the authentication schemes to be enabled for the IS-IS instance.

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

```
...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     ...
|     +--rw isis {vpn-common:rtg-isis}?
|     |  +--rw address-family?   identityref
|     |  +--rw area-address      yang:dotted-quad
|     |  +--rw level?            identityref
|     |  +--rw metric?           uint16
|     |  +--rw mode?             enumeration
|     |  +--rw security
|     |  |  +--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-updated?  yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-updated?  yang:date-and-time
...
```

Figure 18: IS-IS Routing Subtree Structure

RIP: The module covers only a list of address-family and status as shown in Figure 19. The meaning of these data nodes is similar to the other routing protocols.

```
          ...
          +--rw routing-protocols
          |  +--rw routing-protocol* [id]
          |     ...
          |     +--rw rip {vpn-common:rtg-rip}?
          |     |  +--rw address-family*   identityref
          |     |  +--rw status
          |     |     +--rw admin-status
          |     |     |  +--rw status?        identityref
          |     |     |  +--rw last-updated?  yang:date-and-time
          |     |     +--ro oper-status
          |     |        +--ro status?        identityref
          |     |        +--ro last-updated?  yang:date-and-time
          ...
```

Figure 19: RIP Subtree Structure

VRRP: The model (Figure 20) allows to enable VRRP on the 'vpn-network-access' interface. 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': Is used to identify the VRRP group.

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

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

'ping-reply': Controls whether ping requests can be replied to.

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

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

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

Figure 20: VRRP Subtree Structure

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. The current data nodes can be specified:

holdtime': Is used to indicate the expected BFD holddown time. The value can be set by the customer or selected from a profile.

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

'status': Indicates the status of BFD.

```
...
+--rw oam
|  +--rw bfd {vpn-common:bfd}?
|     +--rw (holdtime)?
|     |  +--:(fixed)
|     |  |  +--rw fixed-value?    uint32
|     |  +--:(profile)
|     |  |  +--rw profile-name?   leafref
|     +--rw authentication!
|     |  +--rw key-chain?    key-chain:key-chain-ref
|     |  +--rw meticulous?   boolean
|     +--rw status
|        +--rw admin-status
|           |  +--rw status?         identityref
|           |  +--rw last-updated?   yang:date-and-time
|           +--ro oper-status
|              +--ro status?         identityref
|              +--ro last-updated?   yang:date-and-time
...
```

Figure 21: IP Connection Subtree Structure (OAM)

7.6.5. Security

The 'security' container specifies the authentication and the encryption to be applied for a given VPN network access traffic. As depicted in the subtree shown in Figure 22, the L3NM can be used to directly control the encryption to put in place (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?
                         |                   kc:key-chain-ref
                         +--rw service
                             ...
```

Figure 22: Security Subtree Structure

7.6.6. Services

The 'services' 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 input-bandwidth     uint64
                      +--rw output-bandwidth    uint64
                      +--rw mtu                 uint16
                      +--rw qos {vpn-common:qos}?
                      |  ...
                      +--rw carrierscarrier
                      |       {vpn-common:carrierscarrier}?
                      |  +--rw signalling-type?   enumeration
                      +--rw multicast {vpn-common:multicast}?
                         ...
```

Figure 23: Services Subtree Structure

The following data nodes are defined:

'input-bandwidth': Indicates the inbound bandwidth of the connection (i.e., download bandwidth from the SP to the site).

'output-bandwidth': Indicates the outbound bandwidth of the connection (i.e., upload bandwidth from the site to the SP).

'mtu': Indicates the MTU at service level. It can be the IP MTU or MPLS MTU, for example.

'qos': Is used to define a set of QoS policies to apply on a given connection (Figure 24).

```
...
+--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-profile
|     +--rw qos-profile* [profile]
|        +--rw profile      leafref
|        +--rw direction?   identityref
...
```

Figure 24: Services Subtree Structure

Classification can be based on many criteria such as:

Layer 3: As shown in Figure 26, 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 shown in Figure 26, TCP or UDP-related match crietria can be specified in the L3NM.

```
+--rw qos {vpn-common:qos}?
|  +--rw qos-classification-policy
|  |  +--rw rule* [id]
|  |     +--rw id           string
|  |     +--rw (match-type)?
|  |     |  +--:(match-flow)
|  |     |  |  +--rw (l3)?
|  |     |  |  |  ...
|  |     |  |  +--rw (l4)?
|  |     |  |     +--:(tcp)
|  |     |  |     |  +--rw tcp
|  |     |  |     |     +--rw sequence-number?          uint32
|  |     |  |     |     +--rw acknowledgement-number?   uint32
|  |     |  |     |     +--rw data-offset?              uint8
|  |     |  |     |     +--rw reserved?                 uint8
|  |     |  |     |     +--rw flags?                    bits
|  |     |  |     |     +--rw window-size?              uint16
|  |     |  |     |     +--rw urgent-pointer?           uint16
|  |     |  |     |     +--rw options?                  binary
|  |     |  |     |     +--rw (source-port)?
|  |     |  |     |     |  +--:(source-port-range-or-operator)
|  |     |  |     |     |     +--rw source-port-range-or-operator
|  |     |  |     |     |        +--rw (port-range-or-operator)?
|  |     |  |     |     |           +--:(range)
|  |     |  |     |     |           |  +--rw lower-port
|  |     |  |     |     |           |  |       inet:port-number
|  |     |  |     |     |           |  +--rw upper-port
|  |     |  |     |     |           |          inet:port-number
|  |     |  |     |     |           +--:(operator)
|  |     |  |     |     |              +--rw operator? operator
|  |     |  |     |     |              +--rw port
|  |     |  |     |     |                      inet:port-number
|  |     |  |     |     +--rw (destination-port)?
|  |     |  |     +--:(destination-port-range-or-operator)
|  |     |  |     |          +--rw destination-port-range-or-operator
|  |     |  |     |             +--rw (port-range-or-operator)?
|  |     |  |     |                +--:(range)
|  |     |  |     |                |  +--rw lower-port
|  |     |  |     |                |  |       inet:port-number
|  |     |  |     |                |  +--rw upper-port
|  |     |  |     |                |          inet:port-number
|  |     |  |     |                +--:(operator)
|  |     |  |     |                   +--rw operator? operator
|  |     |  |     |                   +--rw port
|  |     |  |     |                           inet:port-number
```

```
   |  |     |  |     +--:(udp)
   |  |     |  |        +--rw udp
   |  |     |  |           +--rw length?                    uint16
   |  |     |  |           +--rw (source-port)?
   |  |     |  |           |  +--:(source-port-range-or-operator)
   |  |     |  |           |     +--rw source-port-range-or-operator
   |  |     |  |           |        +--rw (port-range-or-operator)?
   |  |     |  |           |           +--:(range)
   |  |     |  |           |           |  +--rw lower-port
   |  |     |  |           |           |  |       inet:port-number
   |  |     |  |           |           |  +--rw upper-port
   |  |     |  |           |           |          inet:port-number
   |  |     |  |           |           +--:(operator)
   |  |     |  |           |              +--rw operator?  operator
   |  |     |  |           |              +--rw port
   |  |     |  |           |                      inet:port-number
   |  |     |  |           +--rw (destination-port)?
   |  |     |  |              +--:(destination-port-range-or-operator)
   |  |     |  |                +--rw destination-port-range-or-operator
   |  |     |  |                    +--rw (port-range-or-operator)?
   |  |     |  |                       +--:(range)
   |  |     |  |                       |  +--rw lower-port
   |  |     |  |                       |  |       inet:port-number
   |  |     |  |                       |  +--rw upper-port
   |  |     |  |                       |          inet:port-number
   |  |     |  |                       +--:(operator)
   |  |     |  |                          +--rw operator?   operator
   |  |     |  |                          +--rw port
   |  |     |  |                                  inet:port-number
   ...
```

Figure 26: QoS Subtree Structure (L4)

Application match: Relies upon application-specific classification.

'carrierscarrier': Groups a set of parameters that are used when CsC is enabled such the use of BGP for signalling purposes [RFC8277].

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

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) can be controlled at the VPN node level or at the VPN network access.

```
              ...
              +--rw vpn-nodes
                 +--rw vpn-node* [vpn-node-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 node level are shown in Figure 29. Disabling multicast at the VPN node level will have an effect to disable it also at the VPN network access level. For IGMP, MLD, and PIM, Global data nodes that are defined at the VPN node level are applicable to all VPN network accesses whose corresponding nodes are not provided at the VPN network access level.

```
...
+--rw vpn-nodes
   +--rw vpn-node* [vpn-node-id]
      ...
      +--rw multicast {vpn-common:multicast}?
      |  +--rw status
      |  |  +--rw admin-status
      |  |  |  +--rw status?         identityref
      |  |  |  +--rw last-updated?   yang:date-and-time
      |  |  +--ro oper-status
      |  |     +--ro status?         identityref
      |  |     +--ro last-updated?   yang:date-and-time
      |  +--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 msdp {msdp}?
         |  |  +--rw peer?            inet:ip-address
         |  |  +--rw local-address?   inet:ip-address
         |  |  +--rw status
         |  |     +--rw admin-status
         |  |     |  +--rw status?         identityref
         |  |     |  +--rw last-updated?   yang:date-and-time
         |  |     +--ro oper-status
         |  |        +--ro status?         identityref
         |  |        +--ro last-updated?   yang:date-and-time
         |  +--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 status
         |  |     +--rw admin-status
         |  |     |  +--rw status?         identityref
         |  |     |  +--rw last-updated?   yang:date-and-time
         |  |     +--ro oper-status
         |  |        +--ro status?         identityref
         |  |        +--ro last-updated?   yang:date-and-time
         |  +--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 status
      |  |     +--rw admin-status
      |  |     |  +--rw status?         identityref
      |  |     |  +--rw last-updated?   yang:date-and-time
      |  |     +--ro oper-status
      |  |        +--ro status?         identityref
      |  |        +--ro last-updated?   yang:date-and-time
      |  +--rw pim {vpn-common:pim}?
      |     +--rw hello-interval?   uint8
      |     +--rw dr-priority?      uint16
      |     +--rw status
      |        +--rw admin-status
      |        |  +--rw status?         identityref
      |        |  +--rw last-updated?   yang:date-and-time
      |        +--ro oper-status
      |           +--ro status?         identityref
      |           +--ro last-updated?   yang:date-and-time
                 ...
```

Figure 28: Multicast Subtree Structure (VPN Node Level)

Multicast-related data nodes at the VPN network access level are shown in Figure 29. Except the 'status' node, the value configured at the VPN network access level overrides the value configured for the corresponding data node at the VPN node level.

```
   ...
   +--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-updated?   yang:date-and-time
                 |     +--ro oper-status
                 |        +--ro status?         identityref
                 |        +--ro last-updated?   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-updated?   yang:date-and-time
                 |     +--ro oper-status
                 |        +--ro status?         identityref
                 |        +--ro last-updated?   yang:date-and-time
                 +--rw pim {vpn-common:pim}?
                    +--rw priority?         uint8
                    +--rw hello-interval?   uint8
                    +--rw dr-priority?      uint16
                    +--rw status
                       +--rw admin-status
                       |  +--rw status?         identityref
                       |  +--rw last-updated?   yang:date-and-time
                       +--ro oper-status
                          +--ro status?         identityref
                          +--ro last-updated?   yang:date-and-time
```

Figure 29: Multicast Subtree Structure (VPN Network Access Level)

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

When ASM is used, the model supports the configuration of rendez-vous points (RPs). RP discovery may be 'static', 'bsr-rp', or 'auto-rp'. When set to 'static', RP to multicast grouping mapping 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 otherwise no RP address is needed.

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

When a particular VPN using ASM requires a more optimal traffic delivery, 'optimal-traffic-delivery' can be set. When set to 'true', the implementation must use any mechanism to provide a more optimal traffic delivery for the customer. For example, anycast is one of the mechanisms to enhance RPs redundancy, resilience against failures, and to recover from failures quickly.

For redundancy purposes, Multicast Source Discovery Protocol (MSDP) [RFC3618] may be enabled and used to share the state about sources between multiple 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, which is useful in a domain that does not support multicast sources, but does support multicast transit.

8. L3NM YANG Module

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

```
<CODE BEGINS>  file "ietf-l3vpn-ntw@2021-02-19.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 UUUU: A Layer 2/3 VPN Common YANG Model";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Key Chain.";
```

```
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

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

     Editor:    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) 2021 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 Simplified BSD License set
     forth in Section 4 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 XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.";

  revision 2021-02-19 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A Layer 3 VPN Network YANG Model";
  }
```

```
  /* 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 (PE)-Customer Edge (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 {
    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 7527: IPv6 Stateless Address Autoconfiguration";
  }

  identity static-address {
    base address-allocation-type;
    description
      "The Provider-to-customer addressing is static.";
  }

  identity slaac {
    if-feature "vpn-common:ipv6";
```

```
    base address-allocation-type;
    description
      "Use IPv6 SLAAC.";
    reference
      "RFC 7527: IPv6 Stateless Address Autoconfiguration";
  }

  identity bearer-inf-type {
    description
      "Identity for the bearer interface type.";
  }

  identity port-id {
    base bearer-inf-type;
    description
      "Identity for the priority-tagged interface.";
  }

  identity lag-id {
    base bearer-inf-type;
    description
      "Identity for the lag-tagged interface.";
  }

  identity local-defined-next-hop {
    description
      "Defines a base identity type 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 blackhole 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.";
  }

  typedef predefined-next-hop {
    type identityref {
      base local-defined-next-hop;
```

```
    }
    description
      "Pre-defined next-hop designation for locally generated routes.";
  }

  /* Typedefs */

  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.";
  }

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

  container l3vpn-ntw {
    description
      "Main container for L3VPN services management.";
    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
        "Top-level 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 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 ie-profiles {
          description
            "Container for Import/Export profiles.";
          list ie-profile {
            key "ie-profile-id";
            description
              "List for Imort/Export profile.";
            leaf ie-profile-id {
              type string;
              description
                "IE profile id.";
            }
            uses vpn-common:rt-rd;
          }
        }
        container underlay-transport {
          description
            "Container for 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
              "List for VPN node.";
            leaf vpn-node-id {
              type union {
                type vpn-common:vpn-id;
                type uint32;
              }
              description
                "Type STRING or NUMBER identifier.";
            }
            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 node-role {
              type identityref {
                base vpn-common:role;
              }
              default "vpn-common:any-to-any-role";
              description
                "Role of the VPN node in the IP VPN.";
            }
            leaf local-autonomous-system {
              if-feature "vpn-common:rtg-bgp";
              type inet:as-number;
              description
                "Provider's AS number in case the customer requests BGP
                 routing.";
            }
            leaf address-family {
              type identityref {
                base vpn-common:address-family;
              }
```

```
            description
              "The address family used for router-id information.";
          }
          leaf router-id {
            type inet:ip-address;
            description
              "The router-id information can be an IPv4 or IPv6
               address.";
          }
          uses vpn-common:rt-rd;
          leaf node-ie-profile {
            type leafref {
              path "/l3vpn-ntw/vpn-services/vpn-service"
                 + "/ie-profiles/ie-profile/ie-profile-id";
            }
            description
              "Node's Import/Export profile.";
          }
          container maximum-routes {
            description
              "Defines maximum routes for the VRF.";
            list selector {
              key "address-family protocol";
              description
                "List of address families.";
              leaf address-family {
                type identityref {
                  base vpn-common:address-family;
                }
                description
                  "Indicates the address family (IPv4 or IPv6).";
              }
              leaf protocol {
                type identityref {
                  base vpn-common:routing-protocol-type;
                }
                description
                  "Indicates the routing protocol. 'any' value can
                   be used to identify a limit that will apply for
                   any active routing protocol.";
              }
              leaf maximum-routes {
                type uint32;
                description
                  "Indicates the maximum prefixes the VRF can accept
                   for this address family and protocol.";
              }
            }
```

```
            }
            uses vpn-common:vpn-components-group;
            container multicast {
              if-feature "vpn-common:multicast";
              description
                "Global multicast parameters.";
              uses vpn-common:service-status;
              leaf-list tree-flavor {
                type identityref {
                  base vpn-common:multicast-tree-type;
                }
                description
                  "Type of tree to be used.";
              }
              container rp {
                description
                  "RP parameters.";
                container rp-group-mappings {
                  description
                    "RP-to-group mappings 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 Rendezvous Point (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 true, a redundancy mechanism for the
                           RP is required.";
                      }
                      leaf optimal-traffic-delivery {
```

```
                type boolean;
                default "false";
                description
                  "If true, the SP must ensure that
                   traffic uses an optimal path.  An SP may
                   use Anycast RP or RP-tree-to-SPT
                   switchover architectures.";
              }
              container anycast {
                when "../rp-redundancy = 'true' and
                      ../optimal-traffic-delivery = 'true'" {
                  description
                    "Only applicable if RP redundancy is enabled
                     and delivery through optimal path is
                     activated.";
                }
                description
                  "PIM Anycast-RP parameters.";
                leaf local-address {
                  type inet:ip-address;
                  description
                    "IP local address for PIM RP. Usually, it
                     corresponds to router ID or primary
                     address";
                }
                leaf-list rp-set-address {
                  type inet:ip-address;
                  description
                    "Address 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
                   provider-managed.";
              }
              type inet:ip-address;
              mandatory true;
              description
                "Defines the address of the RP.
                 Used if the RP is customer-managed.";
            }
            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 discovery type is BSR-RP.";
                }
                description
                  "Container for List of Customer BSR candidate's
                   addresses.";
                leaf-list bsr-candidate-address {
                  type inet:ip-address;
                  description
                    "Specifies the address of candidate Bootstrap
                     Router (BSR).";
                }
              }
            }
          }
          container msdp {
            if-feature "msdp";
            description
              "Includes MSDP-related parameters.";
            leaf peer {
              type inet:ip-address;
              description
                "Indicates the IP address of the MSDP peer.";
            }
            leaf local-address {
              type inet:ip-address;
              description
                "Indicates the IP address of the local end.
                 This local address must be configured on
                 the node.";
            }
            uses vpn-common:service-status;
          }
          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 to the
                 IGMP session";
              leaf group-addr {
```

```
                  type rt-types:ipv4-multicast-group-address;
                  description
                    "Multicast group IPv4 addresss.";
                }
                leaf source-addr {
                  type rt-types:ipv4-multicast-source-address;
                  description
                    "Multicast source IPv4 addresss.";
                }
              }
              leaf max-groups {
                type uint32;
                description
                  "Indicates the maximum groups.";
              }
              leaf max-entries {
                type uint32;
                description
                  "Indicates the maximum IGMP entries.";
              }
              leaf version {
                type identityref {
                  base vpn-common:igmp-version;
                }
                default "vpn-common:igmpv2";
                description
                  "Version of the IGMP.";
              }
              uses vpn-common:service-status;
            }
            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 to the
                   MLD session";
                leaf group-addr {
                  type rt-types:ipv6-multicast-group-address;
                  description
                    "Multicast group IPv6 addresss.";
                }
                leaf source-addr {
                  type rt-types:ipv6-multicast-source-address;
                  description
                    "Multicast source IPv6 addresss.";
```

```
                  }
                }
                leaf max-groups {
                  type uint32;
                  description
                    "Indicates the maximum groups.";
                }
                leaf max-entries {
                  type uint32;
                  description
                    "Indicates the maximum MLD entries.";
                }
                leaf version {
                  type identityref {
                    base vpn-common:mld-version;
                  }
                  default "vpn-common:mldv2";
                  description
                    "Version of the MLD protocol.";
                }
                uses vpn-common:service-status;
              }
              container pim {
                if-feature "vpn-common:pim";
                description
                  "Only applies when protocol type is PIM.";
                leaf hello-interval {
                  type uint8;
                  units "seconds";
                  default "30";
                  description
                    "PIM hello-messages interval.";
                }
                leaf dr-priority {
                  type uint16;
                  description
                    "Value to increase or decrease the
                     chances of a given DR being elected.";
                }
                uses vpn-common:service-status;
              }
            }
            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 port-id {
                  type vpn-common:vpn-id;
                  description
                    "Identifier for the interface.";
                }
                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 or multipoint.";
                }
                uses vpn-common:service-status;
                container connection {
                  description
                    "Encapsulation types.";
                  leaf encapsulation-type {
                    type identityref {
                      base vpn-common:encapsulation-type;
                    }
                    default "vpn-common:untagged-int";
                    description
                      "Encapsulation type. By default, the encapsulation
                       type is set to 'untagged'.";
                  }
                  container logical-interface {
                    description
                      "Reference of a logical interface
                       type.";
                    leaf peer-reference {
                      type uint32;
                      description
                        "Specifies the associated logical peer
                         interface.";
```

```
                   }
                 }
                 container tagged-interface {
                   description
                     "Container for tagged interfaces.";
                   leaf type {
                     type identityref {
                       base vpn-common:encapsulation-type;
                     }
                     default "vpn-common:priority-tagged";
                     description
                       "Tagged interface type. By default, the type of
                        the tagged interface is 'priority-tagged'.";
                   }
                   container dot1q-vlan-tagged {
                     when "derived-from-or-self(../type, "
                        + "'vpn-common:dot1q')" {
                       description
                         "Only applies when the type of the
                          tagged interface is 'dot1q'.";
                     }
                     if-feature "vpn-common: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;
                       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'.";
                    }
                    if-feature "vpn-common:qinq";
                    description
                      "QinQ.";
                    leaf tag-type {
                      type identityref {
                        base vpn-common:tag-type;
                      }
                      default "vpn-common:c-s-vlan";
                      description
                        "Tag type. By default, the tag type is
                         'c-s-vlan'.";
                    }
                    leaf svlan-id {
                      type uint16;
                      mandatory true;
                      description
                        "SVLAN identifier.";
                    }
                    leaf cvlan-id {
                      type uint16;
                      mandatory true;
                      description
                        "CVLAN identifier.";
                    }
                  }
                  container qinany {
                    when "derived-from-or-self(../type, "
                       + "'vpn-common:qinany')" {
                      description
                        "Only applies when the type of the
                         tagged interface is 'qinany'.";
```

```
                      }
                      if-feature "vpn-common:qinany";
                      description
                        "Container for QinAny.";
                      leaf tag-type {
                        type identityref {
                          base vpn-common:tag-type;
                        }
                        default "vpn-common:s-vlan";
                        description
                          "Tag type. By default, the tag type is
                           's-vlan'.";
                      }
                      leaf svlan-id {
                        type uint16;
                        mandatory true;
                        description
                          "Service VLAN ID.";
                      }
                    }
                    container vxlan {
                      when "derived-from-or-self(../type, "
                         + "'vpn-common:vxlan')" {
                        description
                          "Only applies when the type of the
                           tagged interface is 'vxlan'.";
                      }
                      if-feature "vpn-common:vxlan";
                      description
                        "QinQ.";
                      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'.";
                      }
                      list peer-list {
                        key "peer-ip";
                        description
```

```
                "List of peer IP addresses.";
              leaf peer-ip {
                type inet:ip-address;
                description
                  "Peer IP address.";
              }
            }
          }
        }
        container bearer {
          description
            "Defines physical properties of a site
             attachment.";
          leaf bearer-reference {
            if-feature "vpn-common:bearer-reference";
            type string;
            description
              "This is an internal reference for the service
               provider.";
          }
          container pseudowire {
            description
              "Pseudowire termination parameters";
            leaf vcid {
              type uint32;
              description
                "Indicates a PW or VC identifier.";
            }
            leaf far-end {
              type union {
                type uint32;
                type inet:ip-address;
              }
              description
                "SDP/Far End/LDP neighbour reference.";
            }
          }
          container vpls {
            description
              "Pseudowire termination parameters";
            leaf vcid {
              type union {
                type uint32;
                type string;
              }
              description
                "VCID identifier, IRB/RVPPLs interface
                 supported using string format.";
```

```
                      }
                      leaf far-end {
                        type union {
                          type uint32;
                          type inet:ip-address;
                        }
                        description
                          "SDP/Far End/LDP Neighbour reference.";
                      }
                    }
                  }
                }
                container ip-connection {
                  description
                    "Defines connection parameters.";
                  container ipv4 {
                    if-feature "vpn-common:ipv4";
                    description
                      "IPv4-specific parameters.";
                    leaf local-address {
                      type inet:ipv4-prefix;
                      description
                        "This address is used at provider side.";
                    }
                    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 is not enabled.";
                    }
                    choice allocation-type {
                      description
                        "Choice of the IPv4 address allocation.";
                      case provider-dhcp {
                        when "derived-from-or-self(./address-"
                           + "allocation-type, 'provider-dhcp')" {
                          description
```

```
                                "Only applies when addresses are allocated
                                 by DHCP.";
                            }
                            description
                              "DHCP allocated addresses related
                               parameters.";
                            leaf dhcp-server-enable {
                              type boolean;
                              default "true";
                              description
                                "Enables a DHCP service on this access.
                                 The following information are passed to
                                 the provider's DHCP server.";
                            }
                            choice address-assign {
                              default "number";
                              description
                                "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-group {
                                    key "group-id";
                                    description
                                      "Describes IP addresses to be
                                       allocated by DHCP.

                                       When only start-address or only
                                       end-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.  If no address
```

```
                      is specified, it implies customer
                      addresses group is not supported.";
                   leaf group-id {
                     type string;
                     description
                       "Group-id for the address range from
                        start-address to end-address.";
                   }
                   leaf start-address {
                     type inet:ipv4-address;
                     description
                       "Indicates the first address in
                        the group.";
                   }
                   leaf end-address {
                     type inet:ipv4-address;
                     description
                       "Indicates the last address in the
                        group.";
                   }
                 }
               }
             }
           }
         }
         case dhcp-relay {
           when "derived-from-or-self(./address-allocation"
              + "-type, 'provider-dhcp-relay')" {
             description
               "Only applies when the provider is required
                to implement DHCP relay function.";
           }
           description
             "DHCP relay provided by operator.";
           leaf dhcp-relay-enable {
             type boolean;
             default "true";
             description
               "Enables the DHCP relay function for this
                access.";
           }
           container customer-dhcp-servers {
             description
               "Container for list of customer
                DHCP servers.";
             leaf-list server-ip-address {
               type inet:ipv4-address;
               description
```

```
                      "IP address of customer DHCP server.";
                  }
                }
              }
              case static-addresses {
                when "derived-from-or-self(./address-allocation"
                   + "-type, 'static-address')" {
                  description
                    "Only applies when address allocation
                     type is static.";
                }
                description
                  "Describes IPv4 addresses used.";
                leaf primary-address {
                  type leafref {
                    path "../address/address-id";
                  }
                  description
                    "Primary address of the connection.";
                }
                list address {
                  key "address-id";
                  description
                    "Describes IPv4 addresses used.";
                  leaf address-id {
                    type string;
                    description
                      "Used static IPv4 address.";
                  }
                  leaf customer-address {
                    type inet:ipv4-address;
                    description
                      "IPv4 address at the customer side.";
                  }
                }
              }
            }
          }
          container ipv6 {
            if-feature "vpn-common:ipv6";
            description
              "IPv6-specific parameters.";
            leaf local-address {
              type inet:ipv6-prefix;
              description
                "Address of the provider side.";
            }
            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 is
                         not enabled.";
                    }
                    choice allocation-type {
                      description
                        "IPv6 allocation type.";
                      case 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.";
                        }
                        description
                          "DHCPv6 allocated addresses related
                           parameters.";
                        leaf dhcp-server-enable {
                          type boolean;
                          default "true";
                          description
                            "Enables DHCPv6 service for this access.";
                        }
                        choice address-assign {
                          default "number";
                          description
                            "Choice for the way to assign IPv6
                             prefixes.";
                          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-group {
                             key "group-id";
                             description
                               "Describes IPv6 addresses allocated
                                by DHCPv6.

                                When only start-address or only
                                end-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.

                                If no address is specified, it
                                implies customer addresses group
                                is not supported.";
                             leaf group-id {
                               type string;
                               description
                                 "Group-id for the address range
                                  from identified by start-address
                                  and end-address.";
                             }
                             leaf start-address {
                               type inet:ipv6-address;
                               description
                                 "Indicates the first address.";
                             }
                             leaf end-address {
                               type inet:ipv6-address;
                               description
                                 "Indicates the last address.";
                             }
                           }
                         }
                       }
                     }
                   }
                   case dhcp-relay {
                     when "derived-from-or-self(./address-allo"
                        + "cation-type, 'provider-dhcp-relay')" {
                       description
                         "Only applies when the provider is required
                          to implement DHCP relay function.";
```

```
                        }
                        description
                          "DHCP relay provided by operator.";
                        leaf dhcp-relay-enable {
                          type boolean;
                          default "true";
                          description
                            "Enables the DHCP relay function for this
                             access.";
                        }
                        container customer-dhcp-servers {
                          description
                            "Container for list of customer DHCP
                             servers.";
                          leaf-list server-ip-address {
                            type inet:ipv6-address;
                            description
                              "This node contains the IP address of
                               the customer DHCP server.  If the DHCP
                               relay function is implemented by the
                               provider, this node contains the
                               configured value.";
                          }
                        }
                      }
                      case static-addresses {
                        when "derived-from-or-self(./address-allocation"
                           + "-type, 'static-address')" {
                          description
                            "Only applies when protocol allocation type
                             is static.";
                        }
                        description
                          "IPv6-specific parameters for static
                           allocation.";
                        leaf primary-address {
                          type leafref {
                            path "../address/prefix-id";
                          }
                          description
                            "Principal address of the connection";
                        }
                        list address {
                          key "prefix-id";
                          description
                            "Describes IPv6 prefixes used.";
                          leaf prefix-id {
                            type string;
```

```
                            description
                              "An identifier of an IPv6 prefix.";
                          }
                          leaf customer-prefix {
                            type inet:ipv6-prefix;
                            description
                              "An IPv6 prefix 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 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')" {
              description
                "Only applies when protocol is static.";
            }
            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, an interface,
                     or a pre-defined 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 of the static
                     routes.";
                }
                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, an interface, or a
                     pre-defined 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')" {
              description
                "Only applies when protocol is BGP.";
            }
            if-feature "vpn-common:rtg-bgp";
            description
              "BGP-specific configuration.";
            leaf description {
              type string;
              description
                "Includes a description of the BGP session.

                 Such description is meant to be used for
                 diagnosis purposes. The semantic of the
                 description is local to an
                 implementation.";
            }
            leaf local-autonomous-system {
              type inet:as-number;
              description
                "Is set to the ASN to override a peers' ASN
                 if such feature is requested by the
                 Customer.";
            }
            leaf peer-autonomous-system {
              type inet:as-number;
              mandatory true;
```

```
            description
              "Indicates the Customer's AS Number (ASN) in
               case 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-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 AS override is enabled,
               i.e., replace the ASN of the customer
               specified in the AS Path attribute with
               the local ASN.";
          }
          leaf default-route {
            type boolean;
            default "false";
            description
              "Defines whether default route(s) can be
               advertised to its peer. If set, the
               default route(s) is advertised to its
               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
                "RFC4364, 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
                "IPv6 Route Origins are IPv6 Address Specific
                 BGP Extended that are meant to the Site of
                 Origin for VRF information.";
              reference
                "RFC 5701: IPv6 Address Specific BGP Extended
                           Community Attribute";
            }
            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 to control how many prefixes
                   can be received from a neighbor.

                   If the limit is exceeded, the action
                   indicated in violate-action will be
                   followed.";
                reference
                  "RFC4271, 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
                    "Restarts after a time interval.";
                }
              }
              description
                "BGP neighbour max-prefix violate
                 action";
            }
            leaf restart-interval {
              type uint16;
              units "minutes";
              description
                "Time interval (min) 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 CE-PE routing
               protocol.";
```

```
            leaf keep-alive {
              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 KEEPALIVE
                 messages are disabled.

                 It is suggested that the maximum time
                 between  KEEPALIVEmessages would be
                 one third of the Hold Time interval.";
              reference
                "Section 4.4 of RFC 4271";
            }
            leaf hold-time {
              type uint16 {
                range "0 | 3..65535";
              }
              units "seconds";
              default "90";
              description
                "It 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
                "Section 4.2 of RFC 4271";
            }
          }
          container security {
            description
              "Container for BGP security 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 tcp-ao {
                              description
                                "Uses TCP-Authentication Option
                                (TCP-AO).";
                              reference
                                "RFC 5925: The TCP Authentication
                                           Option.";
                              leaf enable-tcp-ao {
                                type boolean;
                                description
                                  "Enables TCP-AO.";
                              }
                              leaf ao-keychain {
                                type key-chain:key-chain-ref;
                                description
                                  "Reference to the TCP-AO key chain.";
                                reference
                                  "RFC 8177: YANG Key Chain.";
                              }
                            }
                            case md5 {
                              description
                                "Uses MD5 to secure the session.";
                              reference
                                "Section 13.2 of RFC 4364";
                              leaf md5-keychain {
                                type key-chain:key-chain-ref;
                                description
                                  "Reference to the MD5 key chain.";
                                reference
                                  "RFC 8177: YANG Key Chain.";
                              }
                            }
                            case explicit {
                              leaf key-id {
                                type uint32;
                                description
                                  "Key Identifier";
                              }
```

```
                  leaf key {
                    type string;
                    description
                      "OSPF authentication key.";
                  }
                  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 IKE
                     Security Association (SA).";
                  leaf sa {
                    type string;
                    description
                      "Indicates the name of the SA.";
                  }
                }
              }
            }
          }
          uses vpn-common:service-status;
        }
        container ospf {
          when "derived-from-or-self(../type, "
             + "'vpn-common:ospf')" {
            description
              "Only applies when protocol is OSPF.";
          }
          if-feature "vpn-common:rtg-ospf";
          description
            "OSPF-specific configuration.";
          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.";
                }
                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.";
                  list sham-link {
                    key "target-site";
                    description
                      "Creates a sham link with another site.";
                    leaf target-site {
                      type vpn-common:vpn-id;
                      description
                        "Target site for the sham link connection.
                         The site is referred to by its ID.";
                    }
                    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.";
                    }
                  }
                }
                leaf max-lsa {
                  type uint32 {
                    range "1..4294967294";
                  }
                  description
                    "Maximum number of allowed LSAs OSPF.";
                }
                container security {
                  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
                                  "key-chain name.";
                              }
                            }
                            case auth-key-explicit {
                              leaf key-id {
                                type uint32;
                                description
                                  "Key Identifier";
                              }
                              leaf key {
                                type string;
                                description
                                  "OSPF authentication key.";
                              }
                              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 name of the SA.";
                              }
                            }
```

```
                           }
                         }
                       }
                       uses vpn-common:service-status;
                     }
                     container isis {
                       when "derived-from-or-self(../type, "
                          + "'vpn-common:isis')" {
                         description
                           "Only applies when protocol is IS-IS.";
                       }
                       if-feature "vpn-common:rtg-isis";
                       description
                         "IS-IS specific configuration.";
                       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 yang:dotted-quad;
                         mandatory true;
                         description
                           "Area address.";
                       }
                       leaf level {
                         type identityref {
                           base vpn-common:isis-level;
                         }
                         description
                           "Can be level1, level2, or level1-2.";
                       }
                       leaf metric {
                         type uint16;
                         default "1";
                         description
                           "Metric of the PE-CE link.  It is used
                            in the routing state calculation and
                            path selection.";
                       }
                       leaf mode {
                         type enumeration {
                           enum active {
                             description
                               "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 security {
                         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
                                   "key-chain name.";
                               }
                             }
                             case auth-key-explicit {
                               leaf key-id {
                                 type uint32;
                                 description
                                   "Key Identifier";
                               }
                               leaf key {
                                 type string;
                                 description
                                   "IS-IS authentication key.";
```

```
                    }
                    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')" {
              description
                "Only applies when the protocol is RIP.
                 For IPv4, the model assumes that RIP
                 version 2 is used.";
            }
            if-feature "vpn-common:rtg-rip";
            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.";
            }
            uses vpn-common:service-status;
          }
          container vrrp {
            when "derived-from-or-self(../type, "
               + "'vpn-common:vrrp')" {
              description
                "Only applies when protocol is VRRP.";
            }
            if-feature "vpn-common:rtg-vrrp";
            description
              "Configuration specific to VRRP.";
            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 priority {
                  type uint8 {
                    range "1..254";
                  }
                  default "100";
                  description
                    "Sets the local priority of the VRRP
                     speaker.";
                }
                leaf ping-reply {
                  type boolean;
                  description
                    "Controls whether the VRRP speaker should
                     answer 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.";
```

```
                    choice holdtime {
                      default "fixed";
                      description
                        "Choice for holdtime flavor.";
                      case fixed {
                        leaf fixed-value {
                          type uint32;
                          units "msec";
                          description
                            "Expected BFD holdtime.

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

                             If the provider doesn't allow the
                             customer to use this function,
                             the fixed-value will not be set.";
                        }
                      }
                      case profile {
                        description
                          "Well-known SP profile.";
                        leaf profile-name {
                          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
                          "Section 6.7 of RFC 5880";
                      }
                    }
                    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 true, traffic encryption on the
                         connection is required. It is
                         disabled, otherwise.";
                    }
                    leaf layer {
                      when "../enabled = 'true'" {
                        description
                          "Indicates the layer on which 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 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 on the attachment.";
                  leaf input-bandwidth {
                    type uint64;
                    units "bps";
                    mandatory true;
                    description
                      "From the customer site's perspective, the
                       service input bandwidth of the connection
                       or download bandwidth from the SP to
                       the site.";
```

```
                  }
                  leaf output-bandwidth {
                    type uint64;
                    units "bps";
                    mandatory true;
                    description
                      "From the customer site's perspective,
                       the service output bandwidth of the
                       connection or upload bandwidth from
                       the site to the SP.";
                  }
                  leaf mtu {
                    type uint16;
                    units "bytes";
                    mandatory true;
                    description
                      "MTU at service level.  If the service is IP,
                       it refers to the IP MTU.  If CsC is enabled,
                       the requested MTU will refer
                       to the MPLS MTU 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-profile {
                      description
                        "QoS profile configuration.";
                      list qos-profile {
                        key "profile";
                        description
                          "QoS profile.
                           Can be standard profile or 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 carrierscarrier {
          if-feature "vpn-common:carrierscarrier";
          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 MPLSservice using an
             IP VPN to carry its traffic).";
          leaf signalling-type {
            type enumeration {
              enum ldp {
                description
                  "Use LDP as the signalling protocol
                   between the PE and the CE.  In this
                   case, an IGP routing protocol must
                   also be activated.";
              }
              enum bgp {
                description
                  "Use BGP as the signalling 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 signalling type.";
          }
        }
        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
                  "When true, there is no PIM adjacency on
                   the interface.";
              }
              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 to
                     to IGMP session";
                  leaf group-addr {
                    type rt-types:ipv4-multicast-group-address;
                    description
                      "Multicast group IPv4 addresss.";
                  }
                  leaf source-addr {
                    type rt-types:ipv4-multicast-source-address;
                    description
                      "Multicast source IPv4 addresss.";
                  }
                }
                leaf max-groups {
                  type uint32;
                  description
                    "Indicates the maximum groups.";
                }
```

```
                      leaf max-entries {
                        type uint32;
                        description
                          "Indicates the maximum 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
                          "Version of the IGMP.";
                      }
                      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 to
                           the MLD session";
                        leaf group-addr {
                          type rt-types:ipv6-multicast-group-address;
                          description
                            "Multicast group IPv6 addresss.";
                        }
                        leaf source-addr {
                          type rt-types:ipv6-multicast-source-address;
                          description
                            "Multicast source IPv6 addresss.";
                        }
                      }
                      leaf max-groups {
                        type uint32;
                        description
                          "Indicates the maximum groups.";
                      }
```

```
                      leaf max-entries {
                        type uint32;
                        description
                          "Indicates the maximum 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
                          "Version of the MLD protocol.";
                      }
                      uses vpn-common:service-status;
                    }
                    container pim {
                      when "../protocol-type = 'router'";
                      if-feature "vpn-common:pim";
                      description
                        "Only applies when protocol type is PIM.";
                      leaf priority {
                        type uint8;
                        description
                          "PIM priority definition.";
                      }
                      leaf hello-interval {
                        type uint8;
                        units "seconds";
                        default "30";
                        description
                          "PIM hello-messages interval.";
                      }
                      leaf dr-priority {
                        type uint16;
                        description
                          "Value to increase or decrease the
                           chances of a given DR being elected.";
                      }
                      uses vpn-common:service-status;
                    }
                  }
                }
              }
```

```
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

9. IANA Considerations

This document requests IANA to register 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.
```

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

```
      name: ietf-l3vpn-ntw
      namespace: urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw
      maintained by IANA: N
      prefix: l3nm
      reference: RFC XXXX
```

10. Security Considerations

The YANG module specified in this document defines 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 [RFC8466].

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.

The "ietf-l3vpn-ntw" module is used to manage Layer 3 VPNs in a service provider backbone network. Hence, the module can be used to request, modify, or retrieve L3VPN services. For example, the creation of a 'vpn-service' leaf instance triggers the creation of an L3VPN Service in a service provider network.

Due to the foreseen use of the "ietf-l3vpn-ntw" module, there are a number of data nodes defined in the 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-service': 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), leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create a L3VPN Service or adding a new network access. Such activity can be detected by adequately monitoring and tracking network configuration changes.

Some of the readable data nodes in the "ietf-l3vpn-ntw" 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 as a violation of the customer-provider trust relationship.

The following summarizes the foreseen risks of using the "ietf-l3vpn-ntw" module can be classified into:

- Malicious clients attempting to delete or modify VPN services.

- Unauthorized clients attempting to create/modify/delete a VPN service.

- Unauthorized clients attempting to read VPN service related information.

11. Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from (listed alphabetically): Raul Arco, Miguel Cros Cecilia, Joe Clarke, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, and Julian Lucek. Many thanks

to them. Thanks to Philip Eardly for the review of an early version of the document.

Daniel King, Daniel Voyer, Luay Jalil, and Stephane Litkowski contributed to early version of the individual submission.

This work was supported in part by the European Commission funded H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727).

12. Contributors

Victor Lopez
Telefonica
Email: victor.lopezalvarez@telefonica.com

Qin Wu
Huawei
Email: bill.wu@huawei.com>

Manuel Julian
Vodafone
Email: manuel-julian.lopez@vodafone.com>

Lucia Oliva Ballega
Telefonica
Email: lucia.olivaballega.ext@telefonica.com>

Erez Segev
ECI Telecom
Email: erez.segev@ecitele.com>

13. References

13.1. Normative References

[I-D.ietf-opsawg-vpn-common]
barguil, s., Dios, O., Boucadair, M., and Q. WU, "A Layer 2/3 VPN Common YANG Model", draft-ietf-opsawg-vpn-common-03 (work in progress), January 2021.

[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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

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

[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>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[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>.

[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>.

[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>.

[RFC7988] Rosen, E., Ed., Subramanian, K., and Z. Zhang, "Ingress Replication Tunnels in Multicast VPN", RFC 7988, DOI 10.17487/RFC7988, October 2016, <https://www.rfc-editor.org/info/rfc7988>.

[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>.

[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>.

13.2. Informative References

[I-D.evenwu-opsawg-yang-composed-vpn]
Even, R., Bo, W., Wu, Q., and Y. Cheng, "YANG Data Model for Composed VPN Service Delivery", draft-evenwu-opsawg-yang-composed-vpn-03 (work in progress), March 2019.

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-10 (work in progress), November 2020.

[I-D.ietf-pim-yang]
Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG Data Model for Protocol Independent Multicast (PIM)", draft-ietf-pim-yang-17 (work in progress), May 2018.

[I-D.ietf-rtgwg-qos-model]
Choudhary, A., Jethanandani, M., Strahle, N., Aries, E., and I. Chen, "YANG Model for QoS", draft-ietf-rtgwg-qos-model-02 (work in progress), July 2020.

[I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Networks (VPN+) Service", draft-ietf-teas-enhanced-vpn-06 (work in progress), July 2020.

[I-D.ietf-teas-ietf-network-slice-definition]
Rokui, R., Homma, S., Makhijani, K., Contreras, L., and J. Tantsura, "Definition of IETF Network Slices", draft-ietf-teas-ietf-network-slice-definition-00 (work in progress), January 2021.

[PYANG] "pyang", November 2020, <https://github.com/mbj4668/pyang>.

[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>.

[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>.

[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>.

[RFC7527] Asati, R., Singh, H., Beebee, W., Pignataro, C., Dart, E., and W. George, "Enhanced Duplicate Address Detection", RFC 7527, DOI 10.17487/RFC7527, April 2015, <https://www.rfc-editor.org/info/rfc7527>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

[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>.

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.

As it is shown in the Figure 30, typically, 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'.

```
+-------------+                  +------------------+
|             |                  | PE               |
|             | 192.0.2.2        |  198.51.100.1    |
|   eNodeB    |>--------/------->|...........       |
|             |          Vlan 1  |          |       |
|             |>--------/------->|......    |       |
|             |          Vlan 2  |     |    |       |
|             | Direct           |  +-------------+ |
+-------------+ Routing          |  | vpn-node-id | |
                                 |  | 44          | |
                                 |  +-------------+ |
                                 |                  |
                                 +------------------+
```

Figure 30: Mobile Backhaul Example

To create a L3VPN service using the L3NM, the following sample steps can be followed:

First: Create the 4G VPN service (Figure 31).

```
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",
        "customer-name": "mycustomer",
        "vpn-service-topology": "custom",
        "description": "VPN to deploy 4G services"
      }
    ]
  }
}
```


Figure 31: Create VPN Service

Second: Create a VPN node as depicted in Figure 32. In this type of service, the VPN node is equivalent to the VRF configured in the physical device ('ne-id'=198.51.100.1).

```
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",
        "local-autonomous-system": "65550",
        "rd": "0:65550:1",
        "vpn-targets": {
          "vpn-target": [
            {
              "id": "1",
              "route-targets": [
                "0:65550:1"
              ],
              "route-target-type": "both"
            }
          ]
        }
      }
    ]
  }
}
```


Figure 32: Create VPN Node

Finally, two VPN network accesses are created using the same physical port ('port-id'=1/1/1). Each 'vpn-network-access' has a particular VLAN (1,2) to differentiate the traffic between: Sync and data (Figure 33).

```
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": [
      {
        "vpn-network-access-id": "1/1/1.1",
        "port-id": "1/1/1",
        "description": "Interface SYNC to eNODE-B",
        "admin-status": {
```

```
              "status": "vpn-common:administrative-state-up"
            },
            "vpn-network-access-type": "vpn-common:point-to-point",
            "ip-connection": {
              "ipv4": {
                "local-address": "192.0.2.1/32",
                "address-allocation-type": "static-address",
                "static-addresses": {
                  "primary-address": "1",
                  "address": [
                    {
                      "address-id": "1",
                      "s-customer-address": "192.0.2.2"
                    }
                  ]
                }
              }
            },
            "routing-protocols": {
              "routing-protocol": [
                {
                  "id": "1",
                  "type": "vpn-common:direct"
                }
              ]
            }
          },
          {
            "vpn-network-access-id": "1/1/1.2",
            "port-id": "1/1/1",
            "description": "Interface DATA to eNODE-B",
            "admin-status": {
              "status": "vpn-common:administrative-state-up"
            },
            "ip-connection": {
              "ipv4": {
                "local-address": "192.0.2.1/32",
                "address-allocation-type": "static-address",
                "static-addresses": {
                  "primary-address": "1",
                  "address": [
                    {
                      "address-id": "1",
                      "customer-address": "192.0.2.2"
                    }
                  ]
                }
              }
```

```
              },
              "routing-protocols": {
                "routing-protocol": [
                  {
                    "id": "1",
                    "type": "vpn-common:direct"
                  }
                ]
              }
            }
          ]
        }
      }
```


Figure 33: Create VPN Network Access

Similar actions can be followed when IPv6 is supported in a VPN. For example, Figure 34 illustrates how to create a VPN node that is identified with an 'ne-id' set to 2001:db8::1.

```
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": "2001:db8::1",
        "local-autonomous-system": "65550",
        "rd": "0:65550:1",
        "vpn-targets": {
          "vpn-target": [
            {
              "id": "1",
              "route-targets": [
                "0:65550:1"
              ],
              "route-target-type": "both"
            }
          ]
        }
      }
    ]
  }
}
```


Figure 34: Create VPN Node (IPv6)

An example of creating a loopback interface is depicted in .

```
        {
          "ietf-l3vpn-ntw:vpn-network-accesses": {
            "vpn-network-access": [
              {
                "vpn-network-access-id": "Loopback1",
                "port-id": "Loopback1",
                "description": "An example of loopback interface.",
                "admin-status": {
                  "status": "vpn-common:administrative-state-up"
                },
                "vpn-network-access-type": "vpn-common:loopback",
                "ip-connection": {
                  "ipv6": {
                    "local-address": "2001:db8::1/128"
                  }
                }
              }
            ]
          }
        }
```


Figure 35: Create a Loopback Interface

A.2. Multicast VPN Provisioning Example

IPTV is mainly distributed through multicast over the LANs. In the following example, 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 PE and CE is achieved using BGP. Also, RP is statically configured for a multicast group.

```
              +-----------+   +-------+     +-------+    +-----------+
              | Multicast |---|  CE   |--/--|  PE   |----|  Backbone |
              |  source   |   +-------+     +-------+    |   IP/MPLS |
              +-----------+                              +-----------+
```

Figure 36: Multicast L3VPN Service Example

To configure a Multicast L3VPN service using the L3NM model the procedure and the JSON with the data structure is the following:

First, the multicast service is created (see the excerpt of the request message body shown in Figure 37)

```
{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "Multicast-IPTV",
        "customer-name": "310",
        "vpn-service-topology": "vpn-common:hub-spoke",
        "description": "Multicast IPTV VPN service"
      }
    ]
  }
}
```


Figure 37: 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 38). 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",
      "node-role": "vpn-common:hub-role",
      "local-autonomous-system": "3816",
      "address-family": "vpn-common:ipv4",
      "router-id": "198.51.100.10",
      "rd": "3816:31050202",
      "multicast": {
        "status": {
          "admin-status": {
            "status": "vpn-common:administrative-state-up"
          }
        },
        "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": "vpn-common:static-rp"
          }
        }
      }
    }
  ]
}
```


Figure 38: 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 39).

```
       {
         "ietf-l3vpn-ntw:vpn-network-access": {
           "vpn-network-access-id": "1/1/1",
           "description": "Connected_to_source",
           "status": {
             "admin-status": {
               "status": "vpn-common:administrative-state-up"
           },
           "vpn-network-access-type": "vpn-common:point-to-point",
           "ip-connection": {
             "ipv4": {
               "local-address": "203.0.113.1/32",
               "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": "vpn-common:bgp",
                 "bgp": {
                   "description": "Connected to CE"
                   "local-autonomous-system": "3816",
                   "peer-autonomous-system": "6500",
                   "address-family": "vpn-common:ipv4",
                   "neighbor": "203.0.113.2",
                 }
               }
             ]
           },
           "service": {
             "multicast": {
               "multicast-site-type": "source-only",
               "address-family": "vpn-common:ipv4",
               "protocol-type": "router",
               "pim": {
                 "hello-interval": 30,
                 "status": {
                   "admin-status": {
```

```
                     "status": "vpn-common:administrative-state-up"
                   }
                 }
               }
             }
           }
         }
       }
```


Figure 39: Create VPN Network Access (Excerpt of the Message Request Body)

Appendix B. Implementation Status

This section records the status of known implementations of the Yang module defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note the RFC Editor: As per [RFC7942] guidelines, please remove this Implementation Status apendix prior publication.

B.1. Nokia Implementation

Details can be found at: https://github.com/IETF-OPSAWG-WG/l3nm/blob/master/Implementattion/Nokia.txt

B.2. Huawei Implementation

Details can be found at: https://github.com/IETF-OPSAWG-WG/l3nm/blob/master/Implementattion/Huawei.txt

B.3. Infinera Implementation

Details can be found at: https://github.com/IETF-OPSAWG-WG/l3nm/blob/master/Implementattion/Infinera.txt

B.4. Ribbon-ECI Implementation

Details can be found at: https://github.com/IETF-OPSAWG-WG/l3nm/blob/master/Implementattion/Ribbon-ECI.txt

Authors' Addresses

Samier Barguil
Telefonica
Madrid
ES

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone
ES

Email: luis-angel.munoz@vodafone.com

Alejandro Aguado
Nokia
Madrid
ES

Email: alejandro.aguado_martin@nokia.com