OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

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

March 09, 2020

A Layer 3 VPN Network YANG Model
draft-ietf-opsawg-l3sm-l3nm-02

Abstract

This document defines a L3 VPN Network YANG Data model, called L3NM that can be used to manage the provisioning of Layer 3 VPN services within a Service Provider Network. The module is meant to be used by a Network Controller to derive the configuration information that will be sent to relevant network devices.

The L3VPN Network YANG Model (L3NM) can also facilitates the communication between a service orchestrator and a network controller/orchestrator. The model provides a network-centric view of the L3VPN services.

The L3NM YANG module is aimed at managing BGP PE-based Layer 3 VPNs as described in RFCs 4026, 4110 and 4364 and Multicast VPNs as described in RFCs 6037, 6513 and 7988.

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Requirements Language 4
3. Terminology 4
4. Reference Architecture 6
5. Relation with other YANG Models 8
6. Description of the L3NM YANG Module 10
  6.1. Overall Structure of the Module 10
  6.2. VPN Profiles 10
  6.3. Modeling a Layer 3 VPN Service 11
    6.3.1. Service Status 12
    6.3.2. VPN Node 13
      6.3.2.1. Node Status 15
      6.3.2.2. VPN Network Access 15
        6.3.2.2.1. Connection 17

6.3.2.2.2. IP Connections 20
6.3.2.2.3. CE PE Routing Protocols 22
6.3.2.3. Multicast 26
6.3.3. Concept of Import/Export Profiles 28
6.3.4. Underlay Transport 28
7. L3NM Module Tree Structure 28
8. Sample Uses of the L3NM Data Model 39
8.1. Enterprise L3 VPN Services 39
8.2. Multi-Domain Resource Management 39
8.3. Management of Multicast services 39
9. L3VPN Examples 40
9.1. 4G VPN Provissioning Example 40
9.2. Multicast VPN Provisioning Example 44
10. L3NM YANG Module 46
11. IANA Considerations 108
12. Security Considerations 109
13. Acknowledgements 110
14. Contributors 110
15. References 111
15.1. Normative References 111
15.2. Informative References 112
Appendix A. Implementation Status 113
A.1. Nokia Implementation 113
A.2. Huawei Implementation 114
A.3. Infinera Implementation 118
Authors' Addresses 118

1. Introduction

[RFC8299] defines an L3VPN 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 VPN services, and provides an abstracted view of the customer's requested services. That approach limits the usage of the L3SM module to the role of a Customer Service Model, according to the terminology defined in [RFC8309].

The YANG data model defined in this document is called L3VPN Network Model (L3NM). The L3NM module is aimed at providing a network-centric view of L3 VPN Services. The 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 synchronized.

This document does not obsolete, but uses, the definitions in [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, this module 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 supposed to maintain a "glue" between a Customer view and its network instantiation. Likewise, some of the information captured and exposed using L3NM can fed the service layer (e.g., capabilities) to derive L3SM and drive VPN service order handling.

The L3NM module 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 end-to-end VPN connectivity are addressed by a complementary YANG model defined in [I-D.evenwu-opsawg-yang-composed-vpn].

2. Requirements Language

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.

3. Terminology

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.

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

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.

This document makes use of the following terms:

- L3 VPN Customer Service Model (L3SM): Describes the requirements of a L3 VPN 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 L3 VPN Customer Service model is defined in [RFC8299].

- L3 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 L3 VPN. The Service Orchestrator interacts with the customer using L3SM. The Service Orchestrator is responsible of the CE-PE attachment circuits, the PE selection, and requesting the VPN service to the network controller.

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

- VPN node (vpn-node): An abstraction that represents a set of policies applied to a PE and that belong to a single VPN service (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 VRF.

- VPN network access (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 (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 (SP): A Service Provider offers VPN-related services [RFC4176].

- Service Provider (SP) Network: A network able to provide VPN-related services.

4. Reference Architecture

Figure 1 depicts the reference architecture for L3NM. The figure is an expansion of the architecture presented in Section 5 of [RFC8299] and decomposes the box marked "orchestration" in that figure into three separate functional components called "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]. The above 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".

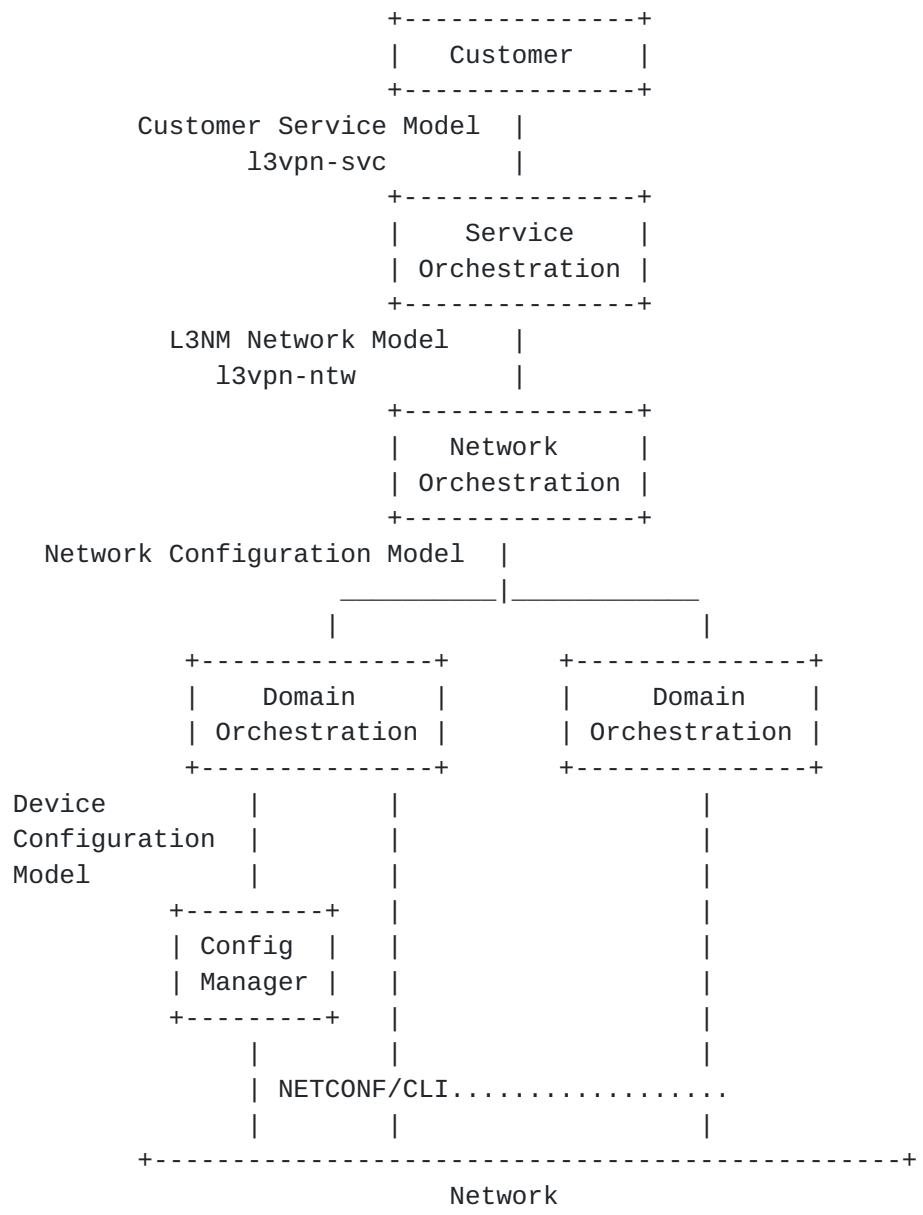

Figure 1: L3SM and L3NM

The L3SM and L3NM modules may also be set in the context of the ACTN architecture [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC). It also shows the interfaces between these functional blocks: the CNC-MDSC Interface (CMI), the MDSC-PNC Interface (MPI), and the Southbound Interface (SBI).

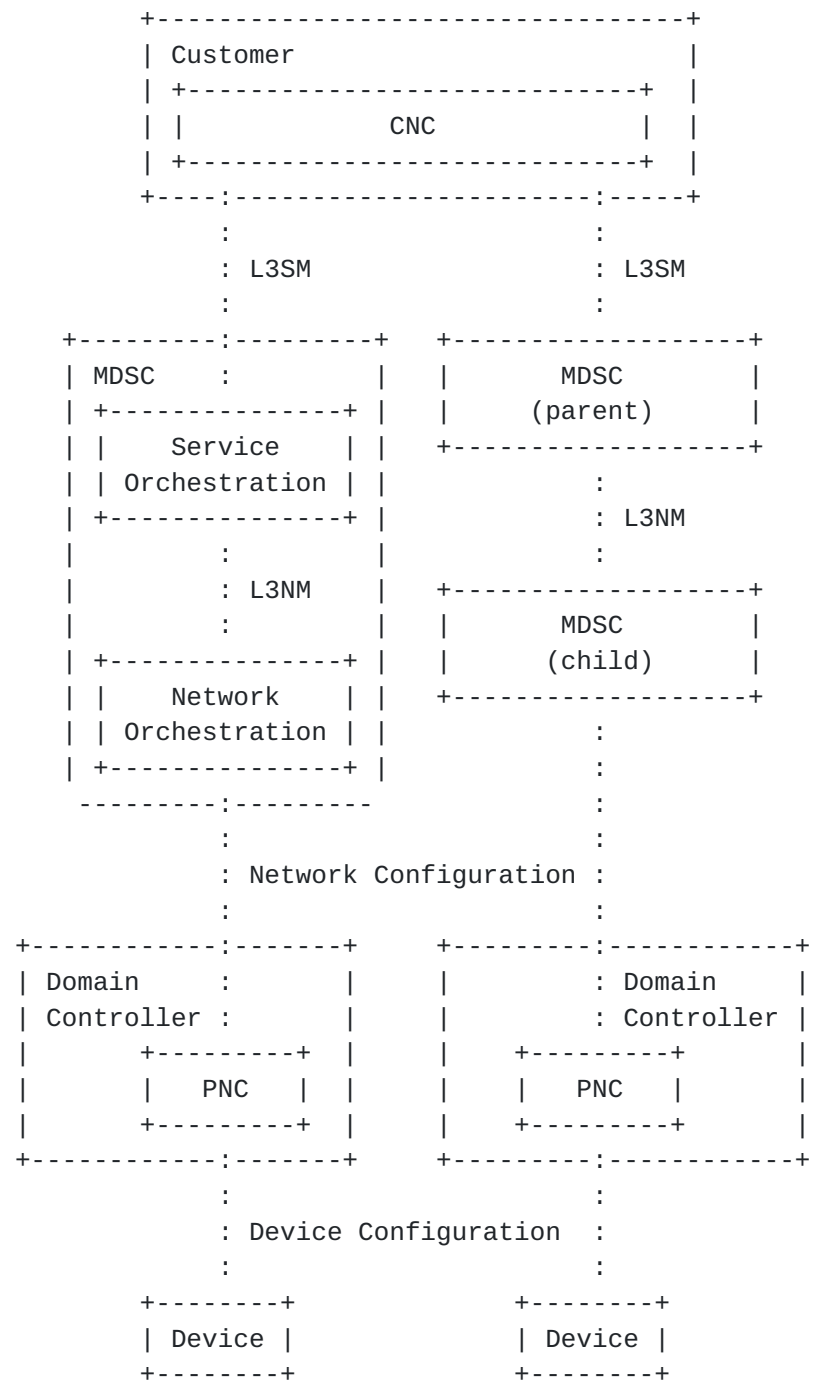

Figure 2: L3SM and L3NM in the Context of ACTN

5. Relation with other YANG Models

As discussed in the previous section, the L3NM YANG module is meant to manage L3VPN Services within a Service Provider network. The module provides a network-wise view of the service. Such view is

only visible within the Service Provider and is not exposed outside. The following discusses how L3NM interfaces with other YANG modules:

L3SM: L3NM is not a Customer Service Model.

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

Typically, the L3NM module 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 assume nor preclude exposing the VPN service via 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 Backbone 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.liu-pim-yang])

* NAT management ([RFC8512])

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

* ACL ([RFC8519])

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

6. Description of the L3NM YANG Module

The L3NM module ('ietf-l3vpn-ntw') is meant to manage L3 VPNs 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 detailed tree structure is provided in Figure 15.

6.1. Overall Structure of the Module

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

The 'vpn-services' container maintains the set of VPN services managed within the service provider's network. 'vpn-service' is the data structure that abstracts a VPN service (Section 6.3).

The 'vpn-profiles' container is used by the provider to maintain a set of common VPN profiles that apply to several VPN services (Section 6.2).

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
         +--rw vpn-service* [vpn-id]
            ...
```

Figure 3: Overall L3NM Tree Structure

6.2. VPN Profiles

The 'vpn-profiles' containers (Figure 4) allow the network provider to define and maintain a set of common VPN profiles that apply to several VPN services. The exaact definition of the profiles is local to each network provider.

```
      +--rw l3vpn-ntw
         +--rw vpn-profiles
         |  +--rw valid-provider-identifiers
         |     +--rw cloud-identifier* [id] {l3vpn-svc:cloud-access}?
         |     |  +--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 routing-profile-identifier* [id]
         |        +--rw id    string
         +--rw vpn-services
             +--rw vpn-service* [vpn-id]
                ...
```

Figure 4: VPN Profiles Tree Structure

6.3. Modeling a Layer 3 VPN Service

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.

In order to facilitate the identification of the service, 'customer-name' and 'description' attributes may be provided.

The 'vpn-service' parameters are:

- service-status: Allows the control of the operative and administrative status of the service as a whole.
- vpn-id: Unique identifier of the L3VPN Service within L3NM scope.
- l3sm-vpn-id: Refers to the L3SNM Id of this service. This identifier allows to easily correlate the service as built in the network with a service request.
- vpn-service-topology: Typical network topologies are supported. Hub-Spoke, Any-to-Any, and Custom. Real deployment on the network is defined by the correct usage of import and export profiles
- ie-profiles: Define reusable import/export policies for the same VPN-Service. Described in detail in Section 6.3.3

- Underlay-Transport: Describes the preference for the transport technology to carry the traffic of the VPN-Service.

A VPN service is typically built by adding instances of 'vpn-node' to the 'vpn-nodes' container. The '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-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 model.

```
+--rw vpn-service* [vpn-id]
   +--rw service-status
   |  ...
   +--rw vpn-id                  l3vpn-svc:svc-id
   +--rw l3sm-vpn-id?            l3vpn-svc:svc-id
   +--rw customer-name?          string
   +--rw vpn-service-topology?   identityref
   +--rw description?            string
   +--rw ie-profiles
   |  ...
   +--rw underlay-transport
   |  ...
```

Figure 5: vpn-service tree structure

6.3.1. Service Status

The L3NM module allows to track service status ('service-status') of a given VPN service (Figure 6). Both operational and administrative status are maintained together with a timestamp. For example, a service can be created but not put into effect.

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

```
   +--rw l3vpn-ntw
      +--rw vpn-profiles
      |  ...
      +--rw vpn-services
         +--rw vpn-service* [vpn-id]
            +--rw service-status
            |  +--rw admin
            |  |  +--rw status?      operational-type
            |  |  +--rw timestamp?   yang:date-and-time
            |  +--ro ops
            |     +--ro status?      operational-type
            |     +--ro timestamp?   yang:date-and-time
            ...
```

Figure 6: VPN Service Status Tree Structure

6.3.2. VPN Node

The 'vpn-node' is an abstraction that represents a set of common policies applied on a given network node (tipcally, a PE) and belong to one L3 VPN Service. In order to indicate the network node where the 'vpn-node' applies the 'ne-id' must be indicated. The 'vpn-node' includes a parameter to indicate in which network node 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. Additionally the 'vpn-node' parameters are:

- status: Allows the control of the operative and administrative status of the 'vpn-node'.

- local-autonomous-system: Autonomous system of locally configured in the instance. It can be overwritten for specific purposes in the CE-PE BGP session.

- maximum-routes: Max-number of prefixes allowed in the vpn-node instance.

- rd and vpn-targets: For the cases the logical resources are managed outside the network controller, the model allows to explicitely indicate the logical resources such as Route targets (RTs) and Route Distinguishers (RDs) (RT,RD).

- Multicast: Enable multicast traffic inside the vpn. Detailed description in Section 6.3.2.3

Under the VPN Node ('vpn-node') container, VPN Network Acesses ('vpn-network-access') can be created. The VPN Network Acess 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. 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 than can be applied during the service creation.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                 l3vpn-svc:svc-id
       + ...
       +--rw vpn-nodes
        +--rw vpn-node* [ne-id]
           +--rw vpn-node-id?              union
           +--rw local-autonomous-system?  inet:as-number
           +--rw description?              string
           +--rw ne-id                     string
           +--rw router-id?                inet:ip-address
           +--rw address-family?
           |       l3vpn-svc:address-family
           +--rw node-role?                identityref
           +--rw rd?
           |       rt-types:route-distinguisher
           +--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?   leafref
           |     +--rw export-policy?   leafref
           +--rw status
           |  +--rw admin-enabled?   boolean
           |  +--ro oper-status?     operational-type
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
```

```
|     ...
+--rw maximum-routes
|  +--rw address-family* [af]
|     +--rw af
|     |       l3vpn-svc:address-family
|     +--rw maximum-routes?   uint32
+--rw multicast {l3vpn-svc:multicast}?
|  ...
+--rw node-ie-profile?           leafref
```

Figure 7: VPN Node Tree Structure

6.3.2.1. Node Status

The L3NM module allows to track the status ('status') of the nodes involved in a VPN service (Figure 8). Both operational and administrative status are maintained. Mismatch between an administrative status vs. the operational status can be used as trigger to detect anomalies.

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           +--rw vpn-id                 l3vpn-svc:svc-id
           ...
           +--rw vpn-nodes
           |  +--rw vpn-node* [ne-id]
           |     +--rw ne-id                  string
           |     ...
           |     +--rw status
           |     |  +--rw admin-enabled?   boolean
           |     |  +--ro oper-status?     operational-type
```

Figure 8: Node Status Tree Structure

6.3.2.2. VPN Network Access

A 'vpn-network-access' represents an entry point to a VPN service (Figure 9). In other words, this container encloses the parameters that describe the access information for the traffic that belongs to a particular L3VPN. As such, every 'vpn-network-access' MUST belong to one and only one 'vpn-node'.

A 'vpn-network-access' includes information such as the connection on which the access is defined (see Section 6.3.2.2.1), the

encapsulation of the traffic, policies that are applied on the access, etc.

A provisioning Network Controller (PNC) [RFC8453] will accept VPN requests containing this construct, using the enclosed data to: configure the router's interface to include the parameters described at the 'vpn-network-access', include the given interface into a VRF, configuring policies or schedulers for processing the incoming traffic, etc.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                  l3vpn-svc:svc-id
       +  ...
        +--rw vpn-node* [ne-id]
           +--rw ne-id                      string
           + ...
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
           |     +--rw port-id?
           |     |       l3vpn-svc:svc-id
           |     +--rw description?              string
           |     +--rw status
           |     |  +--rw admin-enabled?   boolean
           |     |  +--ro oper-status?     operational-type
           |     +--rw vpn-network-access-type?   identityref
           |     +--rw connection
           |     |  ...
           |     |  +--rw bearer
           |     |     ...
           |     +--rw ip-connection
           |     |  ...
           |     +--rw security
           |     |  ...
           |     +--rw routing-protocols
           |     |  ...
           |     +--rw service
           |        ...
           |  ...
```

Figure 9: VPN Network Access Tree Structure

6.3.2.2.1. Connection

The definition of a L3VPN is commonly specified not only at the IP layer, but also requires to identify parameters at the Ethernet layer, such as encapsulation type (e.g., VLAN, QinQ, QinAny, VxLAN, etc.). The 'connection' container represents and groups the set of L2 connectivity from where the traffic of the L3VPN in a particular VPN Network access is coming.

Additionally, the bearer-reference and the pseudowire termination are supported.

Ethernet encapsulation description is not supported in [RFC8299]. However, this parameters are mandatory to configure the PE interfaces. Thus, In the L3NM, these parameters uses the connection container inside the vpn-network-access. This container defines protocols and parameters to enable connectivity at Layer 2.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                  l3vpn-svc:svc-id
       +  ...
        +--rw vpn-node* [ne-id]
           +--rw ne-id                      string
           + ...
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
           |     + ...
           |     +--rw connection
           |     |  +--rw encapsulation-type?   identityref
           |     |  +--rw logical-interface
           |     |  |  +--rw peer-reference?   uint32
           |     |  +--rw tagged-interface
           |     |  |  +--rw type?                identityref
           |     |  |  +--rw dot1q-vlan-tagged {dot1q}?
           |     |  |  |  +--rw tag-type?   identityref
           |     |  |  |  +--rw cvlan-id?   uint16
           |     |  |  +--rw priority-tagged
           |     |  |  |  +--rw tag-type?   identityref
           |     |  |  +--rw qinq {qinq}?
           |     |  |  |  +--rw tag-type?   identityref
           |     |  |  |  +--rw svlan-id    uint16
```

```
|     |  |  |  +--rw cvlan-id    uint16
|     |  |  +--rw qinany {qinany}?
|     |  |  |  +--rw tag-type?   identityref
|     |  |  |  +--rw svlan-id    uint16
|     |  |  +--rw vxlan {vxlan}?
|     |  |     +--rw vni-id       uint32
|     |  |     +--rw peer-mode?   identityref
|     |  |     +--rw peer-list* [peer-ip]
|     |  |        +--rw peer-ip    inet:ip-address
|     |  +--rw bearer
|     |     ...
|     +--rw ip-connection
|     |  ...
|     +--rw security
|     |  ...
|     +--rw routing-protocols
|     |  ...
|     +--rw service
|        ...
|  ...
```

Figure 10: Encapsulation Tree Structure

Depending on the control plane implementation, different network scenarios might require additional information for the L3VPN service to be configured and active. For example, an L3VPN Option C service, if no reflection of IPv4 VPN routes is configured via ASBR or route reflector, may require additional configuration (e.g., a new BGP neighbor) to be coordinated between both management systems. This definition requires for every management system participant in the VPN to receive not just their own sites and site-network-accesses, but also to receive information about external ones, identified as an external site-network-access-type. In addition, this particular site-network-access is augmented to include the loopback address of the far-end (remote/external) PE router.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                 l3vpn-svc:svc-id
       +  ...
        +--rw vpn-node* [ne-id]
           +--rw ne-id                     string
           + ...
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
           |     +  ...
           |     +--rw connection
           |     |  ...
           |     |  +--rw bearer
           |     |     +--rw bearer-reference?   string
           |     |     |       {l3vpn-svc:bearer-reference}?
           |     |     +--rw pseudowire
           |     |     |  +--rw vcid?      uint32
           |     |     |  +--rw far-end?   union
           |     |     +--rw vpls
           |     |        +--rw vcid?      union
           |     |        +--rw far-end?   union
           |     +--rw ip-connection
           |     |  ...
           |     +--rw security
           |     |  ...
           |     +--rw routing-protocols
           |     |  ...
           |     +--rw service
           |        ...
           |  ...
```

Figure 11: Bearer Tree Structure

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 module it is assumed that the bearer has been allocated by the Service Provider at the service orchestration step. The bearer is associated to a network element and a port. Hence, a bearer is just a bearer-reference to allow the translation between L3SM and L3NM.

6.3.2.2.2. IP Connections

IP connection container (Figure 12) has the parameters of the 'vpn-network-access' addressing information. The address allocated in this container would represent the PE interface address configuration. The IP connection container is designed to support both IPv4 and IPv6. It also supports three options for IP address assignment: Provider DHCP, DHCP relay, and static addressing.

In the case of the static addressing, 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'.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                  l3vpn-svc:svc-id
       +  ...
       +--rw vpn-nodes
        +--rw vpn-node* [ne-id]
           +--rw ne-id                      string
           +  ...
           +--rw status
           |  +--rw admin-enabled?   boolean
           |  +--ro oper-status?     operational-type
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
           |     +  ...
           |     +--rw connection
           |     |  ...
           |     +--rw ip-connection
           |     |  +--rw ipv4 {l3vpn-svc:ipv4}?
           |     |  |  +--rw address-allocation-type?
           |     |  |  |       identityref
           |     |  |  +--rw provider-dhcp
           |     |  |  |  +--rw provider-address?
           |     |  |  |  |       inet:ipv4-address
           |     |  |  |  +--rw prefix-length?
           |     |  |  |  |       uint8
           |     |  |  |  +--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
|     |  |  |  +--rw dhcp-relay
|     |  |  |  |  +--rw provider-address?
|     |  |  |  |  |       inet:ipv4-address
|     |  |  |  |  +--rw prefix-length?           uint8
|     |  |  |  |  +--rw customer-dhcp-servers
|     |  |  |  |     +--rw server-ip-address*
|     |  |  |  |             inet:ipv4-address
|     |  |  |  +--rw static-addresses
|     |  |  |     +--rw primary-address?   leafref
|     |  |  |     +--rw address* [address-id]
|     |  |  |        +--rw address-id          string
|     |  |  |        +--rw provider-address?
|     |  |  |        |       inet:ipv4-address
|     |  |  |        +--rw customer-address?
|     |  |  |        |       inet:ipv4-address
|     |  |  |        +--rw prefix-length?      uint8
|     |  |  +--rw ipv6 {l3vpn-svc:ipv6}?
|     |  |  |  +--rw address-allocation-type?
|     |  |  |  |       identityref
|     |  |  |  +--rw provider-dhcp
|     |  |  |  |  +--rw provider-address?
|     |  |  |  |  |       inet:ipv6-address
|     |  |  |  |  +--rw prefix-length?
|     |  |  |  |  |       uint8
|     |  |  |  |  +--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
|     |  |  +--rw dhcp-relay
|     |  |  |  +--rw provider-address?
|     |  |  |  |       inet:ipv6-address
|     |  |  |  +--rw prefix-length?           uint8
|     |  |  |  +--rw customer-dhcp-servers
|     |  |  |     +--rw server-ip-address*
|     |  |  |             inet:ipv6-address
|     |  |  +--rw static-addresses
|     |  |     +--rw primary-address?   leafref
|     |  |     +--rw address* [address-id]
|     |  |        +--rw address-id          string
|     |  |        +--rw provider-address?
|     |  |        |       inet:ipv6-address
|     |  |        +--rw customer-address?
|     |  |        |       inet:ipv6-address
|     |  |        +--rw prefix-length?      uint8
|     |  +--rw oam
|     |     +--rw bfd {l3vpn-svc:bfd}?
|     |        +--rw enabled?              boolean
|     |        +--rw (holdtime)?
|     |           +--:(fixed)
|     |           |  +--rw fixed-value?    uint32
|     |           +--:(profile)
|     |              +--rw profile-name?   leafref
|     +--rw security
|     |  ...
|     +--rw routing-protocols
|     |  ...
|     +--rw service
|       ...
```

Figure 12: IP Connection Tree Structure

6.3.2.2.3. CE PE Routing Protocols

The model allows the Provider to configure one or more routing protocols associated with a particular 'vpn-network-access' (Figure 13). This protocol will run between the PE and the CE. A routing protocol instance MUST have a type (e.g., bgp, ospf) and an identifier. The identifier is necessary when multiple instances of the same protocol have to be configured.

When configuring multiple instances of the same routing protocol, this does not automatically imply that, from a device configuration perspective, there will be parallel instances (multiple processes)

running. It will be up to the implementation to use the most appropriate deployment model. As an example, when multiple BGP peers need to be implemented, multiple instances of BGP must be configured as part of this model. However, from a device configuration point of view, this could be implemented as:

- Multiple BGP processes with a single neighbor running in each process.
- A single BGP process with multiple neighbors running.
- A combination of both.

To be aligned with [RFC8299], this model supports the following protocols:

- VRRP: takes only a list of address-family as parameter. VRRP instance is expected to run on the 'vpn-network-access' interface.
- RIP: takes only a list of address-family as parameter. RIP instance is expected to run on the 'vpn-network-access' interface.
- BGP: allows to configure a BGP neighbor including parameters like authentication using a key. The authentication type will be driven by the implementation but the module supports any authentication that uses a key as a parameter. A BGP neighbor can support IPv4, IPv6, or both address families. The module supports supplying two neighbors (each for a given address family) or one neighbor (for both IPv4 and IPv6 of "address-family" attribute is set to both). It is then up to the implementation to drive the device configuration.
- OSPF: allows the user to configure OSPF to run on the vpn-network-access interface. An OSPF instance can run ipv4, ipv6 or both. When only ipv4 address-family is requested, it will be up to the implementation to drive if OSPFv2 or v3 is used.
- IS-IS: allows the user to configure IS-IS to run on the vpn-network-access interface. An IS-IS instance can run L1, L2 or both levels.

The module allows a user to configure one or more IPv4 and/or IPv6 static routes.

Routing configuration does not include low-level policies. These policies are low level device configurations that must not be part of an abstracted model. A provider's internal policies (such as security filters) will be implemented as part of the device

configuration but does not require any input from this model. Some policies like primary/backup or load-balancing can be inferred from 'access-priority'.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                  l3vpn-svc:svc-id
       +  ...
       +--rw vpn-nodes
        +--rw vpn-node* [ne-id]
           +--rw ne-id                      string
           +  ...
           +--rw status
           |  +--rw admin-enabled?   boolean
           |  +--ro oper-status?     operational-type
           +--rw vpn-network-accesses
           |  +--rw vpn-network-access* [id]
           |     +--rw id
           |     |       l3vpn-svc:svc-id
           |     +  ...
           |     +--rw connection
           |     |  ...
           |     +--rw ip-connection
           |     |  ...
           |     |  +--rw oam
           |     |     ...
           |     +--rw security
           |     |  ...
           |     +--rw routing-protocols
           |     |  +--rw routing-protocol* [id]
           |     |     +--rw id                  string
           |     |     +--rw type?               identityref
           |     |     +--rw routing-profiles* [id]
           |     |     |  +--rw id      leafref
           |     |     |  +--rw type?   ie-type
           |     |     +--rw ospf {l3vpn-svc:rtg-ospf}?
           |     |     |  +--rw address-family*
           |     |     |  |       l3vpn-svc:address-family
           |     |     |  +--rw area-address
           |     |     |  |       yang:dotted-quad
           |     |     |  +--rw metric?           uint16
           |     |     |  +--rw mtu?              uint16
           |     |     |  +--rw process-id?       uint16
           |     |     |  +--rw security
```

```
               |     |     |  |  +--rw auth-key?   string
               |     |     |  +--rw sham-links
               |     |     |          {rtg-ospf-sham-link}?
               |     |     |     +--rw sham-link* [target-site]
               |     |     |        +--rw target-site
               |     |     |        |       l3vpn-svc:svc-id
               |     |     |        +--rw metric?        uint16
               |     |     +--rw bgp {l3vpn-svc:rtg-bgp}?
               |     |     |  +--rw peer-autonomous-system
               |     |     |  |       inet:as-number
               |     |     |  +--rw local-autonomous-system?
               |     |     |  |       inet:as-number
               |     |     |  +--rw address-family*
               |     |     |  |       l3vpn-svc:address-family
               |     |     |  +--rw neighbor*
               |     |     |  |       inet:ip-address
               |     |     |  +--rw multihop?
               |     |     |  |       uint8
               |     |     |  +--rw security
               |     |     |  |  +--rw auth-key?   string
               |     |     |  +--rw status
               |     |     |  |  +--rw admin-enabled?   boolean
               |     |     |  |  +--ro oper-status?
               |     |     |  |          operational-type
               |     |     |  +--rw description?
               |     |     |          string
               |     |     +--rw isis {rtg-isis}?
               |     |     |  +--rw address-family*
               |     |     |  |       l3vpn-svc:address-family
               |     |     |  +--rw area-address      area-address
               |     |     |  +--rw level?            isis-level
               |     |     |  +--rw metric?           uint16
               |     |     |  +--rw process-id?       uint16
               |     |     |  +--rw mode?             enumeration
               |     |     |  +--rw status
               |     |     |     +--rw admin-enabled?   boolean
               |     |     |     +--ro oper-status?
               |     |     |             operational-type
               |     |     +--rw static
               |     |     |  +--rw cascaded-lan-prefixes
               |     |     |     +--rw ipv4-lan-prefixes*
               |     |     |     |       [lan next-hop]
               |     |     |     |       {l3vpn-svc:ipv4}?
               |     |     |     |  +--rw lan
               |     |     |     |  |       inet:ipv4-prefix
               |     |     |     |  +--rw lan-tag?    string
               |     |     |     |  +--rw next-hop
               |     |     |     |          inet:ipv4-address
```

```
|     |     |     +--rw ipv6-lan-prefixes*
|     |     |             [lan next-hop]
|     |     |             {l3vpn-svc:ipv6}?
|     |     |        +--rw lan
|     |     |        |       inet:ipv6-prefix
|     |     |        +--rw lan-tag?    string
|     |     |        +--rw next-hop
|     |     |                inet:ipv6-address
|     |     +--rw rip {l3vpn-svc:rtg-rip}?
|     |     |  +--rw address-family*
|     |     |          l3vpn-svc:address-family
|     |     +--rw vrrp {l3vpn-svc:rtg-vrrp}?
|     |        +--rw address-family*
|     |                l3vpn-svc:address-family
|     +--rw service
|       ...
```

Figure 13: Routing Tree Structure

6.3.2.3. Multicast

Multicast MAY be enabled for a particular vpn-network-node (see Figure 14).

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. 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) 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.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
    +--rw vpn-service* [vpn-id]
       +--rw vpn-id                  l3vpn-svc:svc-id
       + ..
       +--rw vpn-nodes
        +--rw vpn-node* [ne-id]
           +--rw ne-id                      string
           +  ...
           +--rw vpn-network-accesses
           |  ...
           +--rw multicast {l3vpn-svc:multicast}?
           |  +--rw enabled?       boolean
           |  +--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 enabled?         boolean
|     +--rw peer?            inet:ip-address
|     +--rw local-address?   inet:ip-address
+ ...
```

Figure 14: Multicast Tree Structure

6.3.3. Concept of Import/Export Profiles

The import and export profiles construct contains a list with information related with route target and distinguishers (RTs and RDs), grouped and identified by ie-profile-id. The identifier is then referenced in one or multiple vpn-nodes, so the PNC can identify RTs and RDs to be configured in the VRF.

6.3.4. Underlay Transport

The model allows to indicate a preference for the underlay transport technology when activating a L3VPN service. This preference is especially useful in networks with multiple domains and NNI types. The model supports these option: BGP, LDP, GRE, SR, SR-TE, and RSVP-TE as possible underlay transport.

Other profiles can be defined in the future.

This document does not make any assumption about the exact definition of these profiles. How such profiles are defined is deployment-specific.

7. L3NM Module Tree Structure

The L3NM Module Tree Structure is depicted in Figure 15.

```
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
   +--rw vpn-profiles
   |  +--rw valid-provider-identifiers
   |     +--rw cloud-identifier* [id] {l3vpn-svc:cloud-access}?
```

```
   |     |  +--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 routing-profile-identifier* [id]
   |        +--rw id    string
   +--rw vpn-services
    +--rw vpn-service* [vpn-id]
       +--rw service-status
       |  +--rw admin
       |  |  +--rw status?      operational-type
       |  |  +--rw timestamp?   yang:date-and-time
       |  +--ro ops
       |     +--ro status?      operational-type
       |     +--ro timestamp?   yang:date-and-time
       +--rw vpn-id                  l3vpn-svc:svc-id
       +--rw l3sm-vpn-id?            l3vpn-svc:svc-id
       +--rw customer-name?          string
       +--rw vpn-service-topology?   identityref
       +--rw description?            string
       +--rw ie-profiles
       |  +--rw ie-profile* [ie-profile-id]
       |     +--rw ie-profile-id    string
       |     +--rw rd?              rt-types:route-distinguisher
       |     +--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?   leafref
       |           +--rw export-policy?   leafref
       +--rw underlay-transport
       |  +--rw type?   protocols-type
       +--rw vpn-nodes
        +--rw vpn-node* [ne-id]
           +--rw vpn-node-id?               union
           +--rw local-autonomous-system?   inet:as-number
           +--rw description?               string
           +--rw ne-id                      string
           +--rw router-id?                 inet:ip-address
           +--rw address-family?
```

```
|       l3vpn-svc:address-family
+--rw node-role?                identityref
+--rw rd?
|       rt-types:route-distinguisher
+--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?   leafref
|     +--rw export-policy?   leafref
+--rw status
|  +--rw admin-enabled?   boolean
|  +--ro oper-status?     operational-type
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       l3vpn-svc:svc-id
|     +--rw port-id?
|     |       l3vpn-svc:svc-id
|     +--rw description?               string
|     +--rw status
|     |  +--rw admin-enabled?   boolean
|     |  +--ro oper-status?     operational-type
|     +--rw vpn-network-access-type?   identityref
|     +--rw connection
|     |  +--rw encapsulation-type?   identityref
|     |  +--rw logical-interface
|     |  |  +--rw peer-reference?   uint32
|     |  +--rw tagged-interface
|     |  |  +--rw type?                identityref
|     |  |  +--rw dot1q-vlan-tagged {dot1q}?
|     |  |  |  +--rw tag-type?   identityref
|     |  |  |  +--rw cvlan-id?   uint16
|     |  |  +--rw priority-tagged
|     |  |  |  +--rw tag-type?   identityref
|     |  |  +--rw qinq {qinq}?
|     |  |  |  +--rw tag-type?   identityref
|     |  |  |  +--rw svlan-id    uint16
|     |  |  |  +--rw cvlan-id    uint16
|     |  |  +--rw qinany {qinany}?
|     |  |  |  +--rw tag-type?   identityref
|     |  |  |  +--rw svlan-id    uint16
|     |  |  +--rw vxlan {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
|     |     |       {l3vpn-svc:bearer-reference}?
|     |     +--rw pseudowire
|     |     |  +--rw vcid?      uint32
|     |     |  +--rw far-end?   union
|     |     +--rw vpls
|     |        +--rw vcid?      union
|     |        +--rw far-end?   union
|     +--rw ip-connection
|     |  +--rw ipv4 {l3vpn-svc:ipv4}?
|     |  |  +--rw address-allocation-type?
|     |  |  |       identityref
|     |  |  +--rw provider-dhcp
|     |  |  |  +--rw provider-address?
|     |  |  |  |       inet:ipv4-address
|     |  |  |  +--rw prefix-length?
|     |  |  |  |       uint8
|     |  |  |  +--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
|     |  |  +--rw dhcp-relay
|     |  |  |  +--rw provider-address?
|     |  |  |  |       inet:ipv4-address
|     |  |  |  +--rw prefix-length?           uint8
|     |  |  |  +--rw customer-dhcp-servers
|     |  |  |     +--rw server-ip-address*
|     |  |  |             inet:ipv4-address
|     |  |  +--rw static-addresses
|     |  |     +--rw primary-address?   leafref
|     |  |     +--rw address* [address-id]
|     |  |        +--rw address-id          string
|     |  |        +--rw provider-address?
```

```
|     |  |        |       inet:ipv4-address
|     |  |        +--rw customer-address?
|     |  |        |       inet:ipv4-address
|     |  |        +--rw prefix-length?      uint8
|     |  +--rw ipv6 {l3vpn-svc:ipv6}?
|     |  |  +--rw address-allocation-type?
|     |  |  |       identityref
|     |  |  +--rw provider-dhcp
|     |  |  |  +--rw provider-address?
|     |  |  |  |       inet:ipv6-address
|     |  |  |  +--rw prefix-length?
|     |  |  |  |       uint8
|     |  |  |  +--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
|     |  |  +--rw dhcp-relay
|     |  |  |  +--rw provider-address?
|     |  |  |  |       inet:ipv6-address
|     |  |  |  +--rw prefix-length?          uint8
|     |  |  |  +--rw customer-dhcp-servers
|     |  |  |     +--rw server-ip-address*
|     |  |  |             inet:ipv6-address
|     |  |  +--rw static-addresses
|     |  |     +--rw primary-address?   leafref
|     |  |     +--rw address* [address-id]
|     |  |        +--rw address-id          string
|     |  |        +--rw provider-address?
|     |  |        |       inet:ipv6-address
|     |  |        +--rw customer-address?
|     |  |        |       inet:ipv6-address
|     |  |        +--rw prefix-length?      uint8
|     |  +--rw oam
|     |     +--rw bfd {l3vpn-svc:bfd}?
|     |        +--rw enabled?             boolean
|     |        +--rw (holdtime)?
|     |           +--:(fixed)
|     |           |  +--rw fixed-value?    uint32
```

```
|     |          +--:(profile)
|     |             +--rw profile-name?   leafref
|     +--rw security
|     |  +--rw authentication
|     |  +--rw encryption {l3vpn-svc:encryption}?
|     |  |  +--rw enabled?   boolean
|     |  |  +--rw layer?     enumeration
|     |  +--rw encryption-profile
|     |     +--rw (profile)?
|     |     |  +--:(provider-profile)
|     |     |  |  +--rw profile-name?    leafref
|     |     |  +--:(customer-profile)
|     |     |     +--rw algorithm?       string
|     |     +--rw (key-type)?
|     |        +--:(psk)
|     |           +--rw preshared-key?   string
|     +--rw routing-protocols
|     |  +--rw routing-protocol* [id]
|     |     +--rw id                  string
|     |     +--rw type?               identityref
|     |     +--rw routing-profiles* [id]
|     |     |  +--rw id      leafref
|     |     |  +--rw type?   ie-type
|     |     +--rw ospf {l3vpn-svc:rtg-ospf}?
|     |     |  +--rw address-family*
|     |     |  |       l3vpn-svc:address-family
|     |     |  +--rw area-address
|     |     |  |       yang:dotted-quad
|     |     |  +--rw metric?           uint16
|     |     |  +--rw mtu?              uint16
|     |     |  +--rw process-id?       uint16
|     |     |  +--rw security
|     |     |  |  +--rw auth-key?   string
|     |     |  +--rw sham-links
|     |     |           {rtg-ospf-sham-link}?
|     |     |     +--rw sham-link* [target-site]
|     |     |        +--rw target-site
|     |     |        |       l3vpn-svc:svc-id
|     |     |        +--rw metric?        uint16
|     |     +--rw bgp {l3vpn-svc:rtg-bgp}?
|     |     |  +--rw peer-autonomous-system
|     |     |  |       inet:as-number
|     |     |  +--rw local-autonomous-system?
|     |     |  |       inet:as-number
|     |     |  +--rw address-family*
|     |     |  |       l3vpn-svc:address-family
|     |     |  +--rw neighbor*
|     |     |  |       inet:ip-address
```

```
|     |     |  +--rw multihop?
|     |     |  |       uint8
|     |     |  +--rw security
|     |     |  |  +--rw auth-key?   string
|     |     |  +--rw status
|     |     |  |  +--rw admin-enabled?   boolean
|     |     |  |  +--ro oper-status?
|     |     |  |          operational-type
|     |     |  +--rw description?
|     |     |          string
|     |     +--rw isis {rtg-isis}?
|     |     |  +--rw address-family*
|     |     |  |       l3vpn-svc:address-family
|     |     |  +--rw area-address      area-address
|     |     |  +--rw level?            isis-level
|     |     |  +--rw metric?           uint16
|     |     |  +--rw process-id?       uint16
|     |     |  +--rw mode?             enumeration
|     |     |  +--rw status
|     |     |     +--rw admin-enabled?   boolean
|     |     |     +--ro oper-status?
|     |     |              operational-type
|     |     +--rw static
|     |     |  +--rw cascaded-lan-prefixes
|     |     |     +--rw ipv4-lan-prefixes*
|     |     |     |       [lan next-hop]
|     |     |     |       {l3vpn-svc:ipv4}?
|     |     |     |  +--rw lan
|     |     |     |  |       inet:ipv4-prefix
|     |     |     |  +--rw lan-tag?    string
|     |     |     |  +--rw next-hop
|     |     |     |          inet:ipv4-address
|     |     |     +--rw ipv6-lan-prefixes*
|     |     |              [lan next-hop]
|     |     |              {l3vpn-svc:ipv6}?
|     |     |        +--rw lan
|     |     |        |       inet:ipv6-prefix
|     |     |        +--rw lan-tag?    string
|     |     |        +--rw next-hop
|     |     |                inet:ipv6-address
|     |     +--rw rip {l3vpn-svc:rtg-rip}?
|     |     |  +--rw address-family*
|     |     |          l3vpn-svc:address-family
|     |     +--rw vrrp {l3vpn-svc:rtg-vrrp}?
|     |        +--rw address-family*
|     |                l3vpn-svc:address-family
|     +--rw service
|        +--rw svc-input-bandwidth     uint64
```

```
|        +--rw svc-output-bandwidth    uint64
|        +--rw svc-mtu                 uint16
|        +--rw qos {l3vpn-svc: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 (dst-network)?
|        |  |     |  |  |  |     | +--:(dst-ipv4-network)
|        |  |     |  |  |  |     |  +--rw dst-ipv4-network?
|        |  |     |  |  |  |     |    inet:ipv4-prefix
|        |  |     |  |  |  |     +--rw (source-network)?
|        |  |     |  |  |  |      +--:(src-ipv4-network)
|        |  |     |  |  |  |        +--rw src-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)?
|       |  |     |  |  |       |  +--:(dst-ipv6-network)
|       |  |     |  |  |       |     +--rw dst-ipv6-network?
|       |  |     |  |  |       |        inet:ipv6-prefix
|       |  |     |  |  |       +--rw (src-network)?
|       |  |     |  |  |       |  +--:(src-ipv6-network)
|       |  |     |  |  |       |     +--rw src-ipv6-network?
|       |  |     |  |  |       |        inet:ipv6-prefix
|       |  |     |  |  |       +--rw flow-label?
|       |  |     |  |  |                inet:ipv6-flow-label
|       |  |     |  |  +--rw (l4)?
|       |  |     |  |     +--:(tcp)
|       |  |     |  |     |  +--rw tcp
|       |  |     |  |     |     +--rw sequence-number?
|       |  |     |  |     |     |       uint32
|       |  |     |  |     |     +--rw ack-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)?
|       |  |     |  |     |     |   ...
|       |  |     |  |     |     +--rw (destination-port)?
|       |  |     |  |     |     |  ...
|       |  |     |  |     +--:(udp)
|       |  |     |  |        +--rw udp
|       |  |     |  |           +--rw length?
|       |  |     |  |           |       uint16
|       |  |     |  |           +--rw (source-port)?
|       |  |     |  |           |  ...
|       |  |     |  |           +--rw (destination-port)?
|       |  |     |  |           |  ...
|       |  |     |  +--:(match-application)
|       |  |     |     +--rw match-application?
|       |  |     |             identityref
|       |  |     +--rw target-class-id?
|       |  |             string
```

```
|        |  +--rw qos-profile
|        |     +--rw (qos-profile)?
|        |        +--:(standard)
|        |        |  +--rw profile?     leafref
|        |        |  +--rw direction?   identityref
|        |        +--:(custom)
|        |           +--rw classes
|        |                   {l3vpn-svc:qos-custom}?
|        |              +--rw class* [class-id]
|        |                 +--rw class-id
|        |                 |       string
|        |                 +--rw direction?
|        |                 |       identityref
|        |                 +--rw rate-limit?
|        |                 |       decimal64
|        |                 +--rw latency
|        |                 |  +--rw (flavor)?
|        |                 |     +--:(lowest)
|        |                 |     |  +--rw use-lowest-latency?
|        |                 |     |          empty
|        |                 |     +--:(boundary)
|        |                 |        +--rw jitter-boundary?
|        |                 |                uint16
|        |                 +--rw jitter
|        |                 |  +--rw (flavor)?
|        |                 |     +--:(lowest)
|        |                 |     |  +--rw use-lowest-jitter?
|        |                 |     |          empty
|        |                 |     +--:(boundary)
|        |                 |        +--rw latency-boundary?
|        |                 |                uint32
|        |                 +--rw bandwidth
|        |                    +--rw guaranteed-bw-percent
|        |                    |       decimal64
|        |                    +--rw end-to-end?
|        |                            empty
|        +--rw carrierscarrier
|        |       {l3vpn-svc:carrierscarrier}?
|        |  +--rw signalling-type?   enumeration
|        +--rw multicast {l3vpn-svc:multicast}?
|           +--rw site-type?        enumeration
|           +--rw address-family
|           |  +--rw ipv4?   boolean
|           |  |       {l3vpn-svc:ipv4}?
|           |  +--rw ipv6?   boolean
|           |          {l3vpn-svc:ipv6}?
|           +--rw protocol-type?    enumeration
|           +--rw remote-source?    boolean
```

```
           +--rw maximum-routes
           |  +--rw address-family* [af]
           |     +--rw af
           |     |       l3vpn-svc:address-family
           |     +--rw maximum-routes?   uint32
           +--rw multicast {l3vpn-svc:multicast}?
           |  +--rw enabled?       boolean
           |  +--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 enabled?         boolean
           |     +--rw peer?            inet:ip-address
           |     +--rw local-address?   inet:ip-address
```

```
          +--rw node-ie-profile?          leafref
```

Figure 15

8. Sample Uses of the L3NM Data Model

8.1. Enterprise L3 VPN Services

Enterprise L3VPNs are one of the most demanded services for carriers, and therefore, L3NM can be useful to automate the tasks of 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 SDN layer and little manual intervention will be still required.

Also common addition/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 model object.

8.2. 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, RTs shall be synchronized between PEs. When every PE is controlled by the same management system, RT allocation can be performed by the system. In cases where the service spans across multiple management systems, this task of allocating RTs has to be aligned across the domains, therefore, the service model must provide a way to specify RTs. In addition, RDs must also be synchronized to avoid collisions in RD allocation between separate systems. An incorrect allocation might lead to the same RD and IP prefixes being exported by different PE routers.

8.3. Management of Multicast services

Multicast services over L3VPN can be implemented either using dual PIM MVPNs (also known as Draft Rosen model) [RFC 4364] or multiprotocol BGP (MBGP)-based MVPNs called Next Generation Multicast VPN (ng-MVPN) [RFC 6513/6514]. 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

backbone. Multiprotocol BGP multicast VPNs employ the intra-autonomous system (AS) next-generation BGP control plane and PIM sparse mode as the data plane. The PIM state information is maintained between the PE routers using the same architecture that is used for unicast VPNs.

On the other hand, Draft Rosen 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 this, ng-MNPN is the architectural model that has been taken as the base for implementing multicast service on L3VPN. In this scenario, BGP auto discovery is used to discover MVPN PE members and the customer PIM signaling is sent across provider core through MP-BGP. The multicast traffic is transported on MPLS P2MP LSPs. All of the previous information is carried in the MCAST-VPN BGP NRLI.

9. L3VPN Examples

9.1. 4G VPN Provissioning 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 requiremets.

As it is shown in the Figure 16, 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'.

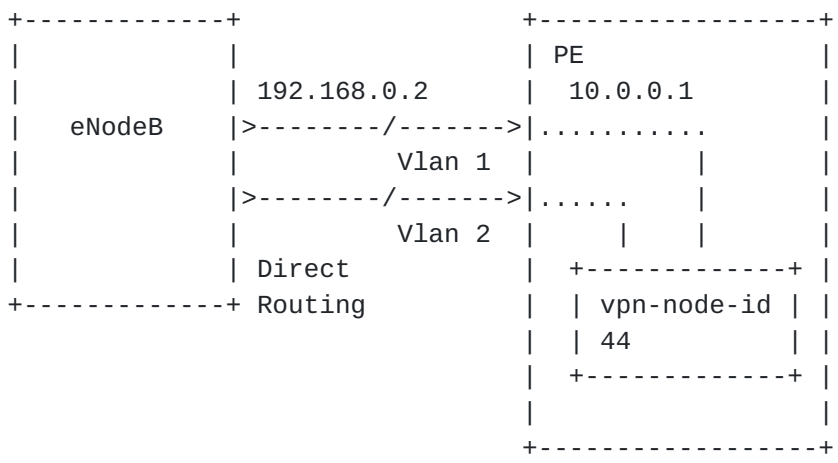

Figure 16: Mobile Backhaul Example

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

First: Create the 4G VPN Service (Figure 17).

```
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 17: Create VPN Service

Second: Create a VPN Node as depicted in Figure 18. In this type of service, the VPN Node is equivalent to the VRF configured in the physical device ('ne-id'=10.0.0.1).

```
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": "10.0.0.1",
      "local-autonomous-system": "65550",
      "rd": "0:65550:1",
      "vpn-targets": {
        "vpn-target": [
          "id": "1",
          "route-targets": ["route-target": "0:65550:1"],
          "route-target-type": "both"
        }
      }
    ]
  }
}
```

Figure 18: 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 differentiante the traffic between: Sync and data (Figure 19).

```
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",
        "status": {"admin-enabled": "true"},
        "vpn-network-access-type": "l3vpn-svc:point-to-point",
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "l3vpn-svc:static-address",
            "static-addresses": {
              "primary-address": "1",
              "address": [
```

```
                  "address-id": "1",
                  "provider-address": "192.168.0.1",
                  "customer-address": "192.168.0.1",
                  "prefix-length": "32"
                ]
              }
            }
          },
          "routing-protocols": {
            "routing-protocol": [
              "id": "1",
              "type": "l3vpn-svc:direct"
            ]
          }
        },
        {
          "vpn-network-access-id": "1/1/1.2",
          "port-id": "1/1/1",
          "description": "Interface DATA to eNODE-B",
          "status": {"admin-enabled": "true"},
          "ip-connection": {
            "ipv4": {
              "static-addresses": {
                "primary-address": "1",
                "address": [
                  "address-id": "1",
                  "provider-address": "192.168.1.1",
                  "customer-address": "192.168.1.2",
                  "prefix-length": "32"
                ]
              }
            }
          },
          "routing-protocols": {
            "routing-protocol": [
              "id": "1",
              "type": "l3vpn-svc:direct"
            ]
          }
        }
      ]
    }
  }
```

Figure 19: Create VPN Network Access

9.2. 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.

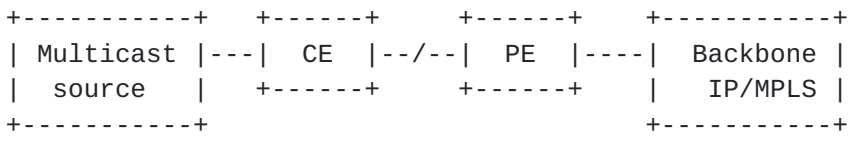

Figure 20: Multicast L3VPN Service Example

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 21)

```
"vpn-services": {
      "vpn-service": {
        "vpn-id": "Multicast_IPTV",
        "customer-name": "310",
        "vpn-service-topology": "hub-spoke",
        "description": "Multicast IPTV VPN service"
      }
  }
```

Figure 21: 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 22). In this example, the VPN Node will represent VRF configured in the physical device.

```
            "vpn-node": [
              "vpn-node-id": "500003105",
              "ne-id": "10.250.2.202",
              "autonomous-system": "3816",
              "description": "VRF_IPTV_MULTICAST",
              "router-id": "10.250.2.202",
              "address-family": "ipv4",
              "node-role": {
                "l3vpn-svc:hub-role"
              },
              "rd": "3816:31050202",
              "multicast": {
                "enabled": "true",
                "rp": {
                  "rp-group-mappings": {
                    "rp-group-mapping": {
                      "id": "1",
                      "rp-address": "172.19.48.17",
                      "groups": {
                        "group": {
                          "id": "1",
                          "group-address": "239.130.0.0/15"
                        }
                      }
                    }
                  },
                  "rp-discovery": {
                    "rp-discovery-type": {
                      "l3vpn-svc:static-rp"
                    }
                  }
                }
              }
            ]
```

Figure 22: 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 23)

```
          "vpn-network-access": {
            "vpn-network-access-id": "1/1/1",
            "description": "Connected_to_source",
            "status": { "admin-enabled": "true" },
            "vpn-network-access-type": {
              "l3vpn-svc:point-to-point"
            },
```

```
            "ip-connection": {
              "ipv4": {
                "address-allocation-type": {
                  "l3vpn-svc:static-address"
                },
                "static-addresses": {
                  "primary-address": "1",
                  "address": {
                    "address-id": "1",
                    "provider-address": "172.19.48.1",
                    "prefix-length": "30"
                  }
                }
              }
            },
            "routing-protocols": {
              "routing-protocol": {
                "id": "1",
                "type": {
                  "l3vpn-svc:bgp"
                },
                "bgp": {
                  "peer-autonomous-system": "6500",
                  "local-autonomous-system": "3816",
                  "address-family": "ipv4",
                  "neighbor": "172.19.48.2",
                  "description": "Connected_to_CE"
                }
              }
            },
            "service": {
              "multicast": {
                "multicast-site-type": "source-only",
                "multicast-address-family": { "ipv4": "true" },
                "protocol-type": "router"
              }
            }
          }
```

Figure 23: Create VPN Network Access (Excerpt of the Message Request Body)

10. L3NM YANG Module

```
<CODE BEGINS>  file "ietf-l3vpn-ntw@2020-03.09.yang"
module ietf-l3vpn-ntw {
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw";
  prefix l3vpn-ntw;

  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-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-l3vpn-svc {
    prefix l3vpn-svc;
    reference
      "RFC 8299: YANG Data Model for L3VPN Service Delivery";
  }
  import ietf-packet-fields {
    prefix packet-fields;
    reference
      "RFC 8519: YANG Data Model for Network Access
                 Control Lists (ACLs)";
  }

  organization
    "IETF OPSA (Operations and Management Area) Working Group ";
  contact
    "WG Web:   <http://tools.ietf.org/wg/opsawg/>
        WG List:  <mailto:opsawg@ietf.org>
        Author:    Samier Barguil
                  <mailto:samier.barguilgiraldo.ext@telefonica.com>
        Editor:    Oscar Gonzalez de Dios
                  <mailto:oscar.gonzalezdedios@telefonica.com>
        Author:   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) 2020 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.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.";

  revision 2020-03-09 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A Layer 3 VPN Network YANG Model";
  }

  /* Features */

  feature msdp {
    description
      "This feature indicates that msdp capabilities
       are supported by the VPN.";
  }

  feature rtg-isis {
    description
      "This features indicates the support of the ISIS
       routing protocol.";
  }

  feature rtg-ospf-sham-link {
    description
      "This feature indicates the support of OSPF sham links.";
  }

  feature input-bw {
    description
```

```
      "This feature indicates the support of
       the 'input-bw' limit.";
  }

  feature dot1q {
    description
      "This feature indicates the support of
       the 'dot1q' encapsulation.";
  }

  feature qinq {
    description
      "This feature indicates the support of
       the 'qinq' encapsulation.";
  }

  feature qinany {
    description
      "This feature indicates the support of
       the 'qinany' encapsulation.";
  }

  feature vxlan {
    description
      "This feature indicates the support of
       the 'vxlan' encapsulation.";
  }

  /* Typedefs */

  typedef protocols-type {
    type enumeration {
      enum GRE {
        value 0;
        description
          "Transport based on GRE.";
      }
      enum LDP {
        value 1;
        description
          "Transport based on LDP.";
      }
      enum BGP {
        value 2;
        description
          "Transport based on BGP.";
      }
      enum SR {
```

```
        value 3;
        description
          "Transport based on Segment Routing (SR)";
      }
      enum SR-TE {
        value 4;
        description
          "Transport based on SR for Traffic Engineering.";
      }
      enum RSVP-TE {
        value 5;
        description
          "Transport based on RSVP for Traffic Engineering";
      }
      enum unknown {
        value 6;
        description
          "Transport UNKNOWN";
      }
    }
    description
      "These attributes are used to identify underlying
       protocols when activating an L3VPN service.";
  }

  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.";
  }

  typedef isis-level {
    type enumeration {
      enum level1 {
        value 0;
        description
          "ISIS level 1";
      }
      enum level2 {
        value 1;
        description
          "ISIS level 2";
      }
      enum level1-2 {
        value 2;
        description
```

```
          "ISIS level 1 and 2";
      }
    }
    description
      "Defines the ISIS level for interface and system.";
  }

  typedef ie-type {
    type enumeration {
      enum import {
        value 0;
        description
          "Import a routing profile.";
      }
      enum export {
        value 1;
        description
          "Export a routing profile.";
      }
      enum both {
        value 2;
        description
          "Import/Export a routing profile.";
      }
    }
    description
      "Defines Import-Export routing profiles.
       Those profiles can be reused between VPN nodes.";
  }

  typedef operational-type {
    type enumeration {
      enum up {
        value 0;
        description
          "Operational status UP/Enabled.";
      }
      enum down {
        value 1;
        description
          "Operational status DOWN/Disabled.";
      }
      enum unknown {
        value 2;
        description
          "Operational status UNKNOWN.";
      }
    }
```

```
    description
      "This attribute is used to determine the
       status of a particular element.";
  }

  /* Identities */

  identity vpn-topology {
    description
      "Base identity for VPN topology.";
  }

  identity any-to-any {
    base vpn-topology;
    description
      "Identity for any-to-any VPN topology.";
  }

  identity hub-spoke {
    base vpn-topology;
    description
      "Identity for Hub-and-Spoke VPN topology.";
  }

  identity hub-spoke-disjoint {
    base vpn-topology;
    description
      "Identity for Hub-and-Spoke VPN topology
       where Hubs cannot communicate with each other.";
  }

  identity custom {
    base vpn-topology;
    description
      "Identity for CUSTOM VPN topology
       where Hubs can act as Spoke for certain part of
       the network or Spokes as Hubs.";
  }

  identity isis {
    base l3vpn-svc:routing-protocol-type;
    description
      "Identity for ISIS protocol type.";
  }

  identity pseudowire {
    base l3vpn-svc:site-network-access-type;
    description
```

```
      "Identity for pseudowire connections.";
  }

  identity loopback {
    base l3vpn-svc:site-network-access-type;
    description
      "Identity for loopback connections.";
  }

  identity encapsulation-type {
    description
      "Identity for the encapsulation type.";
  }

  identity untagged-int {
    base encapsulation-type;
    description
      "Identity for Ethernet type.";
  }

  identity tagged-int {
    base encapsulation-type;
    description
      "Identity for the VLAN type.";
  }

  identity eth-inf-type {
    description
      "Identity of the Ethernet interface type.";
  }

  identity tagged {
    base eth-inf-type;
    description
      "Identity of the tagged interface type.";
  }

  identity untagged {
    base eth-inf-type;
    description
      "Identity of the untagged interface type.";
  }

  identity lag {
    base eth-inf-type;
    description
      "Identity of the LAG interface type.";
  }
```

```
  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 priority-tagged interface.";
  }

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

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

  identity qinq {
    base tagged-inf-type;
    description
      "Identity for the QinQ tagged interface.";
  }

  identity dot1q {
    base tagged-inf-type;
    description
      "Identity for the dot1Q VLAN tagged interface.";
  }

  identity qinany {
    base tagged-inf-type;
    description
      "Identity for the QinAny tagged interface.";
  }

  identity vxlan {
    base tagged-inf-type;
```

```
    description
      "Identity for the VXLAN tagged interface.";
  }

  identity tag-type {
    description
      "Base identity from which all tag types are derived.";
  }

  identity c-vlan {
    base tag-type;
    description
      "A CVLAN tag, normally using the 0x8100 Ethertype.";
  }

  identity s-vlan {
    base tag-type;
    description
      "An SVLAN tag.";
  }

  identity c-s-vlan {
    base tag-type;
    description
      "Using both a CVLAN tag and an SVLAN tag.";
  }

  identity vxlan-peer-mode {
    description
      "Base identity for the VXLAN peer mode.";
  }

  identity static-mode {
    base vxlan-peer-mode;
    description
      "Identity for VXLAN access in the static mode.";
  }

  identity bgp-mode {
    base vxlan-peer-mode;
    description
      "Identity for VXLAN access using BGP EVPN.";
  }

  identity bw-direction {
    description
      "Identity for the bandwidth direction.";
  }
```

```
  identity input-bw {
    base bw-direction;
    description
      "Identity for the input bandwidth.";
  }

  identity output-bw {
    base bw-direction;
    description
      "Identity for the output bandwidth.";
  }

  identity bw-type {
    description
      "Identity of the bandwidth type.";
  }

  identity bw-per-cos {
    base bw-type;
    description
      "Bandwidth is per Class of Service (CoS).";
  }

  identity bw-per-port {
    base bw-type;
    description
      "Bandwidth is per site network access.";
  }

  identity bw-per-site {
    base bw-type;
    description
      "Bandwidth is per site.  It is applicable to
       all the site network accesses within a site.";
  }

  identity bw-per-svc {
    base bw-type;
    description
      "Bandwidth is per VPN service.";
  }

  /* Groupings */

  grouping svc-transport-encapsulation {
    container underlay-transport {
      leaf type {
        type protocols-type;
```

```
        description
          "Protocols used to deliver an L3VPN service.";
      }
      description
        "";
    }
    description
      "";
  }

  grouping multicast-rp-group-cfg {
    choice group-format {
      mandatory true;
      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.";
        }
      }
      description
        "Choice for multicast group format.";
    }
    description
      "This grouping defines multicast group or
       multicast groups for RP-to-group mapping.";
  }

  grouping vpn-service-multicast {
    container multicast {
      if-feature "l3vpn-svc:multicast";
      leaf enabled {
        type boolean;
        default "false";
```

```
        description
          "Enables multicast.";
      }
      leaf-list tree-flavor {
        type identityref {
          base l3vpn-svc:multicast-tree-type;
        }
        description
          "Type of tree to be used.";
      }
      container rp {
        container rp-group-mappings {
          list rp-group-mapping {
            key "id";
            leaf id {
              type uint16;
              description
                "Unique identifier for the mapping.";
            }
            container provider-managed {
              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.";
                }
                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.";
                }
                description
                  "PIM Anycast-RP parameters.";
              }
              description
                "Parameters for a provider-managed RP.";
            }
            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 {
              list group {
                key "id";
                leaf id {
                  type uint16;
                  description
                    "Identifier for the group.";
                }
                uses multicast-rp-group-cfg;
                description
                  "List of multicast groups.";
              }
              description
```

```
              "Multicast groups associated with the RP.";
            }
            description
              "List of RP-to-group mappings.";
          }
          description
            "RP-to-group mappings parameters.";
        }
        container rp-discovery {
          leaf rp-discovery-type {
            type identityref {
              base l3vpn-svc:multicast-rp-discovery-type;
            }
            default "l3vpn-svc:static-rp";
            description
              "Type of RP discovery used.";
          }
          container bsr-candidates {
            when "derived-from-or-self(../rp-discovery-type, "
               + "'l3vpn-ntw:bsr-rp')" {
              description
                "Only applicable if discovery type
                 is BSR-RP.";
            }
            leaf-list bsr-candidate-address {
              type inet:ip-address;
              description
                "Address of candidate Bootstrap Router (BSR).";
            }
            description
              "Container for List of Customer
               BSR candidate's addresses.";
          }
          description
            "RP discovery parameters.";
        }
        description
          "RP parameters.";
      }
      container msdp {
        if-feature "msdp";
        leaf enabled {
          type boolean;
          default "false";
          description
            "If true, Multicast Source Discovery Protocol (MSDP)
             protocol is activated.";
        }
```

```
        leaf peer {
          type inet:ip-address;
          description
            "IP address of the MSDP peer.";
        }
        leaf local-address {
          type inet:ip-address;
          description
            "IP address of the local end. This local address
             must be configured on the node.";
        }
        description
          "MSDP parameters.";
      }
      description
        "Multicast global parameters for the VPN service.";
    }
    description
      "Grouping for multicast VPN definition.";
  }

  grouping vpn-service-mpls {
    leaf carrierscarrier {
      if-feature "l3vpn-svc:carrierscarrier";
      type boolean;
      default "false";
      description
        "The VPN is using CsC, and so MPLS is required.";
    }
    description
      "Grouping for MPLS Carriers'Carrier definition.";
  }

  grouping operational-requirements {
    leaf requested-site-start {
      type yang:date-and-time;
      description
        "Optional leaf indicating requested date and
         time when the service at a particular site is
         expected to start.";
    }
    leaf requested-site-stop {
      type yang:date-and-time;
      description
        "Optional leaf indicating requested date and
         time when the service at a particular site is
         expected to stop.";
    }
```

```
    description
      "This grouping defines some operational
       parameters.";
  }

  grouping status-timestamp {
    leaf status {
      type operational-type;
      description
        "Operations status";
    }
    leaf timestamp {
      type yang:date-and-time;
      description
        "Indicates the actual date and time when
         the service actually started (UP) or
         stopped (DOWN).";
    }
    description
      "This grouping defines some operational
       parameters for the service.";
  }

  grouping service-status {
    container service-status {
      container admin {
        uses status-timestamp;
        description
          "Administrative service status.";
      }
      container ops {
        config false;
        uses status-timestamp;
        description
          "Operational service status.";
      }
      description
        "Service status.";
    }
    description
      "Service status grouping. Reused in
       vpn-node and vpn-network-access.";
  }

  grouping site-service-basic {
    leaf svc-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 svc-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 svc-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 'svc-mtu' leaf will refer to the
         MPLS MTU and not to the IP MTU.";
    }
    description
      "Defines basic service parameters for a site.";
  }

  grouping site-protection {
    container traffic-protection {
      if-feature "l3vpn-svc:fast-reroute";
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enables traffic protection of access link.";
      }
      description
        "Fast Reroute service parameters for the site.";
    }
    description
      "Defines protection service parameters for a site.";
  }

  grouping site-service-mpls {
    container carrierscarrier {
      if-feature "l3vpn-svc:carrierscarrier";
```

```
      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.";
      }
      description
        "This container is used when the customer provides
         MPLS-based services.  This is only used in the case
         of CsC (i.e., a customer builds an MPLS service using
         an IP VPN to carry its traffic).";
    }
    description
      "Defines MPLS service parameters for a site.";
  }

  grouping ports {
    choice source-port {
      container source-port-range-or-operator {
        uses packet-fields:port-range-or-operator;
        description
          "Source port definition.";
      }
      description
        "Choice of specifying the source port or referring to
         a group of source port numbers.";
    }
    choice destination-port {
      container destination-port-range-or-operator {
        uses packet-fields:port-range-or-operator;
        description
```

```
          "Destination port definition.";
      }
      description
        "Choice of specifying a destination port or referring
         to a group of destination port numbers.";
    }
    description
      "Choice of specifying a source or destination port numbers.";
  }

  grouping site-service-qos-profile {
    container qos {
      if-feature "l3vpn-svc:qos";
      container qos-classification-policy {
        list rule {
          key "id";
          ordered-by user;
          leaf id {
            type string;
            description
              "A description identifying the
               qos-classification-policy rule.";
          }
          choice match-type {
            default "match-flow";
            case match-flow {
              //uses l3vpn-svc:flow-definition;
              choice l3 {
                container ipv4 {
                  uses packet-fields:acl-ip-header-fields;
                  uses packet-fields:acl-ipv4-header-fields;
                  description
                    "Rule set that matches IPv4 header.";
                }
                container ipv6 {
                  uses packet-fields:acl-ip-header-fields;
                  uses packet-fields:acl-ipv6-header-fields;
                  description
                    "Rule set that matches IPv6 header.";
                }
                description
                  "Either IPv4 or IPv6.";
              }
              choice l4 {
                container tcp {
                  uses packet-fields:acl-tcp-header-fields;
                  uses ports;
                  description
```

```
                  "Rule set that matches TCP header.";
              }
              container udp {
                uses packet-fields:acl-udp-header-fields;
                uses ports;
                description
                  "Rule set that matches UDP header.";
              }
              description
                "Can be TCP or UDP";
            }
          }
          case match-application {
            leaf match-application {
              type identityref {
                base l3vpn-svc:customer-application;
              }
              description
                "Defines the application to match.";
            }
          }
          description
            "Choice for classification.";
        }
        leaf target-class-id {
          type string;
          description
            "Identification of the class of service.
             This identifier is internal to the administration.";
        }
        description
          "List of marking rules.";
      }
      description
        "Configuration of the traffic classification policy.";
    }
    container qos-profile {
      choice qos-profile {
        description
          "Choice for QoS profile.
           Can be standard profile or customized profile.";
        case standard {
          description
            "Standard QoS 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 l3vpn-svc:qos-profile-direction;
            }
            default "l3vpn-svc:both";
            description
              "The direction to which the QoS profile
               is applied.";
          }
        }
        case custom {
          description
            "Customized QoS profile.";
          container classes {
            if-feature "l3vpn-svc:qos-custom";
            list class {
              key "class-id";
              leaf class-id {
                type string;
                description
                  "Identification of the class of service.
                   This identifier is internal to the
                   administration.";
              }
              leaf direction {
                type identityref {
                  base l3vpn-svc:qos-profile-direction;
                }
                default "l3vpn-svc:both";
                description
                  "The direction to which the QoS profile
                   is applied.";
              }
              leaf rate-limit {
                type decimal64 {
                  fraction-digits 5;
                  range "0..100";
                }
                units "percent";
                description
                  "To be used if the class must be rate-limited.
                   Expressed as percentage of the service
                   bandwidth.";
```

```
                }
                container latency {
                  choice flavor {
                    case lowest {
                      leaf use-lowest-latency {
                        type empty;
                        description
                          "The traffic class should
                           use the path with the
                           lowest latency.";
                      }
                    }
                    case boundary {
                      leaf jitter-boundary {
                        type uint16;
                        units "msec";
                        default "400";
                        description
                          "The traffic class
                           should use a path with a
                           defined maximum latency.";
                      }
                    }
                    description
                      "Latency constraint
                       on the traffic class.";
                  }
                  description
                    "Latency constraint
                     on the traffic class.";
                }
                container jitter {
                  choice flavor {
                    case lowest {
                      leaf use-lowest-jitter {
                        type empty;
                        description
                          "The traffic class
                           should use the path with the
                           lowest jitter.";
                      }
                    }
                    case boundary {
                      leaf latency-boundary {
                        type uint32;
                        units "usec";
                        default "40000";
                        description
```

```
                        "The traffic class
                         should use a path with a
                         defined maximum jitter.";
                      }
                    }
                    description
                      "Jitter constraint on the traffic class.";
                  }
                  description
                    "Jitter constraint on the traffic class.";
                }
                container bandwidth {
                  leaf guaranteed-bw-percent {
                    type decimal64 {
                      fraction-digits 5;
                      range "0..100";
                    }
                    units "percent";
                    mandatory true;
                    description
                      "To be used to define the guaranteed bandwidth
                       as a percentage of the available service
                       bandwidth.";
                  }
                  leaf end-to-end {
                    type empty;
                    description
                      "Used if the bandwidth reservation
                       must be done on the MPLS network too.";
                  }
                  description
                    "Bandwidth constraint on the traffic class.";
                }
                description
                  "List of classes of services.";
              }
              description
                "Container for list of classes of services.";
            }
          }
        }
        description
          "QoS profile configuration.";
      }
      description
        "QoS configuration.";
    }
    description
```

```
      "This grouping defines QoS parameters for a site.";
  }

  grouping site-security-authentication {
    container authentication {
      description
        "Authentication parameters.";
    }
    description
      "This grouping defines authentication parameters
       for a site.";
  }

  grouping site-security-encryption {
    container encryption {
      if-feature "l3vpn-svc: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
            "Require a value for layer when enabled
             is true.";
        }
        type enumeration {
          enum layer2 {
            description
              "Encryption will occur at Layer 2.";
          }
          enum layer3 {
            description
              "Encryption will occur at Layer 3.
               For example, IPsec may be used when
               a customer requests Layer 3 encryption.";
          }
        }
        description
          "Layer on which encryption is applied.";
      }
      description
        "";
    }
    container encryption-profile {
```

```
      choice 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 SP profile to be applied.";
          }
        }
        case customer-profile {
          leaf algorithm {
            type string;
            description
              "Encryption algorithm to be used.";
          }
        }
        description
          "";
      }
      choice key-type {
        default "psk";
        case psk {
          leaf preshared-key {
            type string;
            description
              "Pre-Shared Key (PSK) coming from the customer.";
          }
        }
        description
          "Choice of encryption profile.
           The encryption profile can be the provider profile
           or customer profile.";
      }
      description
        "This grouping defines encryption parameters for
         a site.";
    }
    description
      "";
  }

  grouping site-routing {
    container routing-protocols {
      list routing-protocol {
        key "id";
```

```
        leaf id {
          type string;
          description
            "";
        }
        leaf type {
          type identityref {
            base l3vpn-svc:routing-protocol-type;
          }
          description
            "Type of routing protocol.";
        }
        list routing-profiles {
          key "id";
          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 ie-type;
            description
              "Import, export or both.";
          }
          description
            "Import or Export profile reference";
        }
        container ospf {
          when "derived-from-or-self(../type, 'l3vpn-ntw:ospf')" {
            description
              "Only applies when protocol is OSPF.";
          }
          if-feature "l3vpn-svc:rtg-ospf";
          leaf-list address-family {
            type l3vpn-svc:address-family;
            min-elements 1;
            description
              "If OSPF is used on this site, this node
               contains a configured value.  This node
               contains at least one address family
               to be activated.";
          }
          leaf area-address {
            type yang:dotted-quad;
```

```
            mandatory true;
            description
              "Area address.";
          }
          leaf metric {
            type uint16;
            default "1";
            description
              "Metric of the PE-CE link.  It is used
               in the routing state calculation and
               path selection.";
          }
          /* Extension */
          leaf mtu {
            type uint16;
            description
              "Maximum transmission unit for a given
               OSPF link.";
          }
          leaf process-id {
            type uint16;
            description
              "Process id of the OSPF CE-PE connection.";
          }
          uses security-params;
          /* End of Extension */
          container sham-links {
            if-feature "rtg-ospf-sham-link";
            list sham-link {
              key "target-site";
              leaf target-site {
                type l3vpn-svc:svc-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.";
              }
              description
                "Creates a sham link with another site.";
            }
```

```
            description
              "List of sham links.";
          }
          description
            "OSPF-specific configuration.";
        }
        container bgp {
          when "derived-from-or-self(../type, 'l3vpn-ntw:bgp')" {
            description
              "Only applies when protocol is BGP.";
          }
          if-feature "l3vpn-svc:rtg-bgp";
          leaf peer-autonomous-system {
            type inet:as-number;
            mandatory true;
            description
              "Customer AS number in case the customer
               requests BGP routing.";
          }
          leaf local-autonomous-system {
            type inet:as-number;
            description
              "Local-AS overwrite.";
          }
          leaf-list address-family {
            type l3vpn-svc:address-family;
            min-elements 1;
            description
              "If BGP is used on this site, this node
               contains a configured value.  This node
               contains at least one address family
               to be activated.";
          }
          /*  Extension  */
          leaf-list neighbor {
            type inet:ip-address;
            description
              "IP address(es) of the BGP neighbor. An 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 hops allowed between
               a given BGP neighbor and the PE router.";
          }
          uses security-params;
```

```
          uses status-params;
          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.";
          }
          /* End- Extension  */
          description
            "BGP-specific configuration.";
        }
        container isis {
          when "derived-from-or-self(../type, 'l3vpn-ntw:isis')" {
            description
              "Only applies when protocol is ISIS.";
          }
          if-feature "rtg-isis";
          leaf-list address-family {
            type l3vpn-svc:address-family;
            min-elements 1;
            description
              "If ISIS is used on this site, this node
               contains a configured value.  This node
               contains at least one address family
               to be activated.";
          }
          leaf area-address {
            type area-address;
            mandatory true;
            description
              "Area address.";
          }
          leaf level {
            type isis-level;
            description
              "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 process-id {
```

```
            type uint16;
            description
              "Process id of the ISIS CE-PE connection.";
          }
          leaf mode {
            type enumeration {
              enum active {
                description
                  "Interface sends or receives ISIS protocol
                   control packets.";
              }
              enum passive {
                description
                  "Suppresses the sending of ISIS routing updates
                   through the specified interface.";
              }
            }
            default "active";
            description
              "ISIS interface mode type.";
          }
          uses status-params;
          description
            "ISIS-specific configuration.";
        }
        container static {
          when "derived-from-or-self(../type, 'l3vpn-ntw:static')" {
            description
              "Only applies when protocol is static.
               BGP activation requires the SP to know
               the address of the customer peer.  When
               BGP is enabled, the 'static-address'
               allocation type for the IP connection
               MUST be used.";
          }
          container cascaded-lan-prefixes {
            list ipv4-lan-prefixes {
              if-feature "l3vpn-svc:ipv4";
              key "lan next-hop";
              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 inet:ipv4-address;
              description
                "Next-hop address to use on the customer side.";
            }
            description
              "List of LAN prefixes for the site.";
          }
          list ipv6-lan-prefixes {
            if-feature "l3vpn-svc:ipv6";
            key "lan next-hop";
            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 inet:ipv6-address;
              description
                "Next-hop address to use on the customer side.";
            }
            description
              "List of LAN prefixes for the site.";
          }
          description
            "LAN prefixes from the customer.";
        }
        description
          "Configuration specific to static routing.";
      }
      container rip {
        when "derived-from-or-self(../type, 'l3vpn-ntw:rip')" {
          description
            "Only applies when the protocol is RIP.  For IPv4,
             the model assumes that RIP version 2 is used.";
        }
        if-feature "l3vpn-svc:rtg-rip";
        leaf-list address-family {
          type l3vpn-svc:address-family;
          min-elements 1;
          description
            "If RIP is used on this site, this node
```

```
              contains a configured value.  This node
              contains at least one address family
              to be activated.";
          }
          description
            "Configuration specific to RIP routing.";
        }
        container vrrp {
          when "derived-from-or-self(../type, 'l3vpn-ntw:vrrp')" {
            description
              "Only applies when protocol is VRRP.";
          }
          if-feature "l3vpn-svc:rtg-vrrp";
          leaf-list address-family {
            type l3vpn-svc:address-family;
            min-elements 1;
            description
              "If VRRP is used on this site, this node
               contains a configured value.  This node contains
               at least one address family to be activated.";
          }
          description
            "Configuration specific to VRRP routing.";
        }
        description
          "List of routing protocols used on
           the site.  This list can be augmented.";
      }
      description
        "Defines routing protocols.";
    }
    description
      "Grouping for routing protocols.";
  }

  grouping site-attachment-ip-connection {
    container ip-connection {
      container ipv4 {
        if-feature "l3vpn-svc:ipv4";
        leaf address-allocation-type {
          type identityref {
            base l3vpn-svc:address-allocation-type;
          }
          must "not(derived-from-or-self(current(), 'l3vpn-ntw:slaac')"
             + " or derived-from-or-self(current(), "
             + "'l3vpn-ntw:provider-dhcp-slaac'))" {
            error-message "SLAAC is only applicable to IPv6";
          }
```

```
          description
            "Defines how addresses are allocated.
             If there is no value for the address
             allocation type, then IPv4 is not enabled.";
        }
        container provider-dhcp {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:provider-dhcp')" {
            description
              "Only applies when addresses are allocated by DHCP.";
          }
          leaf provider-address {
            type inet:ipv4-address;
            description
              "Address of provider side.  If provider-address is not
               specified, then prefix length should not be specified
               either.  It also implies provider-dhcp allocation is
               not enabled.  If provider-address is specified, then
               the prefix length may or may not be specified.";
          }
          leaf prefix-length {
            type uint8 {
              range "0..32";
            }
            must '(../provider-address)' {
              error-message
                "If the prefix length is specified, provider-address
                 must also be specified.";
              description
                "If the prefix length is specified, provider-address
                 must also be specified.";
            }
            description
              "Subnet prefix length expressed in bits.
               If not specified, or specified as zero,
               this means the customer leaves the actual
               prefix length value to the provider.";
          }
          choice address-assign {
            default "number";
            case number {
              leaf number-of-dynamic-address {
                type uint16;
                default "1";
                description
                  "Describes the number of IP addresses
                   the customer requires.";
              }
```

```
            }
            case explicit {
              container customer-addresses {
                list address-group {
                  key "group-id";
                  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
                      "First address.";
                  }
                  leaf end-address {
                    type inet:ipv4-address;
                    description
                      "Last address.";
                  }
                  description
                    "Describes IP addresses 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.";
                }
                description
                  "Container for customer addresses is allocated by
                   DHCP.";
              }
            }
            description
              "Choice for the way to assign addresses.";
          }
          description
            "DHCP allocated addresses related parameters.";
        }
        container dhcp-relay {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:provider-dhcp-relay')" {
            description
              "Only applies when provider is required to implement
               DHCP relay function.";
          }
```

```
          leaf provider-address {
            type inet:ipv4-address;
            description
              "Address of provider side.  If provider-address is not
               specified, then prefix length should not be specified
               either.  It also implies provider-dhcp allocation is
               not enabled.  If provider-address is specified, then
               prefix length may or may not be specified.";
          }
          leaf prefix-length {
            type uint8 {
              range "0..32";
            }
            must '(../provider-address)' {
              error-message
                "If prefix length is specified, provider-address
                 must also be specified.";
              description
                "If prefix length is specified, provider-address
                 must also be specified.";
            }
            description
              "Subnet prefix length expressed in bits.  If not
               specified, or specified as zero, this means the
               customer leaves the actual prefix length value
               to the provider.";
          }
          container customer-dhcp-servers {
            leaf-list server-ip-address {
              type inet:ipv4-address;
              description
                "IP address of customer DHCP server.";
            }
            description
              "Container for list of customer DHCP servers.";
          }
          description
            "DHCP relay provided by operator.";
        }
        container static-addresses {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:static-address')" {
          description
            "Only applies when protocol allocation type is static.";
          }
          leaf primary-address {
            type leafref {
              path "/l3vpn-ntw/vpn-services/vpn-service/vpn-nodes/"
```

```
                + "vpn-node/vpn-network-accesses/vpn-network-access/"
                + "ip-connection/ipv4/static-addresses/address/"
                + "address-id";
            }
            description
              "Principal address of the connection.";
          }
          list address {
            key "address-id";
            leaf address-id {
              type string;
              description
                "IPv4 Address";
            }
            leaf provider-address {
              type inet:ipv4-address;
              description
                "IPv4 Address List of the provider side.
                 When the protocol allocation type is static,
                 the provider address must be configured.";
            }
            leaf customer-address {
              type inet:ipv4-address;
              description
                "IPv4 Address of customer side.";
            }
            leaf prefix-length {
              type uint8 {
                range "0..32";
              }
              description
                "Subnet prefix length expressed in bits.
                 It is applied to both provider-address
                 and customer-address.";
            }
            description
              "Describes IPv4 addresses used.";
          }
          description
            "Describes IPv4 addresses used.";
        }
        description
          "IPv4-specific parameters.";
      }
      container ipv6 {
        if-feature "l3vpn-svc:ipv6";
        leaf address-allocation-type {
          type identityref {
```

```
            base l3vpn-svc:address-allocation-type;
          }
          description
            "Defines how addresses are allocated.
             If there is no value for the address
             allocation type, then IPv6 is
             not enabled.";
        }
        container provider-dhcp {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:provider-dhcp') "
             + "or derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:provider-dhcp-slaac')" {
            description
              "Only applies when addresses are allocated by DHCP.";
          }
          leaf provider-address {
            type inet:ipv6-address;
            description
              "Address of the provider side.  If provider-address
               is not specified, then prefix length should not be
               specified either.  It also implies provider-dhcp
               allocation is not enabled.  If provider-address is
               specified, then prefix length may or may
               not be specified.";
          }
          leaf prefix-length {
            type uint8 {
              range "0..128";
            }
            must '(../provider-address)' {
              error-message
                "If prefix length is specified, provider-address
                 must also be specified.";
              description
                "If prefix length is specified, provider-address
                 must also be specified.";
            }
            description
              "Subnet prefix length expressed in bits.  If not
               specified, or specified as zero, this means the
               customer leaves the actual prefix length value
               to the provider.";
          }
          choice address-assign {
            default "number";
            case number {
              leaf number-of-dynamic-address {
```

```
            type uint16;
            default "1";
            description
              "Describes the number of IP addresses the customer
               requires.";
          }
        }
        case explicit {
          container customer-addresses {
            list address-group {
              key "group-id";
              leaf group-id {
                type string;
                description
                  "Group-id for the address range from
                   start-address to end-address.";
              }
              leaf start-address {
                type inet:ipv6-address;
                description
                  "First address.";
              }
              leaf end-address {
                type inet:ipv6-address;
                description
                  "Last address.";
              }
              description
                "Describes IP addresses allocated by DHCP.
                 When only start-address or only end-address
                 is present, it represents a single address.
                 When both start-addressand 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.";
            }
            description
              "Container for customer addresses allocated
               by DHCP.";
          }
        }
        description
          "Choice for the way to assign addresses.";
      }
      description
        "DHCP allocated addresses related parameters.";
```

```
        }
        container dhcp-relay {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:provider-dhcp-relay')" {
            description
              "Only applies when the provider is required
               to implement DHCP relay function.";
          }
          leaf provider-address {
            type inet:ipv6-address;
            description
              "Address of the provider side.  If provider-address
               is not specified, then prefix length should not be
               specified either.  It also implies provider-dhcp
               allocation is not enabled.  If provider address
               is specified, then prefix length may or may
               not be specified.";
          }
          leaf prefix-length {
            type uint8 {
              range "0..128";
            }
            must '(../provider-address)' {
              error-message
                "If prefix length is specified, provider-address
                 must also be specified.";
              description
                "If prefix length is specified, provider-address
                 must also be specified.";
            }
            description
              "Subnet prefix length expressed in bits.  If not
               specified, or specified as zero, this means the
               customer leaves the actual prefix length value
               to the provider.";
          }
          container 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.";
            }
            description
              "Container for list of customer DHCP servers.";
```

```
          }
          description
            "DHCP relay provided by operator.";
        }
        container static-addresses {
          when "derived-from-or-self(../address-allocation-type, "
             + "'l3vpn-ntw:static-address')" {
            description
              "Only applies when protocol allocation type is static.";
          }
          leaf primary-address {
            type leafref {
              path "/l3vpn-ntw/vpn-services/vpn-service/vpn-nodes/"
                 + "vpn-node/vpn-network-accesses/vpn-network-access/"
                 + "ip-connection/ipv6/static-addresses/address/"
                 + "address-id";
            }
            description
              "Principal address of the connection";
          }
          list address {
            key "address-id";
            leaf address-id {
              type string;
              description
                "IPv4 Address";
            }
            leaf provider-address {
              type inet:ipv6-address;
              description
                "IPv6 Address of the provider side.  When the protocol
                 allocation type is static, the provider address
                 must be configured.";
            }
            leaf customer-address {
              type inet:ipv6-address;
              description
                "The IPv6 Address of the customer side.";
            }
            leaf prefix-length {
              type uint8 {
                range "0..128";
              }
              description
                "Subnet prefix length expressed in bits.
                 It is applied to both provider-address and
                 customer-address.";
            }
```

```
            description
              "Describes IPv6 addresses used.";
          }
          description
            "IPv6-specific parameters.";
        }
        description
          "IPv6-specific parameters.";
      }
      container oam {
        container bfd {
          if-feature "l3vpn-svc:bfd";
          leaf enabled {
            type boolean;
            default "false";
            description
              "If true, BFD activation is required.";
          }
          choice holdtime {
            default "fixed";
            case fixed {
              leaf fixed-value {
                type uint32;
                units "msec";
                description
                  "Expected BFD holdtime expressed in msec.  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 {
              leaf profile-name {
                type leafref {
                  path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers/"
                     + "bfd-profile-identifier/id";
                }
                description
                  "Well-known SP profile name.  The provider can propose
                   some profiles to the customer, depending on the service
                   level the customer wants to achieve.  Profile names
                   must be communicated to the customer.";
              }
              description
                "Well-known SP profile.";
            }
            description
```

```
              "Choice for holdtime flavor.";
          }
          description
            "Container for BFD.";
        }
        description
          "Defines the Operations, Administration, and Maintenance (OAM)
           mechanisms used on the connection.  BFD is set as a fault
           detection mechanism, but the 'oam' container can easily
           be augmented by other mechanisms";
      }
      description
        "Defines connection parameters.";
    }
    description
      "This grouping defines IP connection parameters.";
  }

  grouping site-service-multicast {
    container multicast {
      if-feature "l3vpn-svc:multicast";
      leaf site-type {
        type enumeration {
          enum receiver-only {
            description
              "The site only has receivers.";
          }
          enum source-only {
            description
              "The site only has sources.";
          }
          enum source-receiver {
            description
              "The site has both sources and receivers.";
          }
        }
        default "source-receiver";
        description
          "Type of multicast site.";
      }
      container address-family {
        leaf ipv4 {
          if-feature "l3vpn-svc:ipv4";
          type boolean;
          default "false";
          description
            "Enables IPv4 multicast.";
        }
```

```
        leaf ipv6 {
          if-feature "l3vpn-svc:ipv6";
          type boolean;
          default "false";
          description
            "Enables IPv6 multicast.";
        }
        description
          "Defines protocol to carry multicast.";
      }
      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.";
      }
      description
        "Multicast parameters for the site.";
    }
    description
      "Multicast parameters for the site.";
  }
```

```
  grouping site-maximum-routes {
    container maximum-routes {
      list address-family {
        key "af";
        leaf af {
          type l3vpn-svc:address-family;
          description
            "Address family.";
        }
        leaf maximum-routes {
          type uint32;
          description
            "Maximum prefixes the VRF can accept
             for this address family.";
        }
        description
          "List of address families.";
      }
      description
        "Defines 'maximum-routes' for the VRF.";
    }
    description
      "Defines 'maximum-routes' for the site.";
  }

  grouping site-security {
    container security {
      uses site-security-authentication;
      uses site-security-encryption;
      description
        "Site-specific security parameters.";
    }
    description
      "Grouping for security parameters.";
  }

  grouping network-access-service {
    container service {
      uses site-service-basic;
      /* Extension */
      /* uses svc-bandwidth-params; */
      /* EoExt */
      uses site-service-qos-profile;
      uses site-service-mpls;
      uses site-service-multicast;
      description
        "Service parameters on the attachment.";
    }
```

```
    description
      "Grouping for service parameters.";
  }

  grouping vpn-extranet {
    container extranet-vpns {
      if-feature "l3vpn-svc:extranet-vpn";
      list extranet-vpn {
        key "vpn-id";
        leaf vpn-id {
          type l3vpn-svc:svc-id;
          description
            "Identifies the target VPN the local VPN want to access.";
        }
        leaf local-sites-role {
          type identityref {
            base l3vpn-svc:site-role;
          }
          default "l3vpn-svc:any-to-any-role";
          description
            "This describes the role of the
             local sites in the target VPN topology.  In the any-to-any VPN
             service topology, the local sites must have the same role, which
             will be 'any-to-any-role'.  In the Hub-and-Spoke VPN service
             topology or the Hub-and-Spoke disjoint VPN service topology,
             the local sites must have a Hub role or a Spoke role.";
        }
        description
          "List of extranet VPNs or target VPNs the local VPN is
           attached to.";
      }
      description
        "Container for extranet VPN configuration.";
    }
    description
      "Grouping for extranet VPN configuration.
       This provides an easy way to interconnect
       all sites from two VPNs.";
  }

  grouping vpn-profile-cfg {
    container valid-provider-identifiers {
      list cloud-identifier {
        if-feature "l3vpn-svc:cloud-access";
        key "id";
        leaf id {
          type string;
          description
```

```
            "Identification of cloud service.
             Local administration meaning.";
        }
        description
          "List for Cloud Identifiers.";
      }
      list encryption-profile-identifier {
        key "id";
        leaf id {
          type string;
          description
            "Identification of the SP encryption profile
             to be used.  Local administration meaning.";
        }
        description
          "List for encryption profile identifiers.";
      }
      list qos-profile-identifier {
        key "id";
        leaf id {
          type string;
          description
            "Identification of the QoS Profile to be used.
             Local administration meaning.";
        }
        description
          "List for QoS Profile Identifiers.";
      }
      list bfd-profile-identifier {
        key "id";
        leaf id {
          type string;
          description
            "Identification of the SP BFD Profile to be used.
             Local administration meaning.";
        }
        description
          "List for BFD Profile identifiers.";
      }
      list routing-profile-identifier {
        key "id";
        leaf id {
          type string;
          description
            "Identification of the routing Profile to be used
             by the routing-protocols within sites, vpn-
             network-accesses or vpn-nodes for refering
             vrf-import/export policies.
```

```
            This identifier has a local meaning.";
        }
        description
          "List for Routing Profile Identifiers.";
      }
      nacm:default-deny-write;
      description
        "Container for Valid Provider Identifies.";
    }
    description
      "Grouping for VPN Profile configuration.";
  }

  grouping vpn-svc-cfg {
    leaf vpn-id {
      type l3vpn-svc:svc-id;
      description
        "VPN identifier.
         This identifier has a local meaning.";
    }
    leaf l3sm-vpn-id {
      type l3vpn-svc:svc-id;
      description
        "Pointer to the L3SM service.";
    }
    leaf customer-name {
      type string;
      description
        "Name of the customer that actually uses the VPN service.
         In the case that any intermediary (e.g., Tier-2 provider
         or partner) sells the VPN service to their end user
         on behalf of the original service provider (e.g., Tier-1
         provider), the original service provider may require the
         customer name to provide smooth activation/commissioning
         and operation for the service.";
    }
    leaf vpn-service-topology {
      type identityref {
        base vpn-topology;
      }
      default "any-to-any";
      description
        "VPN service topology.";
    }
    leaf description {
      type string;
      description
        "Textual description of a VPN service.";
```

```
    }
    uses ie-profiles-params;
    uses svc-transport-encapsulation;
    uses vpn-nodes-params;
    /* uses vpn-service-multicast; */
    /* uses vpn-service-mpls; */
    /* uses vpn-extranet;*/
    description
      "Grouping for VPN service configuration.";
  }

  grouping site-network-access-top-level-cfg {
    uses status-params;
    leaf vpn-network-access-type {
      type identityref {
        base l3vpn-svc:site-network-access-type;
      }
      default "l3vpn-svc:point-to-point";
      description
        "Describes the type of connection, e.g.,
         point-to-point or multipoint.";
    }
    uses ethernet-params;
    uses site-attachment-ip-connection;
    uses site-security;
    uses site-routing;
    uses network-access-service;
    description
      "Grouping for site network access top-level configuration.";
  }

  /* Bearers in a site */

  grouping site-bearer-params {
    container site-bearers {
      list bearer {
        key "bearer-id";
        leaf bearer-id {
          type string;
          description
            "";
        }
        leaf BearerType {
          type identityref {
            base bearer-inf-type;
          }
          description
            "Request for an Bearer access type.
```

```
            Choose between port or lag connection type.";
        }
        leaf ne-id {
          type string;
          description
            "NE-id reference.";
        }
        leaf port-id {
          type string;
          description
            "Reference to the Port-id.
             The semantic of the Port-Id depends on the vendor's
             semantic. i.e ge-X/Y/Z , xe-X/Y/Z , et-X/Y/Z,AeXXX.YYY,
             aeXXX,GigabitEthernetX/Y/Z";
        }
        leaf lag-id {
          type string;
          description
            "lag-id in format id.";
        }
        description
          "Parameters used to identify each bearer";
      }
      description
        "Grouping to reuse the site bearer assigment";
    }
    description
      "Grouping to reuse the site bearer assigment";
  }

  /* UNUSED */

  grouping svc-bandwidth-params {
    container svc-bandwidth {
      if-feature "input-bw";
      list bandwidth {
        key "direction type";
        leaf direction {
          type identityref {
            base bw-direction;
          }
          description
            "Indicates the bandwidth direction.  It can be
             the bandwidth download direction from the SP to
             the site or the bandwidth upload direction from
             the site to the SP.";
        }
        leaf type {
```

```
          type identityref {
            base bw-type;
          }
          description
            "Bandwidth type.  By default, the bandwidth type
             is set to 'bw-per-cos'.";
        }
        leaf cos-id {
          when "derived-from-or-self(../type, "
             + "'l3vpn-ntw:bw-per-cos')" {
            description
              "Relevant when the bandwidth type is set to
               'bw-per-cos'.";
          }
          type uint8;
          description
            "Identifier of the CoS, indicated by DSCP or a
             CE-VLAN CoS (802.1p) value in the service frame.
             If the bandwidth type is set to 'bw-per-cos',
             the CoS ID MUST also be specified.";
        }
        leaf vpn-id {
          when "derived-from-or-self(../type, "
             + "'l3vpn-ntw:bw-per-svc')" {
            description
              "Relevant when the bandwidth type is
               set as bandwidth per VPN service.";
          }
          type l3vpn-svc:svc-id;
          description
            "Identifies the target VPN.  If the bandwidth
             type is set as bandwidth per VPN service, the
             vpn-id MUST be specified.";
        }
        leaf cir {
          type uint64;
          units "bps";
          mandatory true;
          description
            "Committed Information Rate.  The maximum number
             of bits that a port can receive or send over
             an interface in one second.";
        }
        leaf cbs {
          type uint64;
          units "bps";
          mandatory true;
          description
```

```
            "Committed Burst Size (CBS).  Controls the bursty
             nature of the traffic.  Traffic that does not
             use the configured Committed Information Rate
             (CIR) accumulates credits until the credits
             reach the configured CBS.";
        }
        leaf eir {
          type uint64;
          units "bps";
          description
            "Excess Information Rate (EIR), i.e., excess frame
             delivery allowed that is not subject to an SLA.
             The traffic rate can be limited by the EIR.";
        }
        leaf ebs {
          type uint64;
          units "bps";
          description
            "Excess Burst Size (EBS).  The bandwidth available
             for burst traffic from the EBS is subject to the
             amount of bandwidth that is accumulated during
             periods when traffic allocated by the EIR
             policy is not used.";
        }
        leaf pir {
          type uint64;
          units "bps";
          description
            "Peak Information Rate, i.e., maximum frame
             delivery allowed.  It is equal to or less
             than the sum of the CIR and the EIR.";
        }
        leaf pbs {
          type uint64;
          units "bps";
          description
            "Peak Burst Size.  It is measured in bytes per
             second.";
        }
        description
          "List of bandwidth values (e.g., per CoS,
           per vpn-id).";
      }
      description
        "From the customer site's perspective, the service
         input/output bandwidth of the connection or
         download/upload bandwidth from the SP/site
         to the site/SP.";
```

```
    }
    description
      " ";
  }

  grouping status-params {
    container status {
      leaf admin-enabled {
        type boolean;
        description
          "Administrative Status UP/DOWN";
      }
      leaf oper-status {
        type operational-type;
        config false;
        description
          "Operations status";
      }
      description
        "";
    }
    description
      "Grouping used to join operational and administrative status
       is re used in the Site Network Acess and in the VPN-Node";
  }

  /* Parameters related to vpn-nodes (VRF config.) */

  grouping vpn-nodes-params {
    container vpn-nodes {
      description
        "";
      list vpn-node {
        key "ne-id";
        leaf vpn-node-id {
          type union {
            type l3vpn-svc:svc-id;
            type uint32;
          }
          description
            "Type STRING or NUMBER Serivice-Id";
        }
        leaf local-autonomous-system {
          type inet:as-number;
          description
            "Provider AS number in case the customer
             requests BGP routing.";
        }
```

```
        leaf description {
          type string;
          description
            "Textual description of a VPN node.";
        }
        leaf ne-id {
          type string;
          description
            "";
        }
        leaf router-id {
          type inet:ip-address;
          description
            "router-id information can be ipv4/6 addresses";
        }
        leaf address-family {
          type l3vpn-svc:address-family;
          description
            "Address family used for router-id information.";
        }
        leaf node-role {
          type identityref {
            base l3vpn-svc:site-role;
          }
          default "l3vpn-svc:any-to-any-role";
          description
            "Role of the vpn-node in the IP VPN.";
        }
        uses rt-rd;
        uses status-params;
        uses net-acc;
        uses site-maximum-routes;
        uses vpn-service-multicast;
        leaf node-ie-profile {
          type leafref {
            path "/l3vpn-ntw/vpn-services/"
               + "vpn-service/ie-profiles/ie-profile/ie-profile-id";
          }
          description
            "";
        }
        description
          "";
      }
    }
    description
      "Grouping to define VRF-specific configuration.";
  }
```

```
  /* Parameters related to import and export profiles (RTs RDs.) */

  grouping ie-profiles-params {
    container ie-profiles {
      list ie-profile {
        key "ie-profile-id";
        leaf ie-profile-id {
          type string;
          description
            "";
        }
        uses rt-rd;
        description
          "";
      }
      description
        "";
    }
    description
      "Grouping to specify rules for route import and export";
  }

  grouping pseudowire-params {
    container pseudowire {
      /*leaf far-end {*/
      /*  description "IP of the remote peer of the pseudowire.";*/
      /*  type inet:ip-address;*/
      /*}*/
      leaf vcid {
        type uint32;
        description
          "PW or VC identifier.";
      }
      leaf far-end {
        type union {
          type uint32;
          type inet:ipv4-address;
        }
        description
          "SDP/Far End/LDP Neighbour reference.";
      }
      description
        "Pseudowire termination parameters";
    }
    container vpls {
      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:ipv4-address;
        }
        description
          "SDP/Far End/LDP Neighbour reference.";
      }
      description
        "Pseudowire termination parameters";
    }
    description
      "Grouping pseudowire termination parameters";
  }

  grouping security-params {
    container security {
      leaf auth-key {
        type string;
        description
          "MD5 authentication password for the connection towards the
           customer edge.";
      }
      description
        "Container for aggregating any security parameter for routing
         sessions between a PE and a CE.";
    }
    description
      "Grouping to define security parameters";
  }

  grouping ethernet-params {
    container connection {
      leaf encapsulation-type {
        type identityref {
          base encapsulation-type;
        }
        default "untagged-int";
        description
          "Encapsulation type.  By default, the
           encapsulation type is set to 'untagged'.";
```

```
      }
      container logical-interface {
        leaf peer-reference {
          type uint32;
          description
            "Specify the associated logical peer interface.";
        }
        description
          "Reference of a logical interface type.";
      }
      container tagged-interface {
        leaf type {
          type identityref {
            base tagged-inf-type;
          }
          default "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, "
             + "'l3vpn-ntw:dot1q')" {
            description
              "Only applies when the type of the tagged
               interface is 'dot1q'.";
          }
          if-feature "dot1q";
          leaf tag-type {
            type identityref {
              base tag-type;
            }
            default "c-vlan";
            description
              "Tag type.  By default, the tag type is
               'c-vlan'.";
          }
          leaf cvlan-id {
            type uint16;
            description
              "VLAN identifier.";
          }
          description
            "Tagged interface.";
        }
        container priority-tagged {
          when "derived-from-or-self(../type, "
```

```
           + "'l3vpn-ntw:priority-tagged')" {
          description
            "Only applies when the type of the tagged
             interface is 'priority-tagged'.";
        }
        leaf tag-type {
          type identityref {
            base tag-type;
          }
          default "c-vlan";
          description
            "Tag type.  By default, the tag type is
             'c-vlan'.";
        }
        description
          "Priority tagged.";
      }
      container qinq {
        when "derived-from-or-self(../type, "
           + "'l3vpn-ntw:qinq')" {
          description
            "Only applies when the type of the tagged
             interface is 'qinq'.";
        }
        if-feature "qinq";
        leaf tag-type {
          type identityref {
            base tag-type;
          }
          default "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.";
        }
        description
          "QinQ.";
```

```
        }
        container qinany {
          when "derived-from-or-self(../type, "
             + "'l3vpn-ntw:qinany')" {
            description
              "Only applies when the type of the tagged
               interface is 'qinany'.";
          }
          if-feature "qinany";
          leaf tag-type {
            type identityref {
              base tag-type;
            }
            default "s-vlan";
            description
              "Tag type.  By default, the tag type is
               's-vlan'.";
          }
          leaf svlan-id {
            type uint16;
            mandatory true;
            description
              "Service VLAN ID.";
          }
          description
            "Container for QinAny.";
        }
        container vxlan {
          when "derived-from-or-self(../type, "
             + "'l3vpn-ntw:vxlan')" {
            description
              "Only applies when the type of the tagged
               interface is 'vxlan'.";
          }
          if-feature "vxlan";
          leaf vni-id {
            type uint32;
            mandatory true;
            description
              "VXLAN Network Identifier (VNI).";
          }
          leaf peer-mode {
            type identityref {
              base vxlan-peer-mode;
            }
            default "static-mode";
            description
              "Specifies the VXLAN access mode.  By default,
```

```
              the peer mode is set to 'static-mode'.";
          }
          list peer-list {
            key "peer-ip";
            leaf peer-ip {
              type inet:ip-address;
              description
                "Peer IP.";
            }
            description
              "List of peer IP addresses.";
          }
          description
            "QinQ.";
        }
        description
          "Container for tagged interfaces.";
      }
      container bearer {
        leaf bearer-reference {
          if-feature "l3vpn-svc:bearer-reference";
          type string;
          description
            "This is an internal reference for the SP.";
        }
        uses pseudowire-params;
        description
          "Defines physical properties of a site attachment.";
      }
      description
        "Encapsulation types";
    }
    description
      "Grouping to define encapsulation types";
  }

  grouping rt-rd {
    leaf rd {
      type rt-types:route-distinguisher;
      description
        "";
    }
    container vpn-targets {
      description
        "Set of route-targets to match for import and export routes
         to/from VRF";
      //uses rt-types:vpn-route-targets;
      uses vpn-route-targets;
```

```
    }
    description
      "";
  }

  grouping vpn-route-targets {
    description
      "A grouping that specifies Route Target import-export rules
       used in a BGP-enabled VPN.";
    list vpn-target {
      key "id";
      leaf id {
        type int8;
        description
          "Identifies each VPN Target";
      }
      list route-targets {
        key "route-target";
        leaf route-target {
          type rt-types:route-target;
          description
            "Route Target value";
        }
        description
          "List of Route Targets.";
      }
      leaf route-target-type {
        type rt-types:route-target-type;
        mandatory true;
        description
          "Import/export type of the Route Target.";
      }
      description
        "l3vpn route targets. AND/OR Operations are available
         based on the RTs assigment";
    }
    reference
      "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs)
       RFC4664: Framework for Layer 2 Virtual Private Networks
       (L2VPNs)";
    container vpn-policies {
      description
        "";
      leaf import-policy {
        type leafref {
          path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers/"
             + "routing-profile-identifier/id";
        }
```

```
        description
          "Reference to a VRF import policy.";
      }
      leaf export-policy {
        type leafref {
          path "/l3vpn-ntw/vpn-profiles/valid-provider-identifiers/"
             + "routing-profile-identifier/id";
        }
        description
          "Reference to a VRF export policy.";
      }
    }
  }

  grouping net-acc {
    container vpn-network-accesses {
      list vpn-network-access {
        key "id";
        leaf id {
          type l3vpn-svc:svc-id;
          description
            "Identifier for the access.";
        }
        leaf port-id {
          type l3vpn-svc:svc-id;
          description
            "Identifier for the network access.";
        }
        leaf description {
          type string;
          description
            "Textual description of a VPN service.";
        }
        uses site-network-access-top-level-cfg;
        description
          "List of accesses for a site.";
      }
      description
        "List of accesses for a site.";
    }
    description
      "Main block of the Network Access.";
  }

  /* Main Blocks */

  container l3vpn-ntw {
    container vpn-profiles {
```

```
      uses vpn-profile-cfg;
      description
        "Container for VPN Profiles.";
    }
    container vpn-services {
      list vpn-service {
        key "vpn-id";
        uses service-status;
        uses vpn-svc-cfg;
        description
          "List of VPN services.";
      }
      description
        "Top-level container for the VPN services.";
    }
    description
      "Main container for L3VPN service configuration.";
  }
}
<CODE ENDS>
```

Figure 24

11. 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

12. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040] . The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [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 L3 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 L3 VPN Service in a Service Provider Network.

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

- vpn-service: An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L3 VPN 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 L3 VPN Service. Such activity can be detected by monitoring and tracking network configuration changes.

- COMPLETE rest of critical data nodes and subtrees

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

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

Summing up, the foreseen risks of using the l3vpn-ntw module can be clasified into:

- Malicious clients attempting to delete or modify services
- Unauthorized clients attempting to create/modify/delete a service
- Unauthorized clients attempting to read service information

13. Acknowledgements

Thanks to Adrian Farrel and Miguel Cros for the suggestions on the document. Thanks to Philip Eardlay for the review. Lots of thanks for the discussions on opsawg mailing list and at IETF meeting.

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

14. Contributors

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

Daniel King
Old Dog Consulting
Email: daniel@olddog.co.uk

Daniel Voyer
Bell Canada
Email: daniel.voyer@bell.ca

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

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

Stephane Litkowski
Cisco
Email: slitkows@cisco.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>

15. References

15.1. Normative References

[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>.

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

[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>.

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

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

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

[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>.

15.2. 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-08 (work in progress), February 2020.

[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-00 (work in progress), October 2019.

[I-D.liu-pim-yang]
Liu, Y., Guo, F., and M. Sivakumar, "YANG Data Model for PIM", draft-liu-pim-yang-01 (work in progress), March 2015.

[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>.

[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>.

[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>.

[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>.

[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>.

Appendix A. Implementation Status

A.1. Nokia Implementation

Nokia has a draft implementation of the IETF L3NM model.

The implementation is a prototype and is currently being planned for production.

Nokia NSP (Network Services Platform) supports integration of standard models with the Intent Manager framework. NSP platform provides hot pluggable model definitions and implementations which would enable defining models where standardization is in progress or non-existent. With pluggable architecture for model and implementation injections, NSP also serves as a Multi-Layer, Multi-Domain controller.

The Nokia implementation of L3NM covers, the following

a) RESTConf support

b) Configuration of L3 IP VPN Services. Create/Get/Query/Delete supported on the following operations.

* Site

* Site-Bearer

* VpnService

* IEProfile

* VpnNode

* Site Network Access

* Site Attachments

c) Supports translations to the Device Model (Standard / Properietary)

draft-ietf-opsawg-l3sm-l3nm-00

The current implementation is proprietary, so under no terms the current implementation can be used.

Contact information: Sriram Krishnamurthy (sriram.krishnamurthy@nokia.com)

A.2. Huawei Implementation

The organization responsible for the implementation, if any.

Huawei Technologies Co.,Ltd.

The implementation's name and/or a link to a web page where the implementation or a description of it can be found.

NCE V1R19C00

A brief general description.

This section provides an implementation report summary for Layer 3 VPN Network Model. Layer 3 VPN Network Model is available at: https://tools.ietf.org/html/draft-ietf-opsawg-l3sm-l3nm-00

The implementation's level of maturity: research, prototype, alpha, beta, production, widely used, etc.

Right now, the data model is still subject to change, therefore it is still a Prototype, not put into production yet.

Coverage: which parts of the protocol specification are implemented.

We have implemented pruned L3NM model with the following parameters

```
module: ietf-l3vpn-ntw
+--rw l3vpn-ntw
   +--rw vpn-profiles
   |  +--rw valid-provider-identifiers
   |     +--rw qos-profile-identifier* [id]
   |     |  +--rw id    string
   +--rw vpn-services
   |  +--rw vpn-service* [vpn-id]
   |     +--rw vpn-id                  svc-id
   |     +--rw vpn-service-topology?   identityref
   |     +--rw description?            string
   |     +--rw vpn-nodes
   |     |  +--rw vpn-node* [vpn-node-id ne-id]
   |     |     +--rw vpn-node-id        string
   |     |     +--rw description?       string
   |     |     +--rw ne-id              string
   |     |     +--rw node-role?         identityref
   |     |     +--rw rd?                rt-types:route-distinguisher
   |     |     +--rw vpn-targets
   |     |     +--rw maximum-routes
   |     |     |  +--rw address-family* [af]
   |     |     |     +--rw af                address-family
   |     |     |     +--rw maximum-routes?   uint32
   +--rw sites
      +--rw site* [site-id]
         +--rw site-id                  svc-id
         +--rw locations
         |  +--rw location* [location-id]
         |     +--rw location-id     svc-id
         +--rw site-bearers
```

```
|  +--rw bearer* [bearer-id]
|     +--rw bearer-id    string
|     +--rw ne-id?       string
|     +--rw port-id?     string
+--rw site-network-accesses
   +--rw site-network-access* [site-network-access-id]
      +--rw site-network-access-id      svc-id
      +--rw site-network-access-type?   ref
      +--rw bearer
      |  +--rw bearer-reference?   {bearer-reference}?
      |  +--rw connection
      |  |  +--rw encapsulation-type?   identityref
      |  |  +--rw tagged-interface
      |  |     +--rw type?                identityref
      |  |     +--rw dot1q-vlan-tagged {dot1q}?
      |  |     |  +--rw cvlan-id    uint16
      |  |     +--rw qinq {qinq}?
      |  |     |  +--rw svlan-id    uint16
      |  |     |  +--rw cvlan-id    uint16
      +--rw ip-connection
      |  +--rw ipv4 {ipv4}?
      |  |  +--rw dhcp-relay
      |  |  |  +--rw customer-dhcp-servers
      |  |  |     +--rw server-ip-address*   inet
      |  |  +--rw addresses
      |  |     +--rw provider-address?   inet:ipv4-address
      |  |     +--rw customer-address?   inet:ipv4-address
      |  |     +--rw prefix-length?      uint8
      +--rw service
      |  +--rw qos {qos}?
      |  |  +--rw qos-profile
      |  |     +--rw (qos-profile)?
      |  |        +--:(standard)
      |  |           |  +--rw profile?   leafreaf
      +--rw routing-protocols
      |  +--rw routing-protocol* [type]
      |     +--rw type                identityref
      |     +--rw ospf {rtg-ospf}?
      |     |  +--rw address-family*   address-family
      |     |  +--rw area-address      yang:dotted-quad
      |     |  +--rw metric?           uint16
      |     |  +--rw security
      |     |  |  +--rw auth-key?   string
      |     +--rw bgp {rtg-bgp}?
      |     |  +--rw autonomous-system    uint32
      |     |  +--rw address-family*      address-family
      |     |  +--rw neighbor?            inet:ip-address
      |     |  +--rw multihop?            uint8
```

```
|     |  +--rw security
|     |     +--rw auth-key?   string
|     +--rw static
|     |  +--rw cascaded-lan-prefixes
|     |     +--rw ipv4-lan-prefixes*  {ipv4}?
|     |     |  +--rw lan         inet:ipv4-prefix
|     |     |  +--rw lan-tag?    string
|     |     |  +--rw next-hop    inet:ipv4-address
+--rw node-id?                    leafreaf
+--rw service-id?                 leafreaf
+--rw access-group-id?            yang:uuid
```

Figure 25

Use Cases we have implemented include:

(a).Create VPN

(b).Create Site

(c).Create/add bearers to an existing Site

(d).Create/Include Site Network Access into VPN nodes.

Version compatibility: what version/versions of the Internet-Draft are known to be implemented.

draft-ietf-opsawg-l3sm-l3nm-00

Licensing: the terms under which the implementation can be used. For example: proprietary, royalty licensing, freely distributable with acknowledgement (BSD style), freely distributable with requirement to redistribute source (General Public License (GPL) style), and other (specify).

Not available yet.

Implementation experience: any useful information the implementers want to share with the community.

Contact information: ideally a person's name and email address, but possibly just a URL or mailing list.

Qin Wu (bill.wu@huawei.com)

The date when information about this particular implementation was last updated.

2019-09-30

List other implementations that have been tested for interoperability.

Nokia

A.3. Infinera Implementation

Infinera has a draft implementation of the IETF L3NM model. The implementation is in beta state and is currently being tested and integrated with other suppliers controllers supporting this same model. Infinera is supporting the L3NM model in its Transcend Maestro Multi-layer, Multi-domain Controller.

The Infinera implementation of L3NM covers discovery and configuration of IP VPN services, and is supporting both North-Bound (server) and South-Bound (client) functionality. Versions 01 and 02 of the model are supported.

The current implementation is proprietary, so under no terms the current implementation can be used.

Contact information: Janne Karvonen (JKarvonen@infinera.com)

26 October is the date when information about this particular implementation was last updated.

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
Orange
FR

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