

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: March 27, 2020

A. Aguado
Nokia
O. Gonzalez de Dios, Ed.
V. Lopez
Telefonica
D. Voyer
Bell Canada
L. Munoz
Vodafone
September 24, 2019

Layer 3 VPN Network Model
draft-aguado-opsawg-l3sm-l3nm-02

Abstract

[RFC8299](#) defines a L3VPN Service YANG data Model (L3SM) that can be used for communication between customers and network operators. Such model is adequate for the customer to network operator conversation and plays the role of a Customer Service Model, according to the terminology defined in [RFC8309](#).

There is a need for a YANG model to be used in the communication between the entity that interacts directly with the customer, the service orchestrator, (either fully automated or a human operator) and the entity in charge of network orchestration and control (aka network controller / orchestrator).

This document proposes a L3VPN Network Yang Model (L3NM) to facilitate communication between a service orchestrator and a network controller / orchestrator. The resulting model is called the L3VPN Network Model (L3NM) and provides a network-centric view of the L3VPN services.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 27, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- [1. INTRODUCTION](#) [3](#)
- [1.1. TERMINOLOGY](#) [3](#)
- [1.2. Requirements Language](#) [3](#)
- [2. REFERENCE ARCHITECTURE](#) [4](#)
- [3. YANG MODEL EXPLANATION](#) [6](#)
- [3.1. STRUCTURE OF THE MODEL](#) [7](#)
- [3.2. SITE AND BEARERS](#) [7](#)
- [3.3. BEARER AND ETHERNET ENCAPSULATION](#) [7](#)
- [3.4. MULTI-DOMAIN RESOURCE MANAGEMENT](#) [7](#)
- [3.5. REMOTE FAR-END CONFIGURATION](#) [8](#)
- [3.6. PROVIDE EDGE IDENTIFICATION POINT](#) [8](#)
- [4. DESIGN OF THE DATA MODEL](#) [9](#)
- [5. YANG MODULE](#) [20](#)
- [6. IANA CONSIDERATIONS](#) [93](#)
- [7. SECURITY CONSIDERATIONS](#) [93](#)
- [8. IMPLEMENTATION STATUS](#) [93](#)
- [9. ACKNOWLEDGEMENTS](#) [94](#)
- [10. CONTRIBUTORS](#) [94](#)
- [11. References](#) [94](#)
- [11.1. NORMATIVE REFERENCES](#) [94](#)
- [11.2. INFORMATIVE REFERENCES](#) [94](#)
- Authors' Addresses [95](#)

1. INTRODUCTION

[RFC8299] defines a L3VPN Service YANG data Model (L3SM) model that can be used for communication between customers and network operators. Such model is focused on describing the customer view of the services, and provides an abstracted view of the customer's requested services. That approach limits the usage of the L3SM to the role of a Customer Service Model, according to the terminology defined in [RFC8309].

The YANG data model proposed in this document is called the L3VPN Network Model (L3NM). The L3NM model is aimed at providing a network-centric view of L3 VPN Services. The model can be used to facilitate communication between the service orchestrator, and the network controller / orchestrator. It enables further capabilities, such as resource management or to serve as a multi-domain orchestration interface, where transport resources must be synchronized. The YANG module has been built with a prune and extend approach, taking as a starting points the YANG model described in [RFC8299].

Hence, this document does not obsolete, but complements, the definitions in [RFC8299]. It aims to provide a different scope for the L3SM, but 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].

1.1. TERMINOLOGY

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8299], [RFC8309], and [RFC8453] and uses terminology from those documents. Tree diagrams used in this document follow the notation defined in [RFC8340].

1.2. 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.

2. REFERENCE ARCHITECTURE

Figure 1 shows where the L3NM is used in a management stack. 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".

Note that some implementations may choose to construct a monolithic orchestration component, but this document assumes that there are many benefits for flexibility of implementation and deployment to separate the functional components, and that separation demands the existence of separate YANG models to be used between the components.

At the same time, 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".

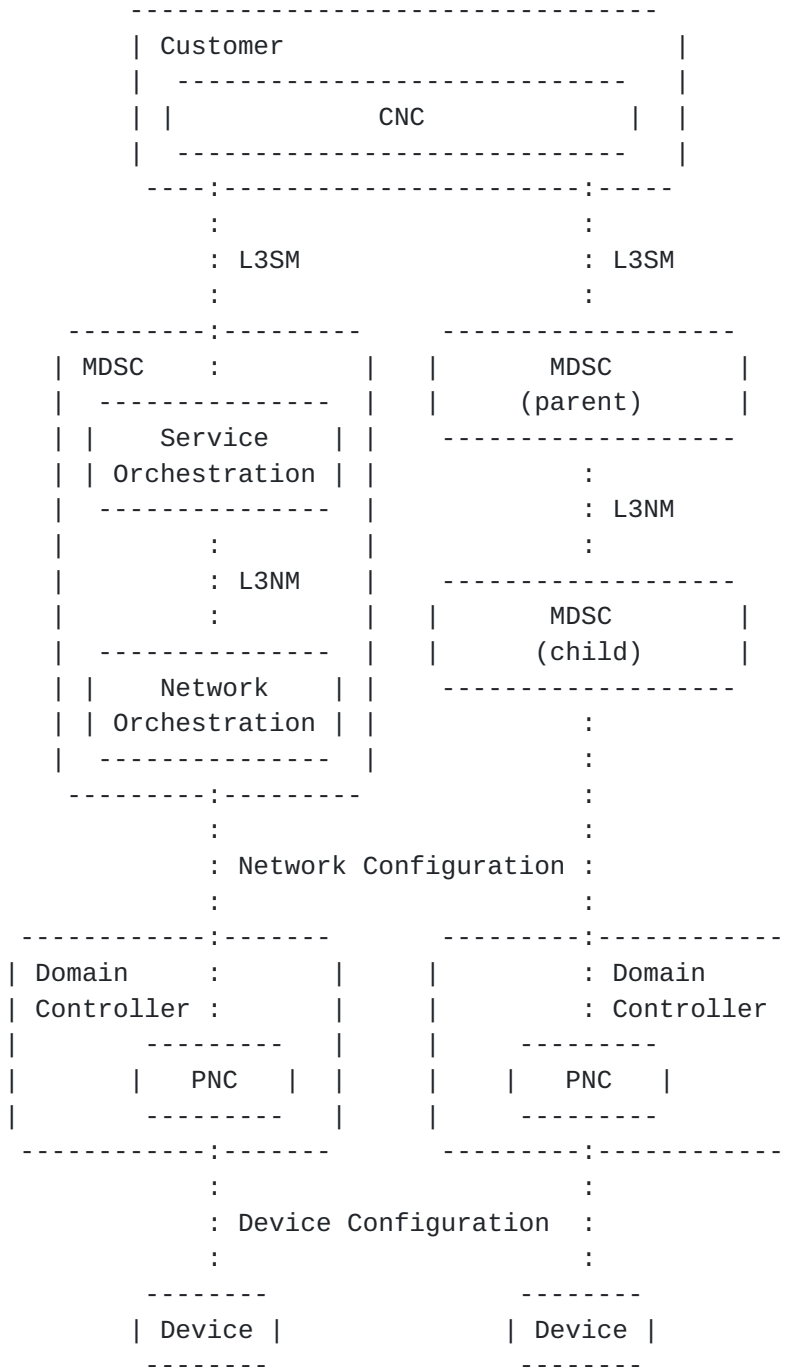


Figure 2: L3SM and L3NM in the Context of ACTN

3. YANG MODEL EXPLANATION

The scenarios covered in the L3NM model includes: the integration of Ethernet and encapsulation parameters, the extension for transport resources (e.g., Route targets and Route distinguishers) to be orchestrated from the management system, far-end configuration of PEs

not managed by the management system and the definition for PE identification.

3.1. STRUCTURE OF THE MODEL

The YANG module is divided into three main containers: "vpn-services", "sites" and "vpn-profiles".

3.2. SITE AND BEARERS

A site, as per [[RFC8299](#)], represents a connection of a customer office to one or more VPN services. As this YANG module, is the network view, each 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 (e.g., by the service orchestrator). The bearer is associated to a network element and a port. Hence, a bearer is not just a bearer-reference, but also a true reference to a given port in the service provider network.

3.3. BEARER AND ETHERNET ENCAPSULATION

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.). This specification is not supported in [[RFC8299](#)], whilst it suggests that any extension on this direction shall be implemented via augmentation of the bearer container. The extension defined to cope with these parameters uses the connection container inside the site-network-access defined by the [[RFC8466](#)]. This container defines protocol parameters to enable connectivity at Layer 2. In the context of L3SM, the augmentation includes only mandatory parameters for the service configuration, which are mainly related to the interface encapsulation. Other definitions from L2SM connection container are left aside. For example, Link Aggregation (LAG) information is not required and it shall be configured prior to the service configuration, being the aggregated interface identified in the model as the bearer-reference, as discussed later in [Section 3.4](#).

3.4. 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.

3.5. REMOTE FAR-END CONFIGURATION

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.

3.6. PROVIDE EDGE IDENTIFICATION POINT

[RFC8299] states that the "bearer-reference" parameter is used in cases where the customer has already ordered a network connection to the service provider (SP) apart from the IP VPN site and wants to reuse this connection. The string used is an internal reference from the SP and describes the already-available connection. Usually, a client interface (either a customer one or an interface used by the SP) is already in place and connected, although it has not being use previously. In some other cases (e.g., for stitching purposes), the termination of a VPN service is done over logical terminations within a PE router.

The bearer-reference must serve as a strict unequivocal parameters to identify the connection between a PE and a client (CE). This means that, despite the type is maintained as a string and there is no restriction in the way this data is formed, the bearer-reference must serve as the unique way to identify the PE router and the client interface. This, together with the encapsulation augments proposed in [Section 3.2](#), serves as the way to identify the client interface and configure L2 specific parameters.

4. DESING OF THE DATA MODEL

The augmentations defined in this document are organised per scenario, as defined in [Section 3](#). The case described [Section 3.4](#) does not need any further extension of the data model and only requires a more restricted definition on how the data model is used for PE router and client port identification, so no augmentation is implemented for this scenario.

The augmentations implemented are distributed as follows:

- o An extension including RT and RD definition for the L3VPN, following the YANG definitions from BESS-L3VPN. This extension was developed creating a container "ie-profiles" under the VPN Service. All the import-export information can be created and reused for several VPN-Nodes.
 - * If the "ie-profile" is empty the domain controller should automatically assing RD and RTs. This is not valid for a multi-domain scenario
- o The second augmentation copes with the information from a remote PE not directly under management system supervision. This augmentation does not follow any previously defined model and includes the loopback IP address of the external router.
- o The third augmentation copes with a pseudowire termination under a VPN service. This termination requires the management of the Virtual Circuit Identifier under the VPN service.
- o Access-group-id has been added within the site network access in order to allow associations between interfaces that have similar behaviors. For example, identify two interfaces in dual homing distribution.
- o The last augmentation includes information below layer 3 that is required for the service. In particular, we include information related to clients interface encapsulation and aggregation.

The high-level model structure defined by this document is as shown below:

```

|----- EXAMPLE -----|
module: ietf-l3vpn-ntw
  +--rw l3vpn-ntw
    +--rw vpn-profiles
      | +--rw valid-provider-identifiers

```



```

|     +--rw cloud-identifier* [id] {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 vpn-id          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* [route-target]
|     |   |   |   |   +--rw route-target          rt-types:route-
target
|     |   |   |   |   +--rw route-target-type    rt-types:route-
target-type
|     |   +--rw vpn-nodes
|     |   |   +--rw vpn-node* [vpn-node-id ne-id]
|     |   |   |   +--rw vpn-node-id  string
|     |   |   |   +--rw description?  string
|     |   |   |   +--rw ne-id        string
|     |   |   |   +--rw router-id?   inet:ip-address
|     |   |   |   +--rw address-family?  address-family
|     |   |   |   +--rw node-role?    identityref
|     |   |   |   +--rw rd?          rt-types:route-
distinguisher
|     |   |   +--rw vpn-targets
|     |   |   |   +--rw vpn-target* [route-target]
|     |   |   |   |   +--rw route-target          rt-types:route-
target
|     |   |   |   |   +--rw route-target-type    rt-types:route-
target-type
|     |   +--rw status
|     |   |   +--rw admin-enabled?  boolean
|     |   |   +--ro oper-status?    operational-type
|     |   +--rw maximum-routes
|     |   |   +--rw address-family* [af]
|     |   |   |   +--rw af          address-family

```

```
      |      |      |      +--rw maximum-routes?   uint32
      |      |      |      +--rw node-ie-profile?   -> /l3vpn-ntw/vpn-services/
vpn-service/ie-profiles/ie-profile/ie-profile-id
      |      +--rw cloud-accesses {cloud-access}?
      |      |      +--rw cloud-access* [cloud-identifier]
      |      |      +--rw cloud-identifier           -> /l3vpn-ntw/vpn-
profiles/valid-provider-identifiers/cloud-identifier/id
```

```

| | +--rw (list-flavor)?
| | | +--:(permit-any)
| | | | +--rw permit-any? empty
| | | +--:(deny-any-except)
| | | | +--rw permit-site* -> /l3vpn-ntw/sites/
site/site-id
| | | +--:(permit-any-except)
| | | +--rw deny-site* -> /l3vpn-ntw/sites/
site/site-id
| | +--rw address-translation
| | +--rw nat44
| | +--rw enabled? boolean
| | +--rw nat44-customer-address? inet:ipv4-
address
| +--rw multicast {multicast}?
| | +--rw enabled? boolean
| | +--rw customer-tree-flavors
| | | +--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 rp-address inet:ip-address
| | | +--rw groups
| | | | +--rw group* [id]
| | | | | +--rw id uint16
| | | | | +--rw (group-format)
| | | | | +--:(singleaddress)
| | | | | | +--rw group-address? inet:ip-
address
| | | | +--:(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 carrierscarrier? boolean {carrierscarrier}?
+--rw extranet-vpns {extranet-vpn}?
| +--rw extranet-vpn* [vpn-id]
| | +--rw vpn-id svc-id
| | +--rw local-sites-role? identityref

```

```
+--rw sites
  +--rw site* [site-id]
    +--rw site-id          svc-id
    +--rw description?    string
    +--rw requested-site-start? yang:date-and-time
    +--rw requested-site-stop? yang:date-and-time
```



```
| | | | +--:(match-flow)
| | | | | +--rw match-flow
```

```

| | | | | +-rw dscp?          inet:dscp
| | | | | +-rw dot1p?         uint8
| | | | | +-rw ipv4-src-prefix? inet:ipv4-
prefix
| | | | | +-rw ipv6-src-prefix? inet:ipv6-
prefix
| | | | | +-rw ipv4-dst-prefix? inet:ipv4-
prefix
| | | | | +-rw ipv6-dst-prefix? inet:ipv6-
prefix
| | | | | +-rw l4-src-port?     inet:port-
number
| | | | | +-rw target-sites*   svc-id
{target-sites}?
| | | | | +-rw l4-src-port-range
| | | | | | +-rw lower-port?   inet:port-
number
| | | | | | +-rw upper-port?  inet:port-
number
| | | | | +-rw l4-dst-port?     inet:port-
number
| | | | | +-rw l4-dst-port-range
| | | | | | +-rw lower-port?   inet:port-
number
| | | | | | +-rw upper-port?  inet:port-
number
| | | | | +-rw protocol-field? union
| | | | | +--:(match-application)
| | | | | | +-rw match-application? identityref
| | | | | | +-rw target-class-id?  string
| | | | | +-rw qos-profile
| | | | | | +-rw (qos-profile)?
| | | | | | +--:(standard)
| | | | | | | +-rw profile?      -> /l3vpn-ntw/vpn-
profiles/valid-provider-identifiers/qos-profile-identifier/id
| | | | | | | +-rw direction?   identityref
| | | | | | | +--:(custom)
| | | | | | | +-rw classes {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 multicast-site-type?          enumeration
|     +--rw multicast-address-family
|     |   +--rw ipv4?    boolean {ipv4}?
|     |   +--rw ipv6?    boolean {ipv6}?
|     +--rw protocol-type?                enumeration
+--rw traffic-protection {fast-reroute}?
| +--rw enabled?    boolean
+--rw routing-protocols
| +--rw routing-protocol* [type]
|   +--rw type          identityref
|   +--rw routing-profiles* [id]
|     | +--rw id        -> /l3vpn-ntw/vpn-profiles/valid-
provider-identifiers/routing-profile-identifier/id
|     | +--rw type?    ie-type
|     +--rw ospf {rtg-ospf}?
|     |   +--rw address-family*    address-family
|     |   +--rw area-address        yang:dotted-quad
|     |   +--rw metric?            uint16
|     |   +--rw mtu?              uint16
|     |   +--rw security
|     |     | +--rw auth-key?    string
|     |     +--rw sham-links {rtg-ospf-sham-link}?
|     |       +--rw sham-link* [target-site]
|     |         +--rw target-site  svc-id
|     |         +--rw metric?     uint16
|     +--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* [lan next-hop]
{ipv4}?
|     |   |   +--rw lan          inet:ipv4-prefix
|     |   |   +--rw lan-tag?    string
|     |   |   +--rw next-hop    inet:ipv4-address
|     |     +--rw ipv6-lan-prefixes* [lan next-hop]
{ipv6}?
|     |       +--rw lan          inet:ipv6-prefix
|     |       +--rw lan-tag?    string
|     |       +--rw next-hop    inet:ipv6-address
+--rw rip {rtg-rip}?
|   +--rw address-family*    address-family
+--rw vrrp {rtg-vrrp}?
|   +--rw address-family*    address-family

```

```
+-ro actual-site-start?      yang:date-and-time
+-ro actual-site-stop?      yang:date-and-time
+rw site-bearers
```

```

| +--rw bearer* [bearer-id]
|   +--rw bearer-id      string
|   +--rw BearerType?   identityref
|   +--rw ne-id?        string
|   +--rw port-id?      string
|   +--rw lag-id?       string
+--rw site-network-accesses
  +--rw site-network-access* [site-network-access-id]
    +--rw site-network-access-id      svc-id
    +--rw description?                string
    +--rw status
    | +--rw admin-enabled?  boolean
    | +--ro oper-status?   operational-type
    +--rw site-network-access-type?  identityref
    +--rw (location-flavor)
    | +--:(location)
    | | +--rw location-reference?  -> ../../../../
locations/location/location-id
    | +--:(device)
    | +--rw device-reference?      -> ../../../../
devices/device/device-id
  +--rw access-diversity {site-diversity}?
  | +--rw groups
  | | +--rw group* [group-id]
  | |   +--rw group-id  string
  | +--rw constraints
  |   +--rw constraint* [constraint-type]
  |   +--rw constraint-type  identityref
  |   +--rw target
  |   +--rw (target-flavor)?
  |   +--:(id)
  |   | +--rw group* [group-id]
  |   |   +--rw group-id  string
  |   +--:(all-accesses)
  |   | +--rw all-other-accesses?  empty
  |   +--:(all-groups)
  |   +--rw all-other-groups?      empty
+--rw bearer
| +--rw requested-type {requested-type}?
| | +--rw requested-type?  string
| | +--rw strict?          boolean
| +--rw always-on?         boolean {always-on}?
| +--rw bearer-reference?  string {bearer-
reference}?
| +--rw connection
| | +--rw encapsulation-type?  identityref
| | +--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 pseudowire
| | +--rw vcid? uint32
+--rw ip-connection
| +--rw ipv4 {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 addresses
| | | +--rw provider-address? inet:ipv4-address
| | | +--rw customer-address? inet:ipv4-address
| | | +--rw prefix-length? uint8
| +--rw ipv6 {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 addresses
| | | +--rw provider-address? inet:ipv6-address
| | | +--rw customer-address? inet:ipv6-address
| | | +--rw prefix-length? uint8
| +--rw oam
| | +--rw bfd {bfd}?
| | | +--rw enabled? boolean
| | | +--rw (holdtime)?
| | | | +--:(fixed)
| | | | | +--rw fixed-value? uint32
| | | | +--:(profile)
| | | | +--rw profile-name? -> /l3vpn-ntw/
vpn-profiles/valid-provider-identifiers/bfd-profile-identifier/id
+--rw security
| +--rw authentication
| +--rw encryption {encryption}?
| | +--rw enabled? boolean
| | +--rw layer? enumeration
| +--rw encryption-profile
| | +--rw (profile)?
| | | +--:(provider-profile)
| | | | +--rw profile-name? -> /l3vpn-ntw/
vpn-profiles/valid-provider-identifiers/encryption-profile-identifier/id
| | | +--:(customer-profile)
| | | +--rw algorithm? string
| | +--rw (key-type)?
| | | +--:(psk)
| | | +--rw preshared-key? string
+--rw service
| +--rw svc-input-bandwidth uint64
| +--rw svc-output-bandwidth uint64
| +--rw svc-mtu uint16
| +--rw qos {qos}?

```



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


				+-: (lowest)
latency?	empty			+-rw use-lowest-
				+-: (boundary)
boundary?	uint16			+-rw latency-
				+-rw jitter
				+-rw (flavor)?
				+-: (lowest)
jitter?	empty			+-rw use-lowest-
				+-: (boundary)
boundary?	uint32			+-rw latency-
				+-rw bandwidth
percent	decimal64			+-rw guaranteed-bw-
				+-rw end-to-
end?	empty			
			+-rw carrierscarrier {carrierscarrier}?	
			+-rw signalling-type? enumeration	
			+-rw multicast {multicast}?	

```

|     +--rw multicast-site-type?          enumeration
|     +--rw multicast-address-family
|     |   +--rw ipv4?    boolean {ipv4}?
|     |   +--rw ipv6?    boolean {ipv6}?
|     +--rw protocol-type?                enumeration
+--rw routing-protocols
|   +--rw routing-protocol* [type]
|     +--rw type                identityref
|     +--rw routing-profiles* [id]
|       |   +--rw id            -> /l3vpn-ntw/vpn-profiles/
valid-provider-identifiers/routing-profile-identifier/id
|       |   +--rw type?        ie-type
|     +--rw ospf {rtg-ospf}?
|       |   +--rw address-family*  address-family
|       |   +--rw area-address      yang:dotted-quad
|       |   +--rw metric?           uint16
|       |   +--rw mtu?              uint16
|       |   +--rw security
|       |     |   +--rw auth-key?   string
|       |     +--rw sham-links {rtg-ospf-sham-link}?
|       |       +--rw sham-link* [target-site]
|       |         +--rw target-site  svc-id
|       |         +--rw metric?      uint16
|     +--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* [lan next-hop]
{ipv4}?
|       |   |   +--rw lan            inet:ipv4-prefix
|       |   |   +--rw lan-tag?      string
|       |   |   +--rw next-hop      inet:ipv4-address
|       |   +--rw ipv6-lan-prefixes* [lan next-hop]
{ipv6}?
|       |   |   +--rw lan            inet:ipv6-prefix
|       |   |   +--rw lan-tag?      string
|       |   |   +--rw next-hop      inet:ipv6-address
|     +--rw rip {rtg-rip}?
|       |   +--rw address-family*    address-family
|     +--rw vrrp {rtg-vrrp}?
|       |   +--rw address-family*    address-family
+--rw availability
|   +--rw access-priority?  uint32

```

```
services/vpn-service/vpn-nodes/vpn-node/vpn-node-id    +--rw node-id?    -> /l3vpn-ntw/vpn-
services/vpn-service/vpn-id                             +--rw service-id? -> /l3vpn-ntw/vpn-
                                                         yang:uuid
services/vpn-service/vpn-id                             +--rw access-group-id?
```

Figure 3

5. YANG MODULE

|----- EXAMPLE -----|

```
<CODE BEGINS>file "ietf-l3vpn-ntw@2019-09-13.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;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }
  import ietf-routing-types {
    prefix rt-types;
  }
  organization
    "Individual draft";
  contact
    "Currently discussed in WG List: <mailto:opsawg@ietf.org>
    Editor: Oscar Gonzalez de Dios
    <mailto:oscar.gonzalezdedios@telefonica.com>";

  description
    "This YANG module defines a generic network-oriented model
    for the configuration of Layer 3 VPNs.
    Copyright (c) 2019 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.
```

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](#) ([RFC 2119](#)) ([RFC 8174](#)) when, and only when, they appear in all capitals, as shown here.";

```
revision 2019-09-24 {
  description
  "Initial document. The document as a whole is based on L3SM
  module, defined in RFC 8299, modified to fit the requirements
  of the platforms at the network layer.";
  reference
  "RFC 8049.";
}
/* Features */
feature cloud-access {
  description
  "Allows the VPN to connect to a CSP.";
}
feature multicast {
  description
  "Enables multicast capabilities in a VPN.";
}
feature ipv4 {
  description
  "Enables IPv4 support in a VPN.";
}
feature ipv6 {
  description
  "Enables IPv6 support in a VPN.";
}
feature lan-tag {
  description
  "Enables LAN Tag support in a VPN Policy filter.";
}
feature carrierscarrier {
  description
  "Enables support of CsC.";
}
feature extranet-vpn {
  description
  "Enables support of extranet VPNs.";
}
feature site-diversity {
  description
  "Enables support of site diversity constraints.";
}
feature encryption {
  description
```



```
"Enables support of encryption.";
}
feature qos {
  description
  "Enables support of classes of services.";
}
feature qos-custom {
  description
  "Enables support of the custom QoS profile.";
}
feature rtg-bgp {
  description
  "Enables support of the BGP routing protocol.";
}
feature rtg-rip {
  description
  "Enables support of the RIP routing protocol.";
}
feature rtg-ospf {
  description
  "Enables support of the OSPF routing protocol.";
}
feature rtg-ospf-sham-link {
  description
  "Enables support of OSPF sham links.";
}
feature rtg-vrrp {
  description
  "Enables support of the VRRP routing protocol.";
}
feature fast-reroute {
  description
  "Enables support of Fast Reroute.";
}
feature bfd {
  description
  "Enables support of BFD.";
}
feature always-on {
  description
  "Enables support of the 'always-on' access constraint.";
}
feature requested-type {
  description
  "Enables support of the 'requested-type' access constraint.";
}
feature bearer-reference {
  description
```



```
"Enables support of the 'bearer-reference' access constraint.";
}
feature target-sites {
  description
  "Enables support of the 'target-sites' match flow parameter.";
}
feature input-bw {
  description
  "Enables support of the 'input-bw' limit.";
}
feature dot1q {
  description
  "Enables support of the 'dot1q' encapsulation.";
}
feature qinq {
  description
  "Enables support of the 'qinq' encapsulation.";
}
feature qinany {
  description
  "Enables support of the 'qinany' encapsulation.";
}
feature vxlan {
  description
  "Enables support of the 'vxlan' encapsulation.";
}
/* Typedefs */
typedef svc-id {
  type string;
  description
  "Defines a type of service component identifier.";
}
typedef template-id {
  type string;
  description
  "Defines a type of service template identifier.";
}
typedef address-family {
  type enumeration {
    enum ipv4 {
      description
      "IPv4 address family.";
    }
    enum ipv6 {
      description
      "IPv6 address family.";
    }
  }
}
```



```
description
"Defines a type for the address family.";
}

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

typedef operational-type {
  type enumeration {
    enum "up" {
      value 0;
      description "Operational status UP.";
    }
    enum "down" {
      value 1;
      description "Operational status DOWN";
    }
    enum "unknown" {
      value 2;
      description "Operational status UNKNOWN";
    }
  }
  description
  "This is a read-only attribute used to determine the
  status of a particular element";
}

/* Identities */
identity site-network-access-type {
  description
  "Base identity for site-network-access type.";
```



```
}
identity point-to-point {
  base site-network-access-type;
  description
  "Identity for point-to-point connection.";
}
/* Extension */
identity pseudowire {
  base site-network-access-type;
  description
  "Identity for pseudowire connection.";
}
/* End of Extension */
identity multipoint {
  base site-network-access-type;
  description
  "Identity for multipoint connection.
  Example: Ethernet broadcast segment.";
}
identity placement-diversity {
  description
  "Base identity for site placement constraints.";
}
identity bearer-diverse {
  base placement-diversity;
  description
  "Identity for bearer diversity.
  The bearers should not use common elements.";
}
identity pe-diverse {
  base placement-diversity;
  description
  "Identity for PE diversity.";
}
identity pop-diverse {
  base placement-diversity;
  description
  "Identity for POP diversity.";
}
identity linecard-diverse {
  base placement-diversity;
  description
  "Identity for linecard diversity.";
}
identity same-pe {
  base placement-diversity;
  description
  "Identity for having sites connected on the same PE.";
```



```
}
identity same-bearer {
  base placement-diversity;
  description
  "Identity for having sites connected using the same bearer.";
}
identity customer-application {
  description
  "Base identity for customer application.";
}
identity web {
  base customer-application;
  description
  "Identity for Web application (e.g., HTTP, HTTPS).";
}
identity mail {
  base customer-application;
  description
  "Identity for mail application.";
}
identity file-transfer {
  base customer-application;
  description
  "Identity for file transfer application (e.g., FTP, SFTP).";
}
identity database {
  base customer-application;
  description
  "Identity for database application.";
}
identity social {
  base customer-application;
  description
  "Identity for social-network application.";
}
identity games {
  base customer-application;
  description
  "Identity for gaming application.";
}
identity p2p {
  base customer-application;
  description
  "Identity for peer-to-peer application.";
}
identity network-management {
  base customer-application;
  description
```



```
"Identity for management application
(e.g., Telnet, syslog, SNMP).";
}
identity voice {
  base customer-application;
  description
  "Identity for voice application.";
}
identity video {
  base customer-application;
  description
  "Identity for video conference application.";
}
identity embb {
  base customer-application;
  description
  "Identity for an enhanced Mobile Broadband (eMBB)
  application. Note that an eMBB application demands
  network performance with a wide variety of
  characteristics, such as data rate, latency,
  loss rate, reliability, and many other parameters.";
}
identity urllc {
  base customer-application;
  description
  "Identity for an Ultra-Reliable and Low Latency
  Communications (URLLC) application. Note that a
  URLLC application demands network performance
  with a wide variety of characteristics, such as latency,
  reliability, and many other parameters.";
}
identity mmtc {
  base customer-application;
  description
  "Identity for a massive Machine Type
  Communications (mMTC) application. Note that an
  mMTC application demands network performance
  with a wide variety of characteristics, such as data
  rate, latency, loss rate, reliability, and many
  other parameters.";
}
identity site-vpn-flavor {
  description
  "Base identity for the site VPN service flavor.";
}
identity site-vpn-flavor-single {
  base site-vpn-flavor;
  description
```



```
"Base identity for the site VPN service flavor.
Used when the site belongs to only one VPN.";
}
identity site-vpn-flavor-multi {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor.
  Used when a logical connection of a site
  belongs to multiple VPNs.";
}
identity site-vpn-flavor-sub {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor.
  Used when a site has multiple logical connections.
  Each connection may belong to different multiple VPNs.";
}
identity site-vpn-flavor-nni {
  base site-vpn-flavor;
  description
  "Base identity for the site VPN service flavor.
  Used to describe an NNI option A connection.";
}
identity management {
  description
  "Base identity for site management scheme.";
}
identity co-managed {
  base management;
  description
  "Base identity for co-managed site.";
}
identity customer-managed {
  base management;
  description
  "Base identity for customer-managed site.";
}
identity provider-managed {
  base management;
  description
  "Base identity for provider-managed site.";
}
identity address-allocation-type {
  description
  "Base identity for address-allocation-type for PE-CE link.";
}
identity provider-dhcp {
  base address-allocation-type;
```



```
    description
    "Provider network provides DHCP service to customer.";
}
identity provider-dhcp-relay {
    base address-allocation-type;
    description
    "Provider network provides DHCP relay service to customer.";
}
identity provider-dhcp-slaac {
    base address-allocation-type;
    description
    "Provider network provides DHCP service to customer,
    as well as SLAAC.";
}
identity static-address {
    base address-allocation-type;
    description
    "Provider-to-customer addressing is static.";
}
identity slaac {
    base address-allocation-type;
    description
    "Use IPv6 SLAAC.";
}
identity site-role {
    description
    "Base identity for site type.";
}
identity any-to-any-role {
    base site-role;
    description
    "Site in an any-to-any IP VPN.";
}
identity spoke-role {
    base site-role;
    description
    "Spoke site in a Hub-and-Spoke IP VPN.";
}
identity hub-role {
    base site-role;
    description
    "Hub site in a Hub-and-Spoke IP VPN.";
}
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 multicast-tree-type {
  description
    "Base identity for multicast tree type.";
}
identity ssm-tree-type {
  base multicast-tree-type;
  description
    "Identity for SSM tree type.";
}
identity asm-tree-type {
  base multicast-tree-type;
  description
    "Identity for ASM tree type.";
}
identity bidir-tree-type {
  base multicast-tree-type;
  description
    "Identity for bidirectional tree type.";
}
identity multicast-rp-discovery-type {
  description
    "Base identity for RP discovery type.";
}
identity auto-rp {
  base multicast-rp-discovery-type;
  description
    "Base identity for Auto-RP discovery type.";
}
identity static-rp {
  base multicast-rp-discovery-type;
  description
    "Base identity for static type.";
}
```



```
identity bsr-rp {
  base multicast-rp-discovery-type;
  description
  "Base identity for BSR discovery type.";
}
identity routing-protocol-type {
  description
  "Base identity for routing protocol type.";
}
identity ospf {
  base routing-protocol-type;
  description
  "Identity for OSPF protocol type.";
}
identity bgp {
  base routing-protocol-type;
  description
  "Identity for BGP protocol type.";
}
identity static {
  base routing-protocol-type;
  description
  "Identity for static routing protocol type.";
}
identity rip {
  base routing-protocol-type;
  description
  "Identity for RIP protocol type.";
}
identity vrrp {
  base routing-protocol-type;
  description
  "Identity for VRRP protocol type.
  This is to be used when LANs are directly connected
  to PE routers.";
}
identity direct {
  base routing-protocol-type;
  description
  "Identity for direct protocol type.";
}
identity protocol-type {
  description
  "Base identity for protocol field type.";
}
identity tcp {
  base protocol-type;
  description
```



```
"TCP protocol type.;"
}
identity udp {
  base protocol-type;
  description
  "UDP protocol type.;"
}

identity icmp {
  base protocol-type;
  description
  "ICMP protocol type.;"
}
identity icmp6 {
  base protocol-type;
  description
  "ICMPv6 protocol type.;"
}
identity gre {
  base protocol-type;
  description
  "GRE protocol type.;"
}
identity ipip {
  base protocol-type;
  description
  "IP-in-IP protocol type.;"
}
identity hop-by-hop {
  base protocol-type;
  description
  "Hop-by-Hop IPv6 header type.;"
}
identity routing {
  base protocol-type;
  description
  "Routing IPv6 header type.;"
}
identity esp {
  base protocol-type;
  description
  "ESP header type.;"
}
identity ah {
  base protocol-type;
  description
  "AH header type.;"
}
```



```
identity vpn-policy-filter-type {
  description
  "Base identity for VPN Policy filter type.";
}
identity ipv4 {
  base vpn-policy-filter-type;
  description
  "Identity for IPv4 Prefix filter type.";
}
identity ipv6 {
  base vpn-policy-filter-type;
  description
  "Identity for IPv6 Prefix filter type.";
}
identity lan {
  base vpn-policy-filter-type;
  description
  "Identity for LAN Tag filter type.";
}

identity qos-profile-direction {
  description
  "Base identity for QoS profile direction.";
}

identity site-to-wan {
  base qos-profile-direction;
  description
  "Identity for Site-to-WAN direction.";
}
identity wan-to-site {
  base qos-profile-direction;
  description
  "Identity for WAN-to-Site direction.";
}
identity both {
  base qos-profile-direction;
  description
  "Identity for both WAN-to-Site direction
  and Site-to-WAN direction.";
}

/* Extended Identities */

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 by BGP EVPN learning.";
}

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 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 the site.";
}

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

/* Groupings */
grouping vpn-service-cloud-access {
    container cloud-accesses {
        if-feature cloud-access;
        list cloud-access {
            key cloud-identifier;
            leaf cloud-identifier {
                type leafref {
                    path "/l3vpn-ntw/vpn-profiles/"+
                    "valid-provider-identifiers/cloud-identifier/id";
                }
            }
            description
                "Identification of cloud service.
                Local administration meaning.";
        }
        choice list-flavor {
            case permit-any {
                leaf permit-any {
                    type empty;
                    description

```



```
    "Allows all sites.";
  }
}
case deny-any-except {
  leaf-list permit-site {
    type leafref {
      path "/l3vpn-ntw/sites/site/site-id";
    }
    description
      "Site ID to be authorized.";
  }
}
case permit-any-except {
  leaf-list deny-site {
    type leafref {
      path "/l3vpn-ntw/sites/site/site-id";
    }
    description
      "Site ID to be denied.";
  }
}
description
  "Choice for cloud access policy.  By
  default, all sites in the IP VPN MUST
  be authorized to access the cloud.";
}
container address-translation {
  container nat44 {
    leaf enabled {
      type boolean;
      default false;
      description
        "Controls whether or not Network address
        translation from IPv4 to IPv4 (NAT44)
        [RFC3022] is required.";
    }
  }
  leaf nat44-customer-address {
    type inet:ipv4-address;
    description
      "Address to be used for network address
      translation from IPv4 to IPv4.  This is
      to be used if the customer is providing
      the IPv4 address.  If the customer address
      is not set, the model assumes that the
      provider will allocate the address.";
  }
}
description
  "IPv4-to-IPv4 translation.";
```



```
    }
    description
    "Container for NAT.";
  }
  description
  "Cloud access configuration.";
}
description
"Container for cloud access configurations.";
}
description
"Grouping for VPN cloud definition.";
}
grouping multicast-rp-group-cfg {
  choice group-format {
    mandatory true;
    case singleaddress {
      leaf group-address {
        type inet:ip-address;
        description
        "A single multicast group address.";
      }
    }
    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 multicast;
    leaf enabled {
```



```
    type boolean;
    default false;
    description
    "Enables multicast.";
}
container customer-tree-flavors {
  leaf-list tree-flavor {
    type identityref {
      base multicast-tree-type;
    }
    description
    "Type of tree to be used.";
  }
  description
  "Type of trees used by customer.";
}
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.";
  }
  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 multicast-rp-discovery-type;
    }
    default 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 BSR candidate.";
    }
    description
    "Container for List of Customer
    BSR candidate's addresses.";
  }
  description
  "RP discovery parameters.";
}
description
"RP parameters.";
}
description
"Multicast global parameters for the VPN service.";
}
description
"Grouping for multicast VPN definition.";
}
grouping vpn-service-mpls {
  leaf carrierscarrier {
    if-feature carrierscarrier;
    type boolean;
    default false;
    description
    "The VPN is using CsC, and so MPLS is required.";
  }
  description
  "Grouping for MPLS CsC definition.";
}
grouping customer-location-info {
  container locations {
    list location {
      key location-id;
      leaf location-id {
        type svc-id;
        description
        "Identifier for a particular location.";
      }
    }
    leaf address {
      type string;
    }
  }
}
```



```
    description
    "Address (number and street) of the site.";
  }
  leaf postal-code {
    type string;
    description
    "Postal code of the site.";
  }
  leaf state {
    type string;
    description
    "State of the site. This leaf can also be
    used to describe a region for a country that
    does not have states.";
  }
  leaf city {
    type string;
    description
    "City of the site.";
  }
  leaf country-code {
    type string {
      pattern '[A-Z]{2}';
    }
    description
    "Country of the site.
    Expressed as ISO ALPHA-2 code.";
  }
  description
  "Location of the site.";
}
description
"List of locations for the site.";
}
description
"This grouping defines customer location parameters.";
}
grouping site-group {
  container groups {
    list group {
      key group-id;
      leaf group-id {
        type string;
        description
        "Group-id the site belongs to.";
      }
    }
    description
    "List of group-ids.";
  }
}
```



```
    }
    description
    "Groups the site or site-network-access belongs to.";
  }
  description
  "Grouping definition to assign
  group-ids to site or site-network-access.";
}
grouping site-diversity {
  container site-diversity {
    if-feature site-diversity;
    uses site-group;
    description
    "Diversity constraint type. All
    site-network-accesses will inherit
    the group values defined here.";
  }
  description
  "This grouping defines site
  diversity parameters.";
}
grouping access-diversity {
  container access-diversity {
    if-feature site-diversity;
    uses site-group;
    container constraints {
      list constraint {
        key constraint-type;
        leaf constraint-type {
          type identityref {
            base placement-diversity;
          }
        }
        description
        "Diversity constraint type.";
      }
    }
    container target {
      choice target-flavor {
        default id;
        case id {
          list group {
            key group-id;
            leaf group-id {
              type string;
              description
              "The constraint will be applied against
              this particular group-id for this site
              network access level.";
            }
          }
        }
      }
    }
  }
}
```



```
        description
        "List of group-ids associated with one specific
        constraint for this site network access level.";
    }
}
case all-accesses {
    leaf all-other-accesses {
        type empty;
        description
        "The constraint will be applied against
        all other site network accesses of this site.";
    }
}
case all-groups {
    leaf all-other-groups {
        type empty;
        description
        "The constraint will be applied against
        all other groups managed by the customer.";
    }
}
description
"Choice for the target flavor definition.";
}
description
"The constraint will be applied against a
Specific target, and the target can be a list
of group-ids,all other site network accesses of
this site, or all other groups managed by the
customer.";
}
description
"List of constraints.";
}
description
"Placement constraints for this site network access.";
}
description
"Diversity parameters.";
}
description
"This grouping defines access diversity parameters.";
}
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 operational-requirements-ops {
    leaf actual-site-start {
        type yang:date-and-time;
        config false;
        description
            "Optional leaf indicating actual date and
            time when the service at a particular site
            actually started.";
    }
    leaf actual-site-stop {
        type yang:date-and-time;
        config false;
        description
            "Optional leaf indicating actual date and
            time when the service at a particular site
            actually stopped.";
    }
    description
        "This grouping defines some operational
        parameters.";
}
grouping flow-definition {
    container match-flow {
        leaf dscp {
            type inet:dscp;
            description
                "DSCP value.";
        }
        leaf dot1p {
            type uint8 {
                range "0..7";
            }
            description

```



```
    "802.1p matching.";
  }
  leaf ipv4-src-prefix {
    type inet:ipv4-prefix;
    description
      "Match on IPv4 src address.";
  }
  leaf ipv6-src-prefix {
    type inet:ipv6-prefix;
    description
      "Match on IPv6 src address.";
  }
  leaf ipv4-dst-prefix {
    type inet:ipv4-prefix;
    description
      "Match on IPv4 dst address.";
  }
  leaf ipv6-dst-prefix {
    type inet:ipv6-prefix;
    description
      "Match on IPv6 dst address.";
  }
  leaf l4-src-port {
    type inet:port-number;
    must "current() < ../l4-src-port-range/lower-port or "+
      "current() > ../l4-src-port-range/upper-port" {
      description
        "If l4-src-port and l4-src-port-range/lower-port and
        upper-port are set at the same time, l4-src-port
        should not overlap with l4-src-port-range.";
    }
    description
      "Match on Layer 4 src port.";
  }
  leaf-list target-sites {
    if-feature target-sites;
    type svc-id;
    description
      "Identify a site as traffic destination.";
  }
  container l4-src-port-range {
    leaf lower-port {
      type inet:port-number;
      description
        "Lower boundary for port.";
    }
    leaf upper-port {
      type inet:port-number;
```



```
    must ". >= ../lower-port" {
      description
      "Upper boundary for port.  If it
      exists, the upper boundary must be
      higher than the lower boundary.";
    }
  description
  "Upper boundary for port.";
}
description
"Match on Layer 4 src port range.  When
only the lower-port is present, it represents
a single port.  When both the lower-port and
upper-port are specified, it implies
a range inclusive of both values.";
}
leaf l4-dst-port {
  type inet:port-number;
  must "current() < ../l4-dst-port-range/lower-port or "+
  "current() > ../l4-dst-port-range/upper-port" {
    description
    "If l4-dst-port and l4-dst-port-range/lower-port
    and upper-port are set at the same time,
    l4-dst-port should not overlap with
    l4-src-port-range.";
  }
  description
  "Match on Layer 4 dst port.";
}
container l4-dst-port-range {
  leaf lower-port {
    type inet:port-number;
    description
    "Lower boundary for port.";
  }
  leaf upper-port {
    type inet:port-number;
    must ". >= ../lower-port" {
      description
      "Upper boundary must be
      higher than lower boundary.";
    }
    description
    "Upper boundary for port.  If it exists,
    upper boundary must be higher than lower
    boundary.";
  }
  description
```



```
"Match on Layer 4 dst port range. When only
lower-port is present, it represents a single
port. When both lower-port and upper-port are
specified, it implies a range inclusive of both
values.";
}
leaf protocol-field {
  type union {
    type uint8;
    type identityref {
      base protocol-type;
    }
  }
}
description
"Match on IPv4 protocol or IPv6 Next Header field.";
}
description
"Describes flow-matching criteria.";
}
description
"Flow definition based on criteria.";
}
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 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 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 per RFC 8277) as the signalling protocol
            between the PE and the CE.
            In this case, BGP must also be configured as
            the routing protocol.";
        }
      }
    }
    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 site-service-qos-profile {
  container qos {
    if-feature 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 flow-definition;
          }
          case match-application {
            leaf match-application {
              type identityref {
                base 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 qos-profile-direction;
        default both;
        description
        "The direction to which the QoS profile
        is applied.";
      }
    }
  }
  case custom {
    description
    "Customized QoS profile.";
    container classes {
      if-feature 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 qos-profile-direction;
          }
          default 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 latency-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 encryption;
    leaf enabled {
      type boolean;
      default false;
      description
      "If true, traffic encryption on the connection is required.";
    }
    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-attachment-bearer {
  container bearer {
    container requested-type {
      if-feature requested-type;
      leaf requested-type {
        type string;
        description
          "Type of requested bearer: Ethernet, DSL,
          Wireless, etc. Operator specific.";
      }
    }
  }
}
```



```
    }
    leaf strict {
      type boolean;
      default false;
      description
        "Defines whether requested-type is a preference
        or a strict requirement.";
    }
    description
      "Container for requested-type.";
  }
  leaf always-on {
    if-feature always-on;
    type boolean;
    default true;
    description
      "Request for an always-on access type.
      For example, this could mean no dial access type.";
  }

  leaf bearer-reference {
    if-feature bearer-reference;
    type string;
    description
      "This is an internal reference for the SP.";
  }
  description
    "Bearer-specific parameters.
    To be augmented.";

  uses ethernet-params;

  uses pseudowire-params {
    when "/l3vpn-ntw/sites/site/site-network-accesses" +
        "/site-network-access/site-network-access-type ='pseudowire'"
    {
      description "pseudowire specific parameters";
    }
  }
}
description
  "Defines physical properties of a site attachment.";
}
grouping site-routing {
  container routing-protocols {
    list routing-protocol {
      key type;
    }
  }
}
```



```
leaf type {
  type identityref {
    base 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 rtg-ospf;
  leaf-list address-family {
    type 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.";
}

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 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 rtg-bgp;
  leaf autonomous-system {
    type uint32;
    mandatory true;
    description
    "Customer AS number in case the customer
    requests BGP routing.";
  }
  leaf-list address-family {
    type 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 neighbor {
    type inet:ip-address;
    description
    "IP address of the BGP neighbor.";
  }

  leaf multihop {
    type uint8;
    description
    "Describes the number of hops allowed between the
    given BGP neighbor and the PE router.";
  }

  uses security-params;

  description
  "BGP-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 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 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 rtg-rip;
  leaf-list address-family {
    type 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 rtg-vrrp;
  leaf-list address-family {
    type 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 ipv4;
      leaf address-allocation-type {
        type identityref {
          base 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 addresses {
  when "derived-from-or-self(..address-allocation-type, "+
    "'l3vpn-ntw:static-address')" {
  description
    "Only applies when protocol allocation type is static.";
  }
  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
    "IPv4-specific parameters.";
  }
  container ipv6 {
    if-feature ipv6;
    leaf address-allocation-type {
      type identityref {
```



```
    base address-allocation-type;
  }
  description
  "Defines how addresses are allocated.
  If there is no value for the address
  allocation type, then IPv6 is
  not enabled.";
}

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-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 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 addresses {
  when "derived-from-or-self(..address-allocation-type, "+
    "'l3vpn-ntw:static-address')" {
    description
      "Only applies when protocol allocation type is static.";
  }
  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.";
}
container oam {
  container bfd {
    if-feature 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 multicast;
        leaf multicast-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 multicast-address-family {
  leaf ipv4 {
    if-feature ipv4;
    type boolean;
    default false;
    description
    "Enables IPv4 multicast.";
  }
  leaf ipv6 {
    if-feature 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.";
}
description
"Multicast parameters for the site.";
}
description
"Multicast parameters for the site.";
}
grouping site-management {
  container management {
    leaf type {
      type identityref {
        base management;
      }
      mandatory true;
      description
      "Management type of the connection.";
    }
    description
    "Management configuration.";
  }
  description
  "Management parameters for the site.";
}
grouping site-devices {
  container devices {
    when "derived-from-or-self(..management/type, "+
    "'l3vpn-ntw:provider-managed') or "+
    "derived-from-or-self(..management/type, 'l3vpn-ntw:co-managed')" {
      description
      "Applicable only for provider-managed or
      co-managed device.";
    }
  }
  list device {
    key device-id;
    leaf device-id {
      type svc-id;
      description
```



```
    "Identifier for the device.";
  }
  leaf location {
    type leafref {
      path "../.../locations/"+
        "location/location-id";
    }
    mandatory true;
    description
      "Location of the device.";
  }
  container management {
    when "derived-from-or-self(../.../management/type,"+
      "'l3vpn-ntw:co-managed')" {
      description
        "Applicable only for co-managed device.";
    }
    leaf address-family {
      type address-family;
      description
        "Address family used for management.";
    }
    leaf address {
      when "(../address-family)" {
        description
          "If address-family is specified, then address should
            also be specified.  If address-family is not specified,
            then address should also not be specified.";
      }
      type inet:ip-address;
      mandatory true;
      description
        "Management address.";
    }
    description
      "Management configuration.  Applicable only for
        co-managed device.";
  }
  description
    "List of devices requested by customer.";
}
description
  "Device configuration.";
}
description
  "Grouping for device allocation.";
}
grouping site-vpn-flavor {
```



```
leaf site-vpn-flavor {
  type identityref {
    base site-vpn-flavor;
  }
  default site-vpn-flavor-single;
  description
  "Defines the way the VPN multiplexing is done, e.g., whether
  the site belongs to a single VPN site or a multiVPN; or, in the case
  of a multiVPN, whether the logical accesses of the sites belong
  to the same set of VPNs or each logical access maps to
  different VPNs.";
}
description
"Grouping for site VPN flavor.";
}
grouping site-maximum-routes {
  container maximum-routes {
    list address-family {
      key af;
      leaf af {
        type 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 site-service {
  container service {
    uses site-service-qos-profile;
    uses site-service-mps;
    uses site-service-multicast;
    description
      "Service parameters on the attachment.";
  }
  description
    "Grouping for service parameters.";
}
grouping site-network-access-service {
  container service {
    uses site-service-basic;
    /* Extension */
    /* uses svc-bandwidth-params; */
    /* EoExt */
    uses site-service-qos-profile;
    uses site-service-mps;
    uses site-service-multicast;
    description
      "Service parameters on the attachment.";
  }
  description
    "Grouping for service parameters.";
}
grouping vpn-extranet {
  container extranet-vpns {
    if-feature extranet-vpn;
    list extranet-vpn {
      key vpn-id;
      leaf vpn-id {
        type svc-id;
        description
          "Identifies the target VPN the local VPN want to access.";
      }
      leaf local-sites-role {
        type identityref {
          base site-role;
        }
        default 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 site-attachment-availability {
  container availability {
    leaf access-priority {
      type uint32;
      default 100;
      description
      "Defines the priority for the access.
      The higher the access-priority value,
      the higher the preference of the
      access will be.";
    }
    description
    "Availability parameters (used for multihoming).";
  }
  description
  "Defines availability parameters for a site.";
}
grouping vpn-profile-cfg {
  container valid-provider-identifiers {
    list cloud-identifier {
      if-feature 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 and site-
    network-accesses. Local administration 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 svc-id;
    description
    "VPN identifier. Local administration meaning.";
  }
  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 vpn-nodes-params;
  uses vpn-service-cloud-access;
  uses vpn-service-multicast;
  uses vpn-service-mpls;
  uses vpn-extranet;
  description
  "Grouping for VPN service configuration.";
}
grouping site-top-level-cfg {
  uses operational-requirements;
  uses customer-location-info;
  uses site-devices;
```



```
uses site-diversity;
uses site-management;
uses site-vpn-flavor;
uses site-maximum-routes;
uses site-security;
uses site-service;
uses site-protection;
uses site-routing;
description
"Grouping for site top-level configuration.";
}
grouping site-network-access-top-level-cfg {

/* Extension */

uses status-params;

/* End of Extension */

leaf site-network-access-type {
  type identityref {
    base site-network-access-type;
  }
  default point-to-point;
  description
  "Describes the type of connection, e.g.,
  point-to-point or multipoint.";
}
choice location-flavor {
  case location {
    when "derived-from-or-self(..../management/type, "+
      "'l3vpn-ntw:customer-managed')" {
      description
      "Applicable only for customer-managed device.";
    }
    leaf location-reference {
      type leafref {
        path "..../locations/location/location-id";
      }
      description
      "Location of the site-network-access.";
    }
  }
}
case device {
  when "derived-from-or-self(..../management/type, "+
    "'l3vpn-ntw:provider-managed') or "+
    "derived-from-or-self(..../management/type, "+
    "'l3vpn-ntw:co-managed')" {
```



```
    description
    "Applicable only for provider-managed or co-managed device.";
  }
  leaf device-reference {
    type leafref {
      path "../../devices/device/device-id";
    }
    description
    "Identifier of CE to use.";
  }
}
mandatory true;
description
"Choice of how to describe the site's location.";
}
uses access-diversity;
uses site-attachment-bearer;
uses site-attachment-ip-connection;
uses site-security;
uses site-network-access-service;
uses site-routing;
uses site-attachment-availability;
description
"Grouping for site network access top-level configuration.";
}
```

```
/* Extensions */
```

```
/* 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
            "Port-id in format slot/ card /port.";
    }

    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 assignment";
    }
    description
        "Grouping to reuse the site bearer assignment";
    }

/* 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 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 Access and in the VPN-Node";
}

/* Parameters related to vpn-nodes (VRF config.) */
grouping vpn-nodes-params {
    container vpn-nodes {
        description "";

        list vpn-node {
            key "vpn-node-id ne-id";

            leaf vpn-node-id {
                type string;
                description "";
            }

            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 address-family;
      description
        "Address family used for router-id information.";
    }

    leaf node-role {
      type identityref {
        base site-role;
      }
      default any-to-any-role;
      description
        "Role of the vpn-node in the IP VPN.";
    }
    uses rt-rd;
    uses status-params;

    /* Here we use the name given to the existing structure in sites */
    uses site-maximum-routes;

    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.";
        }
        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 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.";
}
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;
    }
    description
        "";
}

/* 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 vpn-svc-cfg;
    description
    "List of VPN services.";
  }
  description
  "Top-level container for the VPN services.";
}
container sites {
  list site {
    key site-id;
    leaf site-id {
      type svc-id;
      description
      "Identifier of the site.";
    }
    leaf description {
      type string;
      description
      "Textual description of a site.";
    }
  }
  uses site-top-level-cfg;
  uses operational-requirements-ops;
  uses site-bearer-params;
  container site-network-accesses {
    list site-network-access {
      key site-network-access-id;
      leaf site-network-access-id {
        type svc-id;
        description
        "Identifier for the access.";
      }
      leaf description {
        type string;
        description
        "Textual description of a VPN service.";
      }
    }
    uses site-network-access-top-level-cfg;
    leaf node-id {
      type leafref{
        path "/l3vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/vpn-
node-id";
      }
      description
      "Reference the VPN node id";
    }
  }
}
```

}

Aguado, et al.

Expires March 27, 2020

[Page 92]

```
leaf service-id {
  type leafref{
    path "/l3vpn-ntw/vpn-services/vpn-service/vpn-id";
  }
  description
  "Reference the VPN node id";
}
leaf access-group-id {
  type yang:uuid;
  description
  "Reference the Access Goup ID.
  It is used to group and identify SNA with common behavior
  such as dual-homming";
}
description
"List of accesses for a site.";
}
description
"List of accesses for a site.";
}
description
"List of sites.";
}
description
"Container for sites.";
}
description
"Main container for L3VPN service configuration.";
}
}
```

Figure 4

6. IANA CONSIDERATIONS

This memo includes no request to IANA.

7. SECURITY CONSIDERATIONS

All the security considerations of [[RFC8299](#)] apply to this document. Subsequent versions will provide additional security considerations.

8. IMPLEMENTATION STATUS

This section will be used to track the status of the implementations of the model. It is aimed at being removed if the document becomes RFC.

9. ACKNOWLEDGEMENTS

Thanks to Adrian Farrel and Miguel Cros for the suggestions on the document. Thanks to Stephane Litowski and Philip Eardlay for the review. Lots of thanks for the discussions on opsawg mailing list and at IETF meeting. Some of the comments have already been incorporated and the other part of the comments will be addressed in the next versions.

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

10. CONTRIBUTORS

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

Samier Barguil
Telefonica
Email: samier.barguilgiraldo.ext@telefonica.com

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

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

11. References

11.1. 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>>.

11.2. INFORMATIVE REFERENCES

[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>>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

Authors' Addresses

Alejandro Aguado
Nokia
Madrid
ES

Email: alejandro.aguado_martin@nokia.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Victor Lopez
Telefonica
Madrid
ES

Email: victor.lopezalvarez@telefonica.com

Daniel Voyer
Bell Canada
CA

Email: daniel.voyer@bell.ca

Luis Angel Munoz
Vodafone
ES

Email: luis-angel.munoz@vodafone.com

