OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 4 December 2022

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

2 June 2022

A YANG Network Data Model for Layer 2 VPNs
draft-ietf-opsawg-l2nm-19

Abstract

This document defines an L2VPN Network YANG Model (L2NM) which can be used to manage the provisioning of Layer 2 Virtual Private Network services within a network (e.g., service provider network). The L2NM complements the Layer 2 Service Model (L2SM) by providing a network-centric view of the service that is internal to a service provider. The L2NM is particularly meant to be used by a network controller to derive the configuration information that will be sent to relevant network devices.

Also, this document defines a YANG module to manage Ethernet segments and the initial versions of two IANA-maintained modules that include a set of identities of BGP Layer 2 encapsulation types and pseudowire types.

Editorial Note (To be removed by RFC Editor)

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

* "This version of this YANG module is part of RFC XXXX;"

* "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs";

* reference: RFC XXXX

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

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 4 December 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Acronyms and Abbreviations 6
4. Reference Architecture 6
5. Relationship to Other YANG Data Models 11
6. Description of the Ethernet Segment YANG Module 12
7. Description of the L2NM YANG Module 15
7.1. Overall Structure of the Module 15
7.2. VPN Profiles 16
7.3. VPN Services 17
7.4. Global Parameters Profiles 21
7.5. VPN Nodes 25
7.5.1. BGP Auto-Discovery 28
7.5.2. Signaling Options 29
7.5.2.1. BGP 31
7.5.2.2. LDP 33
7.5.2.3. L2TP 34
7.6. VPN Network Accesses 34
7.6.1. Connection 36
7.6.2. EVPN-VPWS Service Instance 39

7.6.3. Ethernet OAM 41
7.6.4. Services 42
8. YANG Modules 48
8.1. IANA-Maintained Module for BGP Layer 2 Encapsulation Types 48
8.2. IANA-Maintained Module for Pseudowire Types 54
8.3. Ethernet Segments 61
8.4. L2NM 69
9. Security Considerations 123
10. IANA Considerations 124
10.1. Registering YANG Modules 124
10.2. BGP Layer 2 Encapsulation Types 126
10.3. Pseudowire Types 126
11. References 127
11.1. Normative References 127
11.2. Informative References 130
Appendix A. Examples 136
A.1. BGP-based VPLS 136
A.2. BGP-based VPWS with LDP Signaling 142
A.3. LDP-based VPLS 145
A.4. VPWS-EVPN Service Instance 149
A.5. Automatic ESI Assignment 155
A.6. VPN Network Access Precedence 158
Acknowledgements 161
Contributors 161
Authors' Addresses 162

1. Introduction

[RFC8466] defines an L2VPN Service Model (L2SM) YANG data model that can be used between customers and service providers for ordering Layer 2 Virtual Private Network (L2VPN) services. This document complements the L2SM by creating a network-centric view of the service: the L2VPN Network Model (L2NM).

Also, this document defines the initial versions of two IANA-maintained modules that define a set of identities of BGP Layer 2 encapsulation types (Section 8.1) and pseudowire types (Section 8.2). These types are used in the L2NM to identify a Layer 2 encapsulation type as a function of the signalling option used to deliver an L2VPN service. Relying upon these IANA-maintained modules is meant to provide more flexibility in handling new types rather than being limited by a set of identities defined in the L2NM itself.
Section 8.3 defines another YANG module to manage Ethernet Segments (ESes) that are required for instantiating Ethernet VPNs (EVPNs). References to Ethernet segments that are created using the module in Section 8.3 can be included in the L2NM for EVPNs.

The L2NM (Section 8.4) can be exposed, for example, by a network controller to a service controller within the service provider's network. In particular, the model can be used in the communication interface between the entity that interacts directly with the customer (i.e., the service orchestrator) and the entity in charge of network orchestration and control (a.k.a., network controller/ orchestrator) by allowing for more network-centric information to be included.

The L2NM supports capabilities, such as exposing operational parameters, transport protocols selection, and precedence. It can also serve as a multi-domain orchestration interface.

The L2NM is scoped for a variety of Layer 2 Virtual Private Networks, such as:

* Virtual Private LAN Service (VPLS) [RFC4761][RFC4762]
* Virtual Private Wire Service (VPWS) (Section 3.1.1 of [RFC4664])
* Various flavors of EVPNs:
  - VPWS EVPN [RFC8214],
  - Provider Backbone Bridging Ethernet VPNs (PBB EVPNs) [RFC7623],
  - EVPN over MPLS [RFC7432], and
  - EVPN over Virtual eXtensible Local Area Network (VXLAN) [RFC8365].

The L2NM is designed to easily support future Layer 2 VPN flavors and procedures (e.g., advanced configuration such as pseudowires resilience or Multi-Segment pseudowires [RFC7267]). A set of examples to illustrate the use of the L2NM are provided in Appendix A.

This document uses the common Virtual Private Network (VPN) YANG module defined in [RFC9181].

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

2. Terminology

This document assumes that the reader is familiar with [RFC6241], [RFC7950], [RFC8466], [RFC4026], and [RFC8309]. This document uses terminology from those documents.

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

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

This document makes use of the following terms:

Ethernet segment (ES): Refers to the set of the Ethernet links that are used by a customer site (device or network) to connect to one or more Provider Edges (PEs).

Layer 2 VPN Service Model (L2SM): Describes the service characterization of an L2VPN that interconnects a set of sites from the customer's perspective. The customer service model does not provide details on the service provider network. An L2VPN customer service model is defined in [RFC8466].

Layer 2 VPN Network Model (L2NM): Refers to the YANG data model that describes an L2VPN service with a network-centric view. It contains information on the service provider network and might include allocated resources. Network controllers can use it to manage the Layer 2 VPN service configuration in the service provider's network. The corresponding YANG module can be used by a service orchestrator to request a VPN service to a network controller or to expose the list of active L2VPN services. The L2NM can also be used to retrieve a set of L2VPN-related state information (including OAM).

MAC-VRF: Refers to a Virtual Routing and Forwarding (VRF) table for Media Access Control (MAC) addresses on a PE.

Network controller: Denotes a functional entity responsible for the management of the service provider network.

Service orchestrator: Refers to a functional entity that interacts with the customer of an L2VPN relying upon, e.g., the L2SM. The service orchestrator is responsible for the Customer Edge - to Provider Edge (CE-PE) attachment circuits, the PE selection, and requesting the activation of the L2VPN service to a network controller.

Service provider network: Is a network able to provide L2VPN-related services.

VPN node: Is an abstraction that represents a set of policies applied on a PE and belonging to a single VPN service. A VPN service involves one or more VPN nodes. The VPN node will identify the service providers' node on which the VPN is deployed.

VPN network access: Is an abstraction that represents the network

interfaces that are associated with a given VPN node. Traffic coming from the VPN network access belongs to the VPN. The attachment circuits (bearers) between Customer Edges (CEs) and Provider Edges (PEs) are terminated in the VPN network access.

VPN service provider: Is a service provider that offers L2VPN-related services.

3. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document:

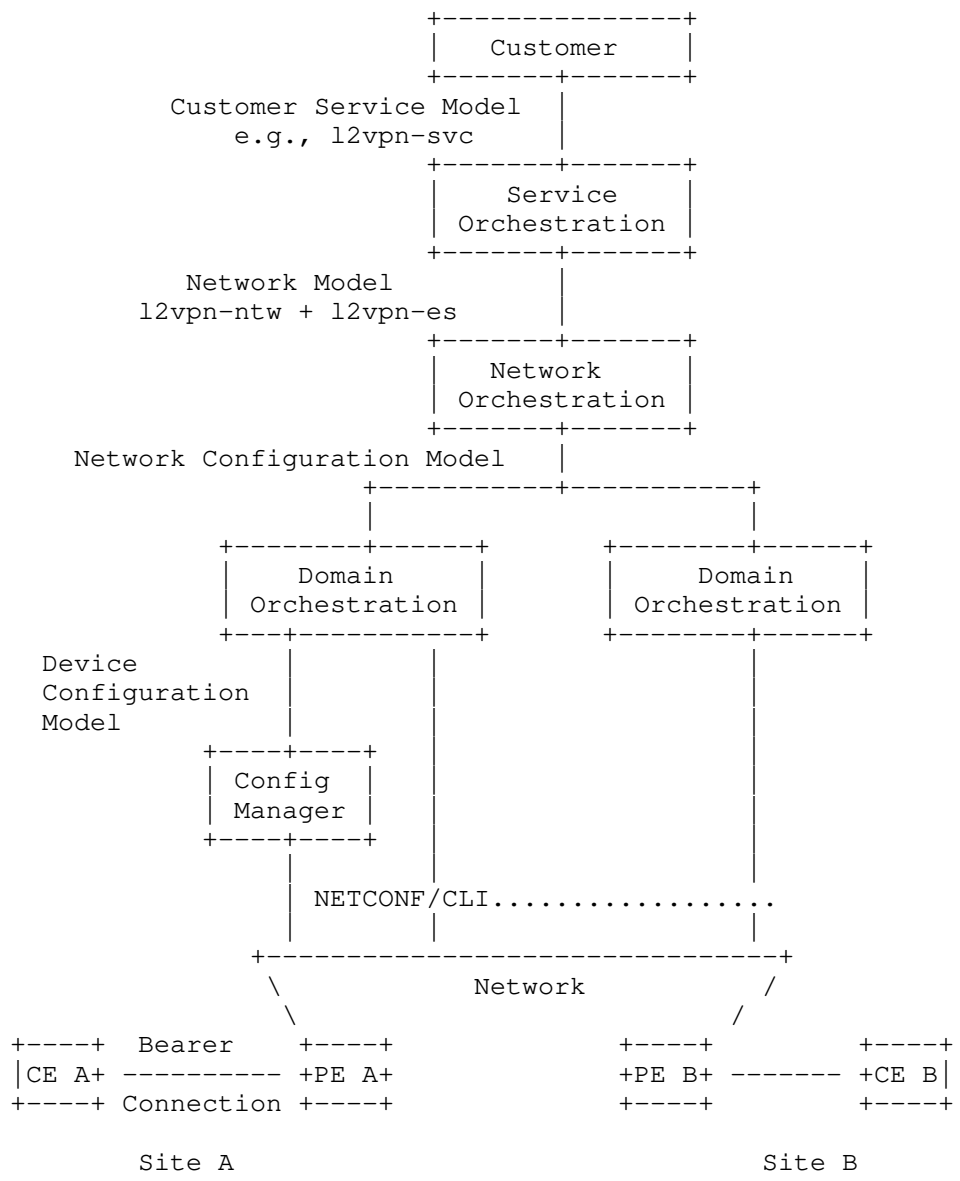
ACL	Access Control List
BGP	Border Gateway Protocol
BUM	Broadcast, unknown unicast, or multicast
CE	Customer Edge
ES	Ethernet Segment
ESI	Ethernet Segment Identifier
EVPN	Ethernet VPN
L2VPN	Layer 2 Virtual Private Network
L2SM	L2VPN Service Model
L2NM	L2VPN Network Model
MAC	Media Access Control
PBB	Provider Backbone Bridging
PCP	Priority Code Point
PE	Provider Edge
QoS	Quality of Service
RD	Route Distinguisher
RT	Route Target
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VPWS	Virtual Private Wire Service
VRF	Virtual Routing and Forwarding

4. Reference Architecture

Figure 1 illustrates how the L2NM is used. As a reminder, this figure is an expansion of the architecture presented in Section 3 of [RFC8466] and decomposes the box marked "orchestration" in that figure into three separate functional components called "Service Orchestration", "Network Orchestration", and "Domain Orchestration".

Similar to Section 3 of [RFC8466], CE to PE attachment is achieved through a bearer with a Layer 2 connection on top. The bearer refers to properties of the attachment that are below Layer 2, while the connection refers to Layer 2 protocol-oriented properties.

The reader may refer to [RFC8309] for the distinction between the "Customer Service Model", the "Service Delivery Model", the "Network Configuration Model", and the "Device Configuration Model". The "Domain Orchestration" and "Config Manager" roles may be performed by "SDN Controllers".



NETCONF: Network Configuration Protocol
CLI: Command-Line Interface

Figure 1: L2SM and L2NM Interaction

The customer may use various means to request a service that may trigger the instantiation of an L2NM. The customer may use the L2SM or may rely upon more abstract models to request a service that relies upon an L2VPN service. For example, the customer may supply an IP Connectivity Provisioning Profile (CPP) that characterizes the requested service [RFC7297], an enhanced VPN (VPN+) service [I-D.ietf-teas-enhanced-vpn], or an IETF network slice service [I-D.ietf-teas-ietf-network-slices].

Note also that both the L2SM and the L2NM may be used in the context of the Abstraction and Control of TE Networks (ACTN) framework [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC).

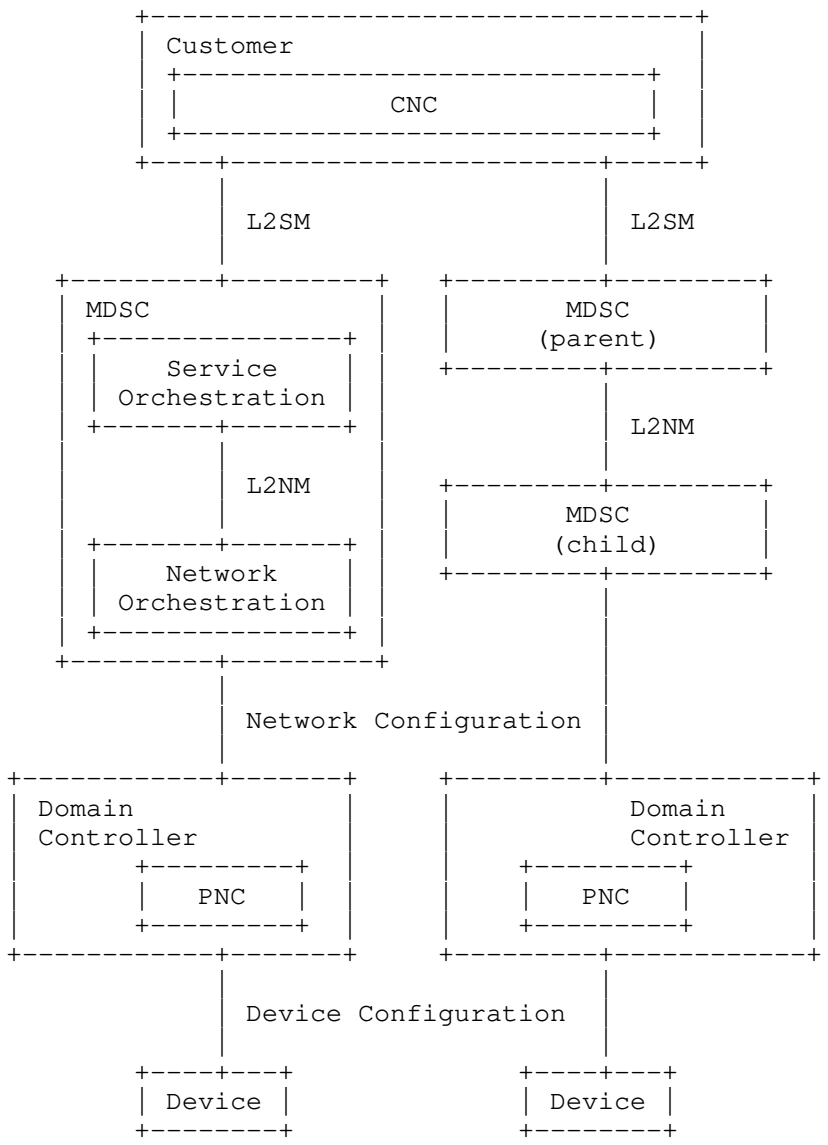


Figure 2: L2SM and L2NM in the Context of ACTN

5. Relationship to Other YANG Data Models

The "ietf-vpn-common" module [RFC9181] includes a set of identities, types, and groupings that are meant to be reused by VPN-related YANG modules independently of the layer (e.g., Layer 2, Layer 3) and the type of the module (e.g., network model, service model) including future revisions of existing models (e.g., [RFC8466]). The L2NM reuses these common types and groupings.

Also, the L2NM uses the IANA-maintained modules "iana-bgp-l2-encaps" (Section 8.1) and "iana-pseudowire-types" (Section 8.2) to identify Layer 2 encapsulation and pseudowire types. More details are provided in Sections 7.5.2.1 and 7.5.2.3.

For the particular case of EVPN, the L2NM includes a name that refers to an Ethernet segment that is created using the "ietf-ethernet-segment" module (Section 8.3). Some ES-related examples are provided in Appendices A.4 and A.5.

As discussed in Section 4, the L2NM is used to manage L2VPN services within a service provider network. The module provides a network view of the L2VPN service. Such a view is only visible to the service provider and is not exposed outside (to customers, for example). The following discusses how the L2NM interfaces with other YANG modules:

L2SM: The L2NM is not a customer service model.

> The internal view of the service (i.e., the L2NM) may be mapped to an external view which is visible to customers: L2VPN Service Model (L2SM) [RFC8466].
>
> The L2NM can be fed with inputs that are requested by customers, typically, relying upon an L2SM template. Concretely, some parts of the L2SM module can be directly mapped into the L2NM while other parts are generated as a function of the requested service and local guidelines. Finally, there are parts local to the service provider and do not map directly to the L2SM.
>
> Note that using the L2NM within a service provider does not assume, nor does it preclude, exposing the VPN service via the L2SM. This is deployment specific. Nevertheless, the design of L2NM tries to align as much as possible with the features supported by the L2SM to ease the grafting of both the L2NM and the L2SM for the sake of highly automated VPN service provisioning and delivery.

Network Topology Modules: An L2VPN involves nodes that are part of a

topology managed by the service provider network. Such a topology can be represented using the network topology module in [RFC8345] or its extension, such as a network YANG module for Service Attachment Points (SAPs) [I-D.ietf-opsawg-sap].

Device Modules: The L2NM is not a device model.

Once a global VPN service is captured by means of the L2NM, the actual activation and provisioning of the VPN service will involve a variety of device modules to tweak the required functions for the delivery of the service. These functions are supported by the VPN nodes and can be managed using device YANG modules. A non-comprehensive list of such device YANG modules is provided below:

* Interfaces [RFC8343].

* BGP [I-D.ietf-idr-bgp-model].

* MPLS [RFC8960].

* Access Control Lists (ACLs) [RFC8519].

How the L2NM is used to derive device-specific actions is implementation specific.

6. Description of the Ethernet Segment YANG Module

The 'ietf-ethernet-segment' module (Figure 3) is used to manage a set of Ethernet segments in the context of an EVPN service.

```
module: ietf-ethernet-segment
  +--rw ethernet-segments
     +--rw ethernet-segment* [name]
        +--rw name                                    string
        +--rw esi-type?                               identityref
        +--rw (esi-choice)?
        |  +--:(directly-assigned)
        |  |  +--rw ethernet-segment-identifier?   yang:hex-string
        |  +--:(auto-assigned)
        |     +--rw esi-auto
        |        +--rw (auto-mode)?
        |        |  +--:(from-pool)
        |        |  |  +--rw esi-pool-name?               string
        |        |  +--:(full-auto)
        |        |     +--rw auto?                        empty
        |        +--ro auto-ethernet-segment-identifier?
        |                 yang:hex-string
        +--rw esi-redundancy-mode?                    identityref
        +--rw df-election
        |  +--rw df-election-method?   identityref
        |  +--rw revertive?            boolean
        |  +--rw election-wait-time?   uint32
        +--rw split-horizon-filtering?                boolean
        +--rw pbb
        |  +--rw backbone-src-mac?   yang:mac-address
        +--rw member* [ne-id interface-id]
           +--rw ne-id           string
           +--rw interface-id    string
```

Figure 3: Ethernet Segments Tree Structure

The descriptions of the data nodes depicted in Figure 3 are as follows:

'name': Sets a name to uniquely identify an ES within a service provider network. In order to ease referencing ESes by their name in other modules, "es-ref" typedef is defined.

This typedef is used in the VPN network access level of the L2NM to reference an ES (Section 7.6). An example to illustrate such a use in the L2NM is provided in Appendix A.4.

'esi-type': Indicates the Ethernet Segment Identifier (ESI) type as discussed in Section 5 of [RFC7432]. ESIs can be automatically assigned either with or without indicating a pool from which an ESI should be taken ('esi-pool-name'). The following types are supported:

'esi-type-0-operator': The ESI is directly configured by the VPN service provider. The configured value is provided in 'ethernet-segment-identifier'.

'esi-type-1-lacp': The ESI is auto-generated from the IEEE 802.1AX Link Aggregation Control Protocol (LACP) [IEEE802.1AX].

'esi-type-2-bridge': The ESI is auto-generated and determined based on the Layer 2 bridge protocol.

'esi-type-3-mac': The ESI is a MAC-based ESI value that can be auto-generated or configured by the VPN service provider.

'esi-type-4-router-id': The ESI is auto-generated or configured by the VPN service provider based on the Router ID. The 'router-id' supplied in Section 7.5 can be used to auto-derive an ESI when this type is used.

'esi-type-5-asn': The ESI is auto-generated or configured by the VPN service provider based on the Autonomous System (AS) number. The 'local-autonomous-system' supplied in Section 7.4 can be used to auto-derive an ESI when this type is used.

Auto-generated values can be retrieved using 'auto-ethernet-segment-identifier'.

'esi-redundancy-mode': Specifies the EVPN redundancy mode for a given ES. The following modes are supported: Single-Active (Section 14.1.1 of [RFC7432]) or All-Active (Section 14.1.2 of [RFC7432]).

'df-election': Specifies a set of parameters related to the Designated Forwarder (DF) election (Section 8.5 of [RFC7432]). For example, this data node can be used to indicate an election method (e.g., [RFC8584] or [I-D.ietf-bess-evpn-pref-df]). If no election method is indicated, the default method defined in Section 8.5 of [RFC7432] is used.

As discussed in Section 1.3.2 of [RFC8584], the default behavior is to trigger the DF election procedure when a DF fails (e.g., link failure). The former DF will take over when it is available again. Such a mode is called revertive. The behavior can be overridden by setting the 'revertive' leaf to 'false'.

Also, this data node can be used to configure a DF Wait timer ('election-wait-time') (Section 2.1 of [RFC8584]).

'split-horizon-filtering': Controls the activation of the split-

horizon filtering for an ES (Section 8.3 of [RFC7432]).

'pbb': Indicates data nodes that are specific to PBB [IEEE-802-1ah]:

'backbone-src-mac': Associates a Provider Backbone MAC (B-MAC) address with an ES. This is particularly useful for All-Active multihomed ESes (Section 9.1 of [RFC7623]).

'member': Lists the members of an ES in a service provider network.

7. Description of the L2NM YANG Module

The L2NM ('ietf-l2vpn-ntw', Section 8.4) is used to manage L2VPNs within a service provider network. In particular, the 'ietf-l2vpn-ntw' module can be used to create, modify, delete and retrieve L2VPN services in a network controller. The module is designed to minimize the amount of customer-related information.

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

Note that the following subsections introduce some data nodes that enclose textual descriptions (e.g., VPN service (Section 7.3), VPN node (Section 7.5), or VPN network access (Section 7.6)). Such descriptions are not intended for random end users but for network/system/software engineers that use their local context to provide and interpret such information. Therefore, no mechanism for language tagging is needed.

7.1. Overall Structure of the Module

The 'ietf-l2vpn-ntw' module uses two main containers: 'vpn-profiles' and 'vpn-services' (see Figure 4).

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

The 'vpn-services' container maintains the set of L2VPN services managed in the service provider network. The module allows creating a new L2VPN service by adding a new instance of 'vpn-service'. The 'vpn-service' is the data structure that abstracts the VPN service (Section 7.3).

```
module: ietf-l2vpn-ntw
  +--rw l2vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           ...
           +--rw vpn-nodes
              +--rw vpn-node* [vpn-node-id]
                 ...
                 +--rw vpn-network-accesses
                    +--rw vpn-network-access* [id]
                       ...
```

Figure 4: Overall L2NM Tree Structure

7.2. VPN Profiles

The 'vpn-profiles' container (Figure 5) is used by a VPN service provider to define and maintain a set of VPN profiles [RFC9181] that apply to one or several VPN services.

```
+--rw l2vpn-ntw
   +--rw vpn-profiles
   |  +--rw valid-provider-identifiers
   |     +--rw external-connectivity-identifier* [id]
   |     |       {external-connectivity}?
   |     |  +--rw id    string
   |     +--rw encryption-profile-identifier* [id]
   |     |  +--rw id    string
   |     +--rw qos-profile-identifier* [id]
   |     |  +--rw id    string
   |     +--rw bfd-profile-identifier* [id]
   |     |  +--rw id    string
   |     +--rw forwarding-profile-identifier* [id]
   |     |  +--rw id    string
   |     +--rw routing-profile-identifier* [id]
   |        +--rw id    string
   +--rw vpn-services
      ...
```

Figure 5: VPN Profiles Subtree Structure

The exact definition of these profiles is local to each VPN service provider. The model only includes an identifier for these profiles in order to ease identifying and binding local policies when building a VPN service. As shown in Figure 5, the following identifiers can be included:

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

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

'qos-profile-identifier': A Quality of Service (QoS) profile refers to as set of policies, such as classification, marking, and actions (e.g., [RFC3644]).

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

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

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

7.3. VPN Services

The 'vpn-service' is the data structure that abstracts an L2VPN service in the service provider network. Each 'vpn-service' is uniquely identified by an identifier: 'vpn-id'. Such a 'vpn-id' is only meaningful locally within the network controller. The subtree of the 'vpn-services' is shown in Figure 6.

```
+--rw vpn-services
   +--rw vpn-service* [vpn-id]
      +--rw vpn-id                          vpn-common:vpn-id
      +--rw vpn-name?                       string
      +--rw vpn-description?                string
      +--rw customer-name?                  string
      +--rw parent-service-id?              vpn-common:vpn-id
      +--rw vpn-type?                       identityref
      +--rw vpn-service-topology?           identityref
      +--rw bgp-ad-enabled?                 boolean
      +--rw signaling-type?                 identityref
      +--rw global-parameters-profiles
      |  ...
      +--rw underlay-transport
      |  +--rw (type)?
      |     +--:(abstract)
      |     |  +--rw transport-instance-id?   string
      |     |  +--rw instance-type?           identityref
      |     +--:(protocol)
      |        +--rw protocol*                identityref
      +--rw status
      |  +--rw admin-status
      |  |  +--rw status?         identityref
      |  |  +--rw last-change?   yang:date-and-time
      |  +--ro oper-status
      |     +--ro status?         identityref
      |     +--ro last-change?   yang:date-and-time
      +--rw vpn-nodes
         ...
```

Figure 6: VPN Services Subtree

The descriptions of the VPN service data nodes that are depicted in Figure 6 are as follows:

'vpn-id': An identifier that is used to uniquely identify the L2VPN service within the L2NM scope.

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

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

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

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

'parent-service-id': Refers to an identifier of the parent service (e.g., the L2SM, IETF network slice, VPN+) that triggered the creation of the L2VPN service. This identifier is used to easily correlate the (network) service as built in the network with a service order. A controller can use that correlation to enrich or populate some fields (e.g., description fields) as a function of local deployments.

'vpn-type': Indicates the L2VPN type. The following types, defined in [RFC9181], can be used for the L2NM:

'vpls': Virtual Private LAN Service (VPLS) as defined in [RFC4761] or [RFC4762]. This type is also used for hierarchical VPLS (H-VPLS) (Section 10 of [RFC4762]).

'vpws': Virtual Private Wire Service (VPWS) as defined in Section 3.1.1 of [RFC4664].

'vpws-evpn': VPWS as defined in [RFC8214].

'pbb-evpn': Provider Backbone Bridging (PBB) EVPNs as defined in [RFC7623].

'mpls-evpn': MPLS-based EVPNs [RFC7432].

'vxlan-evpn': VXLAN based EVPNs [RFC8365].

The type is used as a condition for the presence of some data nodes in the L2NM.

'vpn-service-topology': Indicates the network topology for the service: hub-spoke, any-to-any, or custom. These types are defined in [RFC9181].

'bgp-ad-enabled': Controls whether BGP auto-discovery is enabled. If so, additional data nodes are included (Section 7.5.1).

'signaling-type': Indicates the signaling that is used for setting up pseudowires. Signaling type values are taken from [RFC9181]. The following signaling options are supported:

'bgp-signaling': The L2NM supports two flavors of BGP-signaled L2VPNs:

'l2vpn-bgp': The service is a Multipoint VPLS that uses a BGP control plane as described in [RFC4761] and [RFC6624].

'evpn-bgp': The service is a Multipoint VPLS that uses also a

BGP control plane, but also includes the additional EVPN features and related parameters [RFC7432] and [RFC7209].

'ldp-signaling': A Multipoint VPLS that uses a mesh of LDP-signaled Pseudowires [RFC6074].

'l2tp-signaling': The L2NM uses L2TP-signaled Pseudowires as described in [RFC6074].

Table 1 summarizes the allowed signaling types for each VPN service type ('vpn-type'). See Section 7.5.2 for more details.

VPN Type	Signaling Options
vpls	l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)
vpws	l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)
vpws-evpn	bgp-signaling (evpn-bgp)
pbb-evpn	bgp-signaling (evpn-bgp)
mpls-evpn	bgp-signaling (evpn-bgp)
vxlan-evpn	bgp-signaling (evpn-bgp)

Table 1: Signaling Options per VPN Service Type

'global-parameters-profiles': Defines reusable parameters for the same L2VPN service.

More details are provided in Section 7.4.

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

A rich set of protocol identifiers that can be used to refer to an underlay transport (or how such an underlay is set up) are defined in [RFC9181].

The model defined in Section 6.3.2 of [I-D.ietf-teas-te-service-mapping-yang] may be used if specific protection and availability requirements are needed between PEs.

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

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

'vpn-node': An abstraction that represents a set of policies applied to a network node and belonging to a single 'vpn-service'. An L2VPN service is typically built by adding instances of 'vpn-node' to the 'vpn-nodes' container.

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

Note that, as this is a network data model, the information about customers sites is not required in the model. Such information is rather relevant in the L2SM. Whether that information is included in the L2NM, e.g., to populate the various 'description' data nodes is implementation specific.

More details are provided in Section 7.5.

7.4. Global Parameters Profiles

The 'global-parameters-profile' defines reusable parameters for the same L2VPN service instance ('vpn-service'). Global parameters profiles are defined at the VPN service level, activated at the VPN node level, and then an activated VPN profile may be used at the VPN network access level. Each VPN instance profile is identified by 'profile-id'. Some of the data nodes can be adjusted at the VPN node or VPN network access levels. These adjusted values take precedence

over the global values. The subtree of 'global-parameters-profile' is depicted in Figure 7.

```
      ...
      +--rw vpn-services
         +--rw vpn-service* [vpn-id]
            ...
            +--rw global-parameters-profiles
            |  +--rw global-parameters-profile* [profile-id]
            |     +--rw profile-id                    string
            |     +--rw (rd-choice)?
            |     |  +--:(directly-assigned)
            |     |  |  +--rw rd?
            |     |  |           rt-types:route-distinguisher
            |     |  +--:(directly-assigned-suffix)
            |     |  |  +--rw rd-suffix?            uint16
            |     |  +--:(auto-assigned)
            |     |  |  +--rw rd-auto
            |     |  |     +--rw (auto-mode)?
            |     |  |     |  +--:(from-pool)
            |     |  |     |  |  +--rw rd-pool-name?   string
            |     |  |     |  +--:(full-auto)
            |     |  |     |     +--rw auto?           empty
            |     |  |     +--ro auto-assigned-rd?
            |     |  |               rt-types:route-distinguisher
            |     |  +--:(auto-assigned-suffix)
            |     |  |  +--rw rd-auto-suffix
            |     |  |     +--rw (auto-mode)?
            |     |  |     |  +--:(from-pool)
            |     |  |     |  |  +--rw rd-pool-name?        string
            |     |  |     |  +--:(full-auto)
            |     |  |     |     +--rw auto?                empty
            |     |  |     +--ro auto-assigned-rd-suffix?   uint16
            |     |  +--:(no-rd)
            |     |     +--rw no-rd?                empty
            |     +--rw vpn-target* [id]
            |     |  +--rw id                   uint8
            |     |  +--rw route-targets* [route-target]
            |     |  |  +--rw route-target    rt-types:route-target
            |     |  +--rw route-target-type
            |     |           rt-types:route-target-type
            |     +--rw vpn-policies
            |     |  +--rw import-policy?   string
            |     |  +--rw export-policy?   string
            |     +--rw local-autonomous-system?    inet:as-number
            |     +--rw svc-mtu?                    uint32
            |     +--rw ce-vlan-preservation?       boolean
            |     +--rw ce-vlan-cos-preservation?   boolean
```

```
|      +--rw control-word-negotiation?   boolean
|      +--rw mac-policies
|      |  +--rw mac-addr-limit
|      |  |  +--rw limit-number?    uint16
|      |  |  +--rw time-interval?   uint32
|      |  |  +--rw action?          identityref
|      |  +--rw mac-loop-prevention
|      |     +--rw window?            uint32
|      |     +--rw frequency?         uint32
|      |     +--rw retry-timer?       uint32
|      |     +--rw protection-type?   identityref
|      +--rw multicast {vpn-common:multicast}?
|         +--rw enabled?                  boolean
|         +--rw customer-tree-flavors
|            +--rw tree-flavor*   identityref
|         ...
```

Figure 7: Global Parameters Profiles Subtree

The description of the global parameters profile is as follows:

'profile-id': Uniquely identifies a global parameter profile in the context of an L2VPN service.

'rd': As defined in [RFC9181], these RD assignment modes are supported: direct assignment, automatic assignment from a given pool, full automatic assignment, and no assignment.

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

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

'local-autonomous-system': Indicates the Autonomous System Number (ASN) that is configured for the VPN node. The ASN can be used to auto-derive some other attributes such as RDs or Ethernet Segment Identifiers (ESIs).

'svc-mtu': Is the service MTU for an L2VPN service (i.e., Layer 2 MTU including L2 frame header/trailer). It is also known as the maximum transmission unit or maximum frame size. It is expressed in bytes.

'ce-vlan-preservation': Is set to preserve the Customer Edge VLAN

IDs (CE-VLAN IDs) from ingress to egress, i.e., CE-VLAN tag of the egress frame are identical to those of the ingress frame that yielded this egress service frame. If all-to-one bundling within a site is enabled, then preservation applies to all ingress service frames. If all-to-one bundling is disabled, then preservation applies to tagged Ingress service frames having CE-VLAN ID 1 through 4094.

'ce-vlan-cos-preservation': Controls the CE VLAN CoS preservation. When set, Priority Code Point (PCP) bits in the CE-VLAN tag of the egress frame are identical to those of the ingress frame that yielded this egress service frame.

'control-word-negotiation': Controls whether control-word negotiation is enabled (if set to true) or not (if set to false). Refer to Section 7 of [RFC8077] for more details.

'mac-policies': Includes a set of MAC policies that apply to the service:

'mac-addr-limit': Is a container of MAC address limit configuration. It includes the following data nodes:

'limit-number': Maximum number of MAC addresses learned from the customer for a single service instance.

'time-interval': The aging time of the MAC address.

'action': Specifies the action when the upper limit is exceeded: drop the packet, flood the packet, or simply send a warning message.

'mac-loop-prevention': Container for MAC loop prevention.

'window': The time interval over which a MAC mobility event is detected and checked.

'frequency': The number of times to detect MAC duplication, where a 'duplicate MAC address' situation has occurred within the 'window' time interval, and the duplicate MAC address has been added to a list of duplicate MAC addresses.

'retry-timer': The retry timer. When the retry timer expires, the duplicate MAC address will be flushed from the MAC-VRF.

'protection-type': It defines the loop prevention type (e.g., shut).

'multicast': Controls whether multicast is allowed in the service.

7.5. VPN Nodes

The 'vpn-node' (Figure 8) is an abstraction that represents a set of policies/configurations applied to a network node and that belong to a single 'vpn-service'. A 'vpn-node' contains 'vpn-network-accesses', which are the interfaces involved in the creation of the VPN. The customer sites are connected to the 'vpn-network-accesses'.

```
      +--rw l2vpn-ntw
         +--rw vpn-profiles
         |  ...
         +--rw vpn-services
            +--rw vpn-service* [vpn-id]
               ...
               +--rw vpn-nodes
                  +--rw vpn-node* [vpn-node-id]
                     +--rw vpn-node-id            vpn-common:vpn-id
                     +--rw description?           string
                     +--rw ne-id?                 string
                     +--rw role?                  identityref
                     +--rw router-id?             rt-types:router-id
                     +--rw active-global-parameters-profiles
                     |  +--rw global-parameters-profile* [profile-id]
                     |     +--rw profile-id                    leafref
                     |     +--rw local-autonomous-system?
                     |     |       inet:as-number
                     |     +--rw svc-mtu?                      uint32
                     |     +--rw ce-vlan-preservation?         boolean
                     |     +--rw ce-vlan-cos-preservation?     boolean
                     |     +--rw control-word-negotiation?     boolean
                     |     +--rw mac-policies
                     |     |  +--rw mac-addr-limit
                     |     |  |  +--rw limit-number?    uint16
                     |     |  |  +--rw time-interval?   uint32
                     |     |  |  +--rw action?          identityref
                     |     |  +--rw mac-loop-prevention
                     |     |     +--rw window?            uint32
                     |     |     +--rw frequency?         uint32
                     |     |     +--rw retry-timer?       uint32
                     |     |     +--rw protection-type?   identityref
                     |     +--rw multicast {vpn-common:multicast}?
                     |        +--rw enabled?                   boolean
                     |        +--rw customer-tree-flavors
                     |           +--rw tree-flavor*   identityref
                     +--rw status
                     |  ...
                     +--rw bgp-auto-discovery
                     |  ...
                     +--rw signaling-option
                     |  ...
                     +--rw vpn-network-accesses
                        ...
```

Figure 8: VPN Nodes Subtree

The descriptions of VPN node data nodes are as follows:

'vpn-node-id': Used to uniquely identify a node that enables a VPN network access.

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

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

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

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

'active-global-parameters-profiles': Lists the set of active global VPN parameters profiles for this VPN node. Concretely, one or more global profiles that are defined at the VPN service level (i.e., under 'l2vpn-ntw/vpn-services/vpn-service' level) can be activated at the VPN node level; each of these profiles is uniquely identified by means of 'profile-id'. The structure of 'active-global-parameters-profiles' uses the same data nodes as Section 7.4 except RD and RT related data nodes.

Values defined in 'active-global-parameters-profiles' overrides the values defined in the VPN service level.

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

'bgp-auto-discovery': See Section 7.5.1.

'signaling-option': See Section 7.5.2.

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

Note that, unlike the L2SM, the L2NM does not need to model the customer site -- only the points that receive traffic from the site are covered (i.e., the PE side of Provider Edge to Customer Edge (PE-CE) connections). Hence, the VPN network access contains the connectivity information between the provider's network and the customer premises. The VPN profiles ('vpn-profiles') have a set of routing policies that can be applied during the service creation.

See Section 7.6 for more details.

7.5.1. BGP Auto-Discovery

The 'bgp-auto-discovery' container (Figure 9) includes the required information for the activation of BGP auto-discovery [RFC4761][RFC6624].

```
  +--rw l2vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           ...
           +--rw vpn-nodes
              +--rw vpn-node* [vpn-node-id]
                 ...
                 +--rw bgp-auto-discovery
                 |  +--rw (bgp-type)?
                 |  |  +--:(l2vpn-bgp)
                 |  |  |  +--rw vpn-id?
                 |  |  |          vpn-common:vpn-id
                 |  |  +--:(evpn-bgp)
                 |  |     +--rw evpn-type?           leafref
                 |  |     +--rw auto-rt-enable?      boolean
                 |  |     +--ro auto-route-target?
                 |  |             rt-types:route-target
                 |  +--rw (rd-choice)?
                 |  |  +--:(directly-assigned)
                 |  |  |  +--rw rd?
                 |  |  |          rt-types:route-distinguisher
                 |  |  +--:(directly-assigned-suffix)
                 |  |  |  +--rw rd-suffix?           uint16
                 |  |  +--:(auto-assigned)
                 |  |  |  +--rw rd-auto
                 |  |  |     +--rw (auto-mode)?
                 |  |  |     |  +--:(from-pool)
                 |  |  |     |  |  +--rw rd-pool-name?   string
                 |  |  |     |  +--:(full-auto)
                 |  |  |     |     +--rw auto?            empty
                 |  |  |     +--ro auto-assigned-rd?
                 |  |  |             rt-types:route-distinguisher
                 |  |  +--:(auto-assigned-suffix)
                 |  |  |  +--rw rd-auto-suffix
                 |  |  |     +--rw (auto-mode)?
                 |  |  |     |  +--:(from-pool)
                 |  |  |     |  |  +--rw rd-pool-name?        string
                 |  |  |     |  +--:(full-auto)
```

```
|  |  |     |     +--rw auto?                empty
|  |  |     +--ro auto-assigned-rd-suffix?   uint16
|  |  +--:(no-rd)
|  |     +--rw no-rd?               empty
|  +--rw vpn-target* [id]
|  |  +--rw id                    uint8
|  |  +--rw route-targets* [route-target]
|  |  |  +--rw route-target    rt-types:route-target
|  |  +--rw route-target-type
|  |           rt-types:route-target-type
|  +--rw vpn-policies
|     +--rw import-policy?   string
|     +--rw export-policy?   string
+--rw signaling-option
|  ...
+--rw vpn-network-accesses
   ...
```

Figure 9: BGP Auto-Discovery Subtree

As discussed in Section 1 of [RFC6624], all of BGP-based methods include the notion of a VPN identifier that serves to unify components of a given VPN and the concept of auto-discovery; hence the support of the data node 'vpn-id'.

For the particular case of EVPN, the L2NM supports RT auto-derivation based on the Ethernet Tag ID specified in Section 7.10.1 of [RFC7432]. A VPN service provider can enable/disable this functionality by means of 'auto-rt-enable'. The assigned RT can be retrieved using 'auto-route-target'.

For all BGP-based L2VPN flavors, other data nodes such as RD and RT are used. These data nodes have the same structure as the one discussed in Section 7.4.

7.5.2. Signaling Options

The 'signaling-option' container (Figure 10) defines a set of data nodes for a given signaling protocol that is used for an L2VPN service. As discussed in Section 7.3, several signaling options to exchange membership information between PEs of an L2VPN are supported. The signaling type to be used for an L2VPN service is controlled at the VPN service level by means of 'signaling-type'.

```
...
+--rw vpn-nodes
   +--rw vpn-node* [vpn-node-id]
   ...
   +--rw signaling-option
   |  +--rw advertise-mtu?        boolean
   |  +--rw mtu-allow-mismatch?   boolean
   |  +--rw signaling-type?       leafref
   |  +--rw (signaling-option)?
   |     +--:(bgp)
   |     |  ...
   |     +--:(ldp-or-l2tp)
   |        +--rw ldp-or-l2tp
   |           ...
   |           +--rw (ldp-or-l2tp)?
   |              +--:(ldp)
   |              |  ...
   |              +--:(l2tp)
   |                 ...
```

Figure 10: Signaling Option Overall Subtree

The following signaling data nodes are supported:

'advertise-mtu': Controls whether MTU is advertised when setting a pseudowire (e.g., Section 4.3 of [RFC4667], Section 5.1 of [RFC6624], or Section 6.1 of [RFC4762]).

'mtu-allow-mismatch': When set to true, it allows MTU mismatch for a pseudowire (see, e.g., Section 4.3 of [RFC4667]).

'signaling-type': Indicates the signaling type. This type inherits the value of 'signaling-type' defined at the service level (Section 7.3).

'bgp': Is provided when BGP is used for L2VPN signaling. Refer to Section 7.5.2.1 for more details.

'ldp': The model supports the configuration of the parameters that are discussed in Section 6 of [RFC4762]. Refer to Section 7.5.2.2 for more details.

'l2tp': The model supports the configuration of the parameters that are discussed in Section 4 of [RFC4667]. Refer to Section 7.5.2.3 for more details.

Note that LDP and L2TP choices are bundled ("ldp-or-l2tp") because they share a set of common parameters that are further detailed in Sections 7.5.2.2 and 7.5.2.3.

7.5.2.1. BGP

The structure of the BGP-related data nodes is provided in Figure 11.

```
...
|  +--rw (signaling-option)?
|     ...
|     +--:(bgp)
|        |  +--rw (bgp-type)?
|        |     +--:(l2vpn-bgp)
|        |     |  +--rw ce-range?     uint16
|        |     |  +--rw pw-encapsulation-type?
|        |     |  |       identityref
|        |     |  +--rw vpls-instance
|        |     |     +--rw vpls-edge-id?         uint16
|        |     |     +--rw vpls-edge-id-range?   uint16
|        |     +--:(evpn-bgp)
|        |        +--rw evpn-type?                 leafref
|        |        +--rw service-interface-type?
|        |        |       identityref
|        |        +--rw evpn-policies
|        |           +--rw mac-learning-mode?
|        |           |       identityref
|        |           +--rw ingress-replication?
|        |           |       boolean
|        |           +--rw p2mp-replication?
|        |           |       boolean
|        |           +--rw arp-proxy {vpn-common:ipv4}?
|        |           |  +--rw enable?           boolean
|        |           |  +--rw arp-suppression?
|        |           |  |       boolean
|        |           |  +--rw ip-mobility-threshold?
|        |           |  |       uint16
|        |           |  +--rw duplicate-ip-detection-interval?
|        |           |          uint16
|        |           +--rw nd-proxy {vpn-common:ipv6}?
|        |           |  +--rw enable?          boolean
|        |           |  +--rw nd-suppression?
|        |           |  |       boolean
|        |           |  +--rw ip-mobility-threshold?
|        |           |  |       uint16
|        |           |  +--rw duplicate-ip-detection-interval?
|        |           |          uint16
|        |           +--rw underlay-multicast?
```

```
|      |           |       boolean
|      |           +--rw flood-unknown-unicast-supression?
|      |           |       boolean
|      |           +--rw vpws-vlan-aware?        boolean
|      |           +--rw bum-management
|      |           |  +--rw discard-broadcast?
|      |           |  |       boolean
|      |           |  +--rw discard-unknown-multicast?
|      |           |  |       boolean
|      |           |  +--rw discard-unknown-unicast?
|      |           |          boolean
|      |           +--rw pbb
|      |              +--rw backbone-src-mac?
|      |                      yang:mac-address
|      +--:(ldp-or-l2tp)
|         ...
```

Figure 11: Signaling Option Subtree (BGP)

Remote CEs that are entitled to connect to the same VPN should fit with the CE range ('ce-range') as discussed in Section 2.2.3 of [RFC6624]. 'pw-encapsulation-type' is used to control the pseudowire encapsulation type (Section 3 of [RFC6624]). The value of the 'pw-encapsulation-type' are taken from the IANA-maintained "iana-bgp-l2-encaps" module (Section 8.1).

For the specific case of VPLS, the VPLS Edge ID (VE ID, 'vpls-edge-id') and a VE ID range ('vpls-edge-id-range') are provided as per Section 3.2 of [RFC4761]. If different VE IDs are required (e.g., multihoming as per Section 3.5 of [RFC4761]), these IDs are configured at the VPN network access level (under 'signaling-option' in Section 7.6).

For EVPN-related L2VPNs, 'service-interface-type' indicates whether this is a VLAN-based, VLAN bundle, or VLAN-aware bundle service interface (Section 6 of [RFC7432]). Moreover, a set of policies can be provided such as MAC address learning mode (Section 9 of [RFC7432]), ingress replication (Section 12.1 of [RFC7432]), Address Resolution Protocol (ARP) and Nighbor Discovery (ND) proxy (Section 10 of [RFC7432]), processing of Broadcast, unknown unicast, or multicast (BUM) (Section 12 of [RFC7432]), etc.

7.5.2.2. LDP

The model supports the configuration of the parameters that are discussed in Section 6 of [RFC4762]. Such parameters include an Attachment Group Identifier (AGI) (a.k.a., VPLS-id), a Source Attachment Individual Identifier (SAII), a list of peers that are associated with a Target Attachment Individual Identifier (TAII), a pseudowire type, and a pseudowire description (Figure 12). Unlike BGP, only Ethernet and Ethernet tagged mode are supported. The AGI, SAII, and TAII are encoded following the types defined in Section 3.4 of [RFC4446].

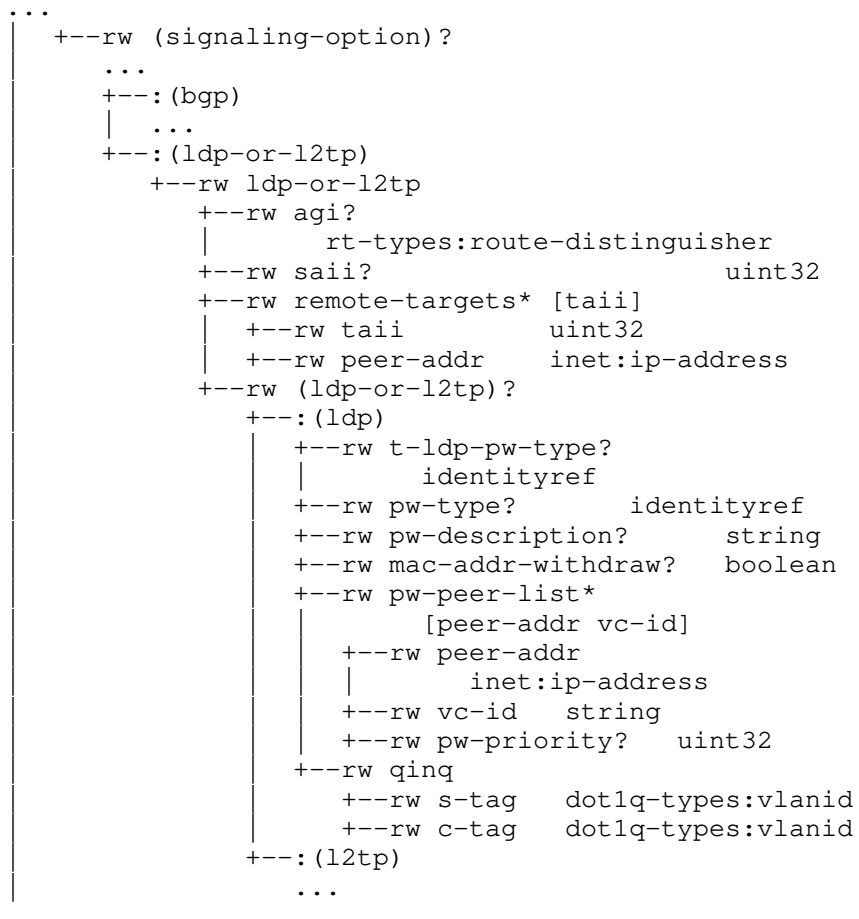


Figure 12: Signaling Option Subtree (LDP)

7.5.2.3. L2TP

The model supports the configuration of the parameters that are discussed in Section 4 of [RFC4667]. Such parameters include a Router ID that is used to uniquely identify a PE, a pseudowire type, an AGI, an SAII, and a list of peers that are associated with a TAII (Figure 13). The pseudowire type ('pseudowire-type') value is taken from the IANA-maintained "iana-pseudowire-types" module (Section 8.2).

```
...
|  +--rw (signaling-option)?
|     ...
|     +--:(bgp)
|     |  ...
|     +--:(ldp-or-l2tp)
|        +--rw ldp-or-l2tp
|           +--rw agi?
|           |       rt-types:route-distinguisher
|           +--rw saii?                        uint32
|           +--rw remote-targets* [taii]
|           |  +--rw taii          uint32
|           |  +--rw peer-addr     inet:ip-address
|           +--rw (ldp-or-l2tp)?
|              +--:(ldp)
|              |  ...
|              +--:(l2tp)
|                 +--rw router-id?
|                 |       rt-types:router-id
|                 +--rw pseudowire-type?
|                         identityref
|
...
```

Figure 13: Signaling Option Subtree (L2TP)

7.6. VPN Network Accesses

A 'vpn-network-access' (Figure 14) represents an entry point to a VPN service. In other words, this container encloses the parameters that describe the access information for the traffic that belongs to a particular L2VPN.

A 'vpn-network-access' includes information such as the connection on which the access is defined, the specific Layer 2 service requirements, etc.

```
          ...
          +--rw vpn-nodes
             +--rw vpn-node* [vpn-node-id]
                ...
                +--rw vpn-network-accesses
                   +--rw vpn-network-access* [id]
                      +--rw id                          vpn-common:vpn-id
                      +--rw description?                string
                      +--rw interface-id?               string
                      +--rw active-vpn-node-profile?    leafref
                      +--rw status
                      |  ...
                      +--rw connection
                      |  ...
                      +--rw (signaling-option)?
                      |  +--:(bgp)
                      |     +--rw (bgp-type)?
                      |        +--:(l2vpn-bgp)
                      |        |  +--rw ce-id?              uint16
                      |        |  +--rw remote-ce-id?       uint16
                      |        |  +--rw vpls-instance
                      |        |     +--rw vpls-edge-id?    uint16
                      |        +--:(evpn-bgp)
                      |           +--rw df-preference?     uint16
                      |           +--rw vpws-service-instance
                      |              ...
                      +--rw group* [group-id]
                      |  +--rw group-id                        string
                      |  +--rw precedence?                identityref
                      |  +--rw ethernet-segment-identifier?
                      |                                l2vpn-es:es-ref
                      +--rw ethernet-service-oam
                      |  ...
                      +--rw service
                         ...
```

Figure 14: VPN Network Access Subtree

The VPN network access comprises:

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

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

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

'active-vpn-node-profile': Provides a pointer to an active 'global-parameters-profile' at the VPN node level. Referencing an active 'global-parameters-profile' implies that all associated data nodes will be inherited by the VPN network access. However, some of the inherited data nodes (e.g., ACL policies) can be overridden at the VPN network access level. In such case, adjusted values take precedence over inherited values.

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

'connection': Represents and groups the set of Layer 2 connectivity from where the traffic of the L2VPN in a particular VPN Network access is coming. See Section 7.6.1.

'signaling-option': Indicates a set of signaling options that are specific to a given VPN network access, e.g., a CE ID ('ce-id' identifying the CE within the VPN) and a remote CE ID as discussed in Section 2.2.2 of [RFC6624].

It can also include a set of data nodes that are required for the configuration of a VPWS-EVPN [RFC8214]. See Section 7.6.2.

'group': Is used for grouping VPN network accesses by assigning the same identifier to these accesses. The precedence attribute is used to differentiate the primary and secondary accesses for a service with multiple accesses. An example to illustrate the use of this container for redundancy purposes is provided in Appendix A.6. This container is also used to identify the link of an ES by allocating the same ESI. An example to illustrate this functionality is provided in Appendices A.4 and A.5.

'ethernet-service-oam': Carries information about the service OAM. See Section 7.6.3.

'service': Specifies the service parameters (e.g., QoS, multicast) to apply for a given VPN network access. See Section 7.6.4.

7.6.1. Connection

The 'connection' container (Figure 15) is used to configure the relevant properties of the interface to which the L2VPN instance is attached to (e.g., encapsulation type, Link Aggregation Group (LAG) interfaces, split-horizon). The L2NM supports tag manipulation operations (e.g., tag rewrite).

Note that the 'connection' container does not include the physical-specific configuration as this is assumed to be directly handled using device modules (e.g., interfaces module). Moreover, this design is also meant to avoid manipulated global parameters at the service level and lower the risk of impacting other services sharing the same physical interface.

A reference to the bearer is maintained to allow keeping the link between the L2SM and the L2NM when both data models are used in a given deployment.

Some consistency checks should be ensured by implementations (typically, network controllers) for LAG interface as the same information (e.g., LACP system-id) should be provided to the involved nodes.

The L2NM inherits the 'member-link-list' structure from the L2SM (including indication of OAM 802.3ah support [IEEE-802-3ah]).

```
            ...
            +--rw vpn-nodes
               +--rw vpn-node* [vpn-node-id]
                  ...
                  +--rw vpn-network-accesses
                     +--rw vpn-network-access* [id]
                        ...
                        +--rw connection
                        |  +--rw l2-termination-point?
                        |  |       string
                        |  +--rw local-bridge-reference?
                        |  |       string
                        |  +--rw bearer-reference?         string
                        |  |       {vpn-common:bearer-reference}?
                        |  +--rw encapsulation
                        |  |  +--rw encap-type?            identityref
                        |  |  +--rw dot1q
                        |  |  |  +--rw tag-type?           identityref
                        |  |  |  +--rw cvlan-id?
                        |  |  |  |       dot1q-types:vlanid
                        |  |  |  +--rw tag-operations
                        |  |  |     +--rw (op-choice)?
                        |  |  |     |  +--:(pop)
                        |  |  |     |  |  +--rw pop?         empty
                        |  |  |     |  +--:(push)
                        |  |  |     |  |  +--rw push?        empty
                        |  |  |     |  +--:(translate)
                        |  |  |     |     +--rw translate?   empty
                        |  |  |     +--rw tag-1?
```

```
|  |  |  |       |       dot1q-types:vlanid
|  |  |  |       +--rw tag-1-type?
|  |  |  |       |       dot1q-types:dot1q-tag-type
|  |  |  |       +--rw tag-2?
|  |  |  |       |       dot1q-types:vlanid
|  |  |  |       +--rw tag-2-type?
|  |  |  |               dot1q-types:dot1q-tag-type
|  |  |  +--rw priority-tagged
|  |  |  |  +--rw tag-type?   identityref
|  |  |  +--rw qinq
|  |  |     +--rw tag-type?         identityref
|  |  |     +--rw svlan-id
|  |  |     |       dot1q-types:vlanid
|  |  |     +--rw cvlan-id
|  |  |     |       dot1q-types:vlanid
|  |  |     +--rw tag-operations
|  |  |        +--rw (op-choice)?
|  |  |        |  +--:(pop)
|  |  |        |  |  +--rw pop?         uint8
|  |  |        |  +--:(push)
|  |  |        |  |  +--rw push?        empty
|  |  |        |  +--:(translate)
|  |  |        |     +--rw translate?   empty
|  |  |        +--rw tag-1?
|  |  |        |       dot1q-types:vlanid
|  |  |        +--rw tag-1-type?
|  |  |        |       dot1q-types:dot1q-tag-type
|  |  |        +--rw tag-2?
|  |  |        |       dot1q-types:vlanid
|  |  |        +--rw tag-2-type?
|  |  |                dot1q-types:dot1q-tag-type
|  |  +--rw lag-interface
|  |     |    {vpn-common:lag-interface}?
|  |     +--rw lag-interface-id?   string
|  |     +--rw lacp
|  |     |  +--rw lacp-state?         boolean
|  |     |  +--rw mode?               identityref
|  |     |  +--rw speed?              uint32
|  |     |  +--rw mini-link-num?      uint32
|  |     |  +--rw system-id?
|  |     |  |       yang:mac-address
|  |     |  +--rw admin-key?          uint16
|  |     |  +--rw system-priority?    uint16
|  |     |  +--rw member-link-list
|  |     |  |  +--rw member-link* [name]
|  |     |  |     +--rw name         string
|  |     |  |     +--rw speed?       uint32
|  |     |  |     +--rw mode?   identityref
```

```
|     |  |     +--rw link-mtu?    uint32
|     |  |     +--rw oam-802.3ah-link
|     |  |        |    {oam-3ah}?
|     |  |        +--rw enable?   boolean
|     |  +--rw flow-control?       boolean
|     |  +--rw lldp?               boolean
|     +--rw split-horizon
|        +--rw group-name?   string
...
```

Figure 15: Connection Subtree

7.6.2. EVPN-VPWS Service Instance

The 'vpws-service-instance' provides the local and remote VPWS Service Instance (VSI) [RFC8214]. This container is only present when the 'vpn-type' is VPWS-EVPN. As shown in Figure 16, the VSIs can be configured by a VPN service provider or auto-generated.

An example to illustrate the use of the L2NM to configure VPWS-EVPN instances is provided in Appendix A.4.

```
...
+--rw vpn-nodes
   +--rw vpn-node* [vpn-node-id]
      ...
      +--rw vpn-network-accesses
         +--rw vpn-network-access* [id]
            ...
            +--rw (signaling-option)?
            |  +--:(bgp)
            |     +--rw (bgp-type)?
            |        +--:(l2vpn-bgp)
            |        |  ...
            |        +--:(evpn-bgp)
            |           +--rw df-preference?     uint16
            |           +--rw vpws-service-instance
            |              +--rw (local-vsi-choice)?
            |              |  +--:(directly-assigned)
            |              |  |  +--rw local-vpws-service-instance?
            |              |  |          uint32
            |              |  +--:(auto-assigned)
            |              |     +--rw local-vsi-auto
            |              |        +--rw (auto-mode)?
            |              |        |  +--:(from-pool)
            |              |        |  |  +--rw vsi-pool-name?
            |              |        |  |          string
            |              |        |  +--:(full-auto)
            |              |        |     +--rw auto?      empty
            |              |        +--ro auto-local-vsi?  uint32
            |              +--rw (remote-vsi-choice)?
            |                 +--:(directly-assigned)
            |                 |  +--rw remote-vpws-service-instance?
            |                 |          uint32
            |                 +--:(auto-assigned)
            |                    +--rw remote-vsi-auto
            |                       +--rw (auto-mode)?
            |                       |  +--:(from-pool)
            |                       |  |  +--rw vsi-pool-name?
            |                       |  |          string
            |                       |  +--:(full-auto)
            |                       |     +--rw auto?       empty
            |                       +--ro auto-remote-vsi?  uint32
            |
            ...
```

Figure 16: EVPN-VPWS Service Instance Subtree

7.6.3. Ethernet OAM

Ethernet OAM refers to both [IEEE-802-1ag] and [ITU-T-Y-1731].

As shown in Figure 17, the L2NM inherits the same structure as in Section 5.3.2.2.6 of [RFC8466] for OAM matters.

```
+--rw l2vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
      +--rw vpn-service* [vpn-id]
         ...
         +--rw vpn-nodes
            +--rw vpn-node* [vpn-node-id]
               ...
               +--rw vpn-network-accesses
                  +--rw vpn-network-access* [id]
                     ...
                     +--rw ethernet-service-oam
                     |  +--rw md-name?        string
                     |  +--rw md-level?       uint8
                     |  +--rw cfm-802.1-ag
                     |  |  +--rw n2-uni-c* [maid]
                     |  |  |  +--rw maid                string
                     |  |  |  +--rw mep-id?             uint32
                     |  |  |  +--rw mep-level?          uint32
                     |  |  |  +--rw mep-up-down?
                     |  |  |  |                    enumeration
                     |  |  |  +--rw remote-mep-id?      uint32
                     |  |  |  +--rw cos-for-cfm-pdus?   uint32
                     |  |  |  +--rw ccm-interval?       uint32
                     |  |  |  +--rw ccm-holdtime?       uint32
                     |  |  |  +--rw ccm-p-bits-pri?
                     |  |  |            ccm-priority-type
                     |  |  +--rw n2-uni-n* [maid]
                     |  |     +--rw maid                string
                     |  |     +--rw mep-id?             uint32
                     |  |     +--rw mep-level?          uint32
                     |  |     +--rw mep-up-down?
                     |  |     |                     enumeration
                     |  |     +--rw remote-mep-id?      uint32
                     |  |     +--rw cos-for-cfm-pdus?   uint32
                     |  |     +--rw ccm-interval?       uint32
                     |  |     +--rw ccm-holdtime?       uint32
                     |  |     +--rw ccm-p-bits-pri?
                     |  |               ccm-priority-type
                     |  +--rw y-1731* [maid]
```

```
|     +--rw maid                string
|     +--rw mep-id?             uint32
|     +--rw pm-type?            identityref
|     +--rw remote-mep-id?      uint32
|     +--rw message-period?     uint32
|     +--rw measurement-interval?   uint32
|     +--rw cos?         uint32
|     +--rw loss-measurement?      boolean
|     +--rw synthethic-loss-measurement?
|     |       boolean
|     +--rw delay-measurement
|     |  +--rw enable-dm?   boolean
|     |  +--rw two-way?     boolean
|     +--rw frame-size?     uint32
|     +--rw session-type?   enumeration
...
```

Figure 17: OAM Subtree

7.6.4. Services

The 'service' container (Figure 18) provides a set of service-specific configuration such as Quality of Service (QoS).

```
+--rw l2vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
      +--rw vpn-service* [vpn-id]
         ...
         +--rw vpn-nodes
            +--rw vpn-node* [vpn-node-id]
               ...
               +--rw vpn-network-accesses
                  +--rw vpn-network-access* [id]
                     ...
                     +--rw service
                        +--rw mtu?            uint32
                        +--rw svc-pe-to-ce-bandwidth
                        |       {vpn-common:inbound-bw}?
                        |  ...
                        +--rw svc-ce-to-pe-bandwidth
                        |       {vpn-common:outbound-bw}?
                        |  ...
                        +--rw qos {vpn-common:qos}?
                        |  ...
                        +--rw mac-policies
                        |  ...
                        +--rw broadcast-unknown-unicast-multicast
                           ...
```

Figure 18: Service Overall Subtree

The description of the service data nodes is as follows:

'mtu': Specifies the Layer 2 MTU, in bytes, for the VPN network access.

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth': Specify the service bandwidth for the L2VPN service.

'svc-pe-to-ce-bandwidth' indicates the inbound bandwidth of the connection (i.e., download bandwidth from the service provider to the site).

'svc-ce-to-pe-bandwidth' indicates the outbound bandwidth of the connection (i.e., upload bandwidth from the site to the service provider).

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth' can be represented using the Committed Information Rate (CIR), the Excess Information Rate (EIR), or the Peak Information Rate (PIR).

As shown in Figure 19, the structure of service bandwidth data nodes is inherited from the L2SM [RFC8466]. The following types, defined in [RFC9181], can be used to indicate the bandwidth type:

'bw-per-cos': The bandwidth is per Class of Service (CoS).

'bw-per-port': The bandwidth is per VPN network access.

'bw-per-site': The bandwidth is to all VPN network accesses that belong to the same site.

'bw-per-service': The bandwidth is per L2VPN service.

```
+--rw service
   ...
   +--rw svc-pe-to-ce-bandwidth
   |       {vpn-common:inbound-bw}?
   |  +--rw pe-to-ce-bandwidth* [bw-type]
   |     +--rw bw-type      identityref
   |     +--rw (type)?
   |        +--:(per-cos)
   |        |  +--rw cos* [cos-id]
   |        |     +--rw cos-id    uint8
   |        |     +--rw cir?      uint64
   |        |     +--rw cbs?      uint64
   |        |     +--rw eir?      uint64
   |        |     +--rw ebs?      uint64
   |        |     +--rw pir?      uint64
   |        |     +--rw pbs?      uint64
   |        +--:(other)
   |           +--rw cir?   uint64
   |           +--rw cbs?   uint64
   |           +--rw eir?   uint64
   |           +--rw ebs?   uint64
   |           +--rw pir?   uint64
   |           +--rw pbs?   uint64
   +--rw svc-ce-to-pe-bandwidth
   |       {vpn-common:outbound-bw}?
   |  +--rw ce-to-pe-bandwidth* [bw-type]
   |     +--rw bw-type      identityref
   |     +--rw (type)?
   |        +--:(per-cos)
   |        |  +--rw cos* [cos-id]
   |        |     +--rw cos-id    uint8
   |        |     +--rw cir?      uint64
   |        |     +--rw cbs?      uint64
   |        |     +--rw eir?      uint64
   |        |     +--rw ebs?      uint64
   |        |     +--rw pir?      uint64
   |        |     +--rw pbs?      uint64
   |        +--:(other)
   |           +--rw cir?   uint64
   |           +--rw cbs?   uint64
   |           +--rw eir?   uint64
   |           +--rw ebs?   uint64
   |           +--rw pir?   uint64
   |           +--rw pbs?   uint64
   ...
```

Figure 19: Service Bandwidth Subtree

'qos': Is used to define a set of QoS policies to apply on a given VPN network access (Figure 20). The QoS classification can be based on many criteria such as source MAC address, destination MAC address, etc. See also Section 5.10.2.1 of [RFC8466] for more discussion of QoS classification including the use of color types.

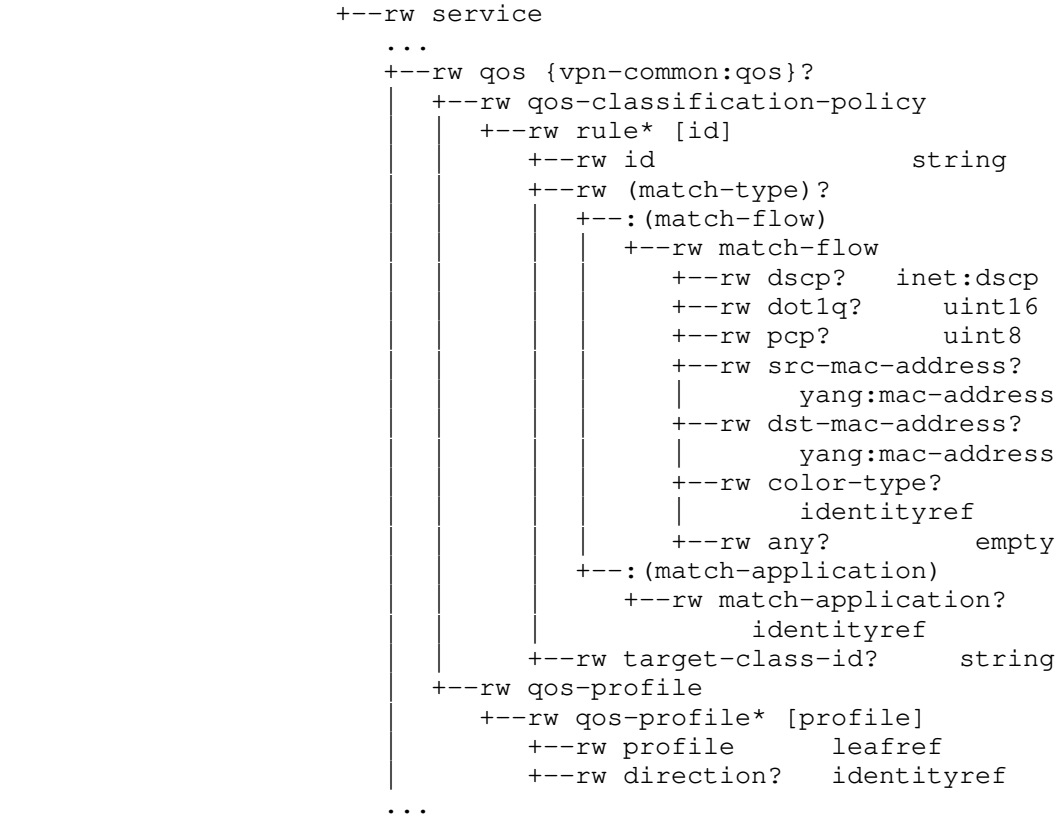


Figure 20: QoS Subtree

'mac-policies': Lists a set of MAC-related policies such as MAC ACLs. Similar to [RFC8519], an ACL match can be based upon source MAC address, source MAC address mask, destination MAC address, destination MAC address mask, or a combination thereof.

A data frame that matches an ACL can be dropped, flooded, or trigger an alarm. A rate-limit policy can be defined for handling frames that match an ACL entry with 'flood' action.

When 'mac-loop-prevention' or 'mac-addr-limit' data nodes are provided, they take precedence over the ones inlcuded in the 'global-parameters-profile' at the VPN service or VPN node levels.

```
+--rw service
   ...
   +--rw mac-policies
   |  +--rw access-control-list* [name]
   |  |  +--rw name                      string
   |  |  +--rw src-mac-address*
   |  |  |       yang:mac-address
   |  |  +--rw src-mac-address-mask*
   |  |  |       yang:mac-address
   |  |  +--rw dst-mac-address*
   |  |  |       yang:mac-address
   |  |  +--rw dst-mac-address-mask*
   |  |  |       yang:mac-address
   |  |  +--rw action?          identityref
   |  |  +--rw rate-limit?      decimal64
   |  +--rw mac-loop-prevention
   |  |  +--rw window?            uint32
   |  |  +--rw frequency?         uint32
   |  |  +--rw retry-timer?       uint32
   |  |  +--rw protection-type?   identityref
   |  +--rw mac-addr-limit
   |     +--rw limit-number?    uint16
   |     +--rw time-interval?   uint32
   |     +--rw action?          identityref
   ...
```

Figure 21: MAC Policies Subtree

'broadcast-unknown-unicast-multicast': Defines the type of site in the customer multicast service topology: source, receiver, or both. It is also used to define multicast group-to-port mappings.

```
+--rw service
   ...
   +--rw broadcast-unknown-unicast-multicast
      +--rw multicast-site-type?
      |       enumeration
      +--rw multicast-gp-address-mapping* [id]
      |  +--rw id                 uint16
      |  +--rw vlan-id            uint32
      |  +--rw mac-gp-address
      |  |       yang:mac-address
      |  +--rw port-lag-number?   uint32
      +--rw bum-overall-rate?     uint64
```

Figure 22: BUM Subtree

8. YANG Modules

8.1. IANA-Maintained Module for BGP Layer 2 Encapsulation Types

The "iana-bgp-l2-encaps" YANG module echoes the registry available at [IANA-BGP-L2].

This module references [RFC3032], [RFC4446], [RFC4448], [RFC4553], [RFC4618], [RFC4619], [RFC4717], [RFC4761], [RFC4816], [RFC4842], and [RFC5086].

```
<CODE BEGINS>
file "iana-bgp-l2-encaps@2021-07-05.yang"
module iana-bgp-l2-encaps {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps";
  prefix iana-bgp-l2-encaps;

  organization
    "IANA";
  contact
    "Internet Assigned Numbers Authority

     Postal: ICANN
          12025 Waterfront Drive, Suite 300
          Los Angeles, CA  90094-2536
          United States of America
     Tel:    +1 310 301 5800
     <mailto:iana@iana.org>";
  description
    "This module contains a collection of IANA-maintained YANG
     data types that are used for referring to BGP Layer 2
     encapsulation types.

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

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

     This version of this YANG module is part of RFC XXXX; see
```

```
       the RFC itself for full legal notices.";

    revision 2021-07-05 {
      description
        "First revision.";
      reference
        "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
    }

    identity bgp-l2-encaps-type {
      description
        "Base BGP Layer 2 encapsulation type.";
      reference
        "RFC 6624: Layer 2 Virtual Private Networks Using BGP for
                   Auto-Discovery and Signaling";
    }

    identity frame-relay {
      base bgp-l2-encaps-type;
      description
        "Frame Relay.";
      reference
        "RFC 4446: IANA Allocations for Pseudowire Edge
                   to Edge Emulation (PWE3)";
    }

    identity atm-aal5 {
      base bgp-l2-encaps-type;
      description
        "ATM AAL5 SDU VCC transport.";
      reference
        "RFC 4446: IANA Allocations for Pseudowire Edge
                   to Edge Emulation (PWE3)";
    }

    identity atm-cell {
      base bgp-l2-encaps-type;
      description
        "ATM transparent cell transport.";
      reference
        "RFC 4816: Pseudowire Emulation Edge-to-Edge (PWE3)
                   Asynchronous Transfer Mode (ATM) Transparent
                   Cell Transport Service";
    }

    identity ethernet-tagged-mode {
      base bgp-l2-encaps-type;
      description
```

```
        "Ethernet (VLAN) Tagged Mode.";
      reference
        "RFC 4448: Encapsulation Methods for Transport of Ethernet
                   over MPLS Networks";
    }

    identity ethernet-raw-mode {
      base bgp-l2-encaps-type;
      description
        "Ethernet Raw Mode.";
      reference
        "RFC 4448: Encapsulation Methods for Transport of Ethernet
                   over MPLS Networks";
    }

    identity hdlc {
      base bgp-l2-encaps-type;
      description
        "Cisco HDLC.";
      reference
        "RFC 4618: Encapsulation Methods for Transport of
                   PPP/High-Level Data Link Control (HDLC)
                   over MPLS Networks";
    }

    identity ppp {
      base bgp-l2-encaps-type;
      description
        "PPP.";
      reference
        "RFC 4618: Encapsulation Methods for Transport of
                   PPP/High-Level Data Link Control (HDLC)
                   over MPLS Networks";
    }

    identity circuit-emulation {
      base bgp-l2-encaps-type;
      description
        "SONET/SDH Circuit Emulation Service.";
      reference
        "RFC 4842: Synchronous Optical Network/Synchronous Digital
                   Hierarchy (SONET/SDH) Circuit Emulation over Packet
                   (CEP)";
    }

    identity atm-to-vcc {
      base bgp-l2-encaps-type;
      description
```

```
        "ATM n-to-one VCC cell transport.";
      reference
        "RFC 4717: Encapsulation Methods for Transport of
                   Asynchronous Transfer Mode (ATM) over MPLS
                   Networks";
    }

    identity atm-to-vpc {
      base bgp-l2-encaps-type;
      description
        "ATM n-to-one VPC cell transport.";
      reference
        "RFC 4717: Encapsulation Methods for Transport of
                   Asynchronous Transfer Mode (ATM) over MPLS
                   Networks";
    }

    identity layer-2-transport {
      base bgp-l2-encaps-type;
      description
        "IP Layer 2 Transport.";
      reference
        "RFC 3032: MPLS Label Stack Encoding";
    }

    identity fr-port-mode {
      base bgp-l2-encaps-type;
      description
        "Frame Relay Port mode.";
      reference
        "RFC 4619: Encapsulation Methods for Transport of Frame Relay
                   over Multiprotocol Label Switching (MPLS)
                   Networks";
    }

    identity e1 {
      base bgp-l2-encaps-type;
      description
        "Structure-agnostic E1 over packet.";
      reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity t1 {
      base bgp-l2-encaps-type;
      description
        "Structure-agnostic T1 (DS1) over packet.";
```

```
      reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity vpls {
      base bgp-l2-encaps-type;
      description
        "VPLS.";
      reference
        "RFC 4761: Virtual Private LAN Service (VPLS)
                   Using BGP for Auto-Discovery and Signaling";
    }

    identity t3 {
      base bgp-l2-encaps-type;
      description
        "Structure-agnostic T3 (DS3) over packet.";
      reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity structure-aware {
      base bgp-l2-encaps-type;
      description
        "Nx64kbit/s Basic Service using Structure-aware.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }

    identity dlci {
      base bgp-l2-encaps-type;
      description
        "Frame Relay DLCI.";
      reference
        "RFC 4619: Encapsulation Methods for Transport of Frame Relay
                   over Multiprotocol Label Switching (MPLS)
                   Networks";
    }

    identity e3 {
      base bgp-l2-encaps-type;
      description
        "Structure-agnostic E3 over packet.";
      reference
```

```
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity ds1 {
      base bgp-l2-encaps-type;
      description
        "Octet-aligned payload for Structure-agnostic DS1 circuits.";
      reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity cas {
      base bgp-l2-encaps-type;
      description
        "E1 Nx64kbit/s with CAS using Structure-aware.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }

    identity esf {
      base bgp-l2-encaps-type;
      description
        "DS1 (ESF) Nx64kbit/s with CAS using Structure-aware.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }

    identity sf {
      base bgp-l2-encaps-type;
      description
        "DS1 (SF) Nx64kbit/s with CAS using Structure-aware.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }
  }
  <CODE ENDS>
```

8.2. IANA-Maintained Module for Pseudowire Types

The initial version of the "iana-pseudowire-types" YANG module echoes the registry available at [IANA-PW-Types].

This module references [MFA], [RFC2507], [RFC2508], [RFC3032], [RFC3545], [RFC4448], [RFC4618], [RFC4619], [RFC4717], [RFC4842], [RFC4863], [RFC4901], [RFC5086], [RFC5087], [RFC5143], [RFC5795], and [RFC6307].

```
<CODE BEGINS>
file "iana-pseudowire-types@2021-07-05.yang"
module iana-pseudowire-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-pseudowire-types";
  prefix iana-pw-types;

  organization
    "IANA";
  contact
    "Internet Assigned Numbers Authority

     Postal: ICANN
          12025 Waterfront Drive, Suite 300
          Los Angeles, CA  90094-2536
          United States of America
     Tel:    +1 310 301 5800
     <mailto:iana@iana.org>";
  description
    "This module contains a collection of IANA-maintained YANG
     data types that are used for referring to Pseudowire Types.

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

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

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

  revision 2021-07-05 {
    description
      "First revision.";
```

```
    reference
      "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
  }

  identity iana-pw-types {
    description
      "Base Pseudowire Layer 2 encapsulation type.";
  }

  identity frame-relay {
    base iana-pw-types;
    description
      "Frame Relay DLCI (Martini Mode).";
    reference
      "RFC 4619: Encapsulation Methods for Transport of Frame Relay
                 over Multiprotocol Label Switching (MPLS)
                 Networks";
  }

  identity atm-aal5 {
    base iana-pw-types;
    description
      "ATM AAL5 SDU VCC transport.";
    reference
      "RFC 4717: Encapsulation Methods for Transport of
                 Asynchronous Transfer Mode (ATM) over MPLS
                 Networks";
  }

  identity atm-cell {
    base iana-pw-types;
    description
      "ATM transparent cell transport.";
    reference
      "RFC 4717: Encapsulation Methods for Transport of
                 Asynchronous Transfer Mode (ATM) over MPLS
                 Networks";
  }

  identity ethernet-tagged-mode {
    base iana-pw-types;
    description
      "Ethernet (VLAN) Tagged Mode.";
    reference
      "RFC 4448: Encapsulation Methods for Transport of Ethernet
                 over MPLS Networks";
  }
```

```
identity ethernet {
  base iana-pw-types;
  description
    "Ethernet.";
  reference
    "RFC 4448: Encapsulation Methods for Transport of Ethernet
               over MPLS Networks";
}

identity hdlc {
  base iana-pw-types;
  description
    "HDLC.";
  reference
    "RFC 4618: Encapsulation Methods for Transport of
               PPP/High-Level Data Link Control (HDLC)
               over MPLS Networks";
}

identity ppp {
  base iana-pw-types;
  description
    "PPP.";
  reference
    "RFC 4618: Encapsulation Methods for Transport of
               PPP/High-Level Data Link Control (HDLC)
               over MPLS Networks";
}

identity circuit-emulation-mpls {
  base iana-pw-types;
  description
    "SONET/SDH Circuit Emulation Service Over MPLS Encapsulation.";
  reference
    "RFC 5143: Synchronous Optical Network/Synchronous Digital
               Hierarchy (SONET/SDH) Circuit Emulation Service over
               MPLS (CEM) Encapsulation";
}

identity atm-to-vcc {
  base iana-pw-types;
  description
    "ATM n-to-one VCC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
               Asynchronous Transfer Mode (ATM) over MPLS
               Networks";
}
```

```
identity atm-to-vpc {
  base iana-pw-types;
  description
    "ATM n-to-one VPC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
               Asynchronous Transfer Mode (ATM) over MPLS
               Networks";
}

identity layer-2-transport {
  base iana-pw-types;
  description
    "IP Layer2 Transport.";
  reference
    "RFC 3032: MPLS Label Stack Encoding";
}

identity atm-one-to-one-vcc {
  base iana-pw-types;
  description
    "ATM one-to-one VCC Cell Mode.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
               Asynchronous Transfer Mode (ATM) over MPLS
               Networks";
}

identity atm-one-to-one-vpc {
  base iana-pw-types;
  description
    "ATM one-to-one VPC Cell Mode.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
               Asynchronous Transfer Mode (ATM) over MPLS
               Networks";
}

identity atm-aal5-vcc {
  base iana-pw-types;
  description
    "ATM AAL5 PDU VCC transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
               Asynchronous Transfer Mode (ATM) over MPLS
               Networks";
}
```

```
identity fr-port-mode {
  base iana-pw-types;
  description
    "Frame-Relay Port mode.";
  reference
    "RFC 4619: Encapsulation Methods for Transport of Frame Relay
               over Multiprotocol Label Switching (MPLS)
               Networks";
}

identity circuit-emulation-packet {
  base iana-pw-types;
  description
    "SONET/SDH Circuit Emulation over Packet.";
  reference
    "RFC 4842: Synchronous Optical Network/Synchronous Digital
               Hierarchy (SONET/SDH) Circuit Emulation over Packet
               (CEP)";
}

identity e1 {
  base iana-pw-types;
  description
    "Structure-agnostic E1 over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
               over Packet (SAToP)";
}

identity t1 {
  base iana-pw-types;
  description
    "Structure-agnostic T1 (DS1) over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
               over Packet (SAToP)";
}

identity e3 {
  base iana-pw-types;
  description
    "Structure-agnostic E3 over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
               over Packet (SAToP)";
}

identity t3 {
```

```
      base iana-pw-types;
      description
        "Structure-agnostic T3 (DS3) over Packet.";
      reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
                   over Packet (SAToP)";
    }

    identity ces-over-psn {
      base iana-pw-types;
      description
        "CESoPSN basic mode.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }

    identity tdm-over-ip-aal1 {
      base iana-pw-types;
      description
        "TDMoIP AAL1 Mode.";
      reference
        "RFC 5087: Time Division Multiplexing over IP (TDMoIP)";
    }

    identity ces-over-psn-cas {
      base iana-pw-types;
      description
        "CESoPSN TDM with CAS.";
      reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
                   Circuit Emulation Service over Packet Switched
                   Network (CESoPSN)";
    }

    identity tdm-over-ip-aal2 {
      base iana-pw-types;
      description
        "TDMoIP AAL2 Mode.";
      reference
        "RFC 5087: Time Division Multiplexing over IP (TDMoIP)";
    }

    identity dlci {
      base iana-pw-types;
      description
        "Frame Relay DLCI.";
```

```
      reference
        "RFC 4619: Encapsulation Methods for Transport of Frame Relay
                   over Multiprotocol Label Switching (MPLS)
                   Networks";
    }

    identity rohc {
      base iana-pw-types;
      description
        "ROHC Transport Header-compressed Packets.";
      reference
        "RFC 5795: The RObust Header Compression (ROHC) Framework
         RFC 4901: Protocol Extensions for Header Compression over
                   MPLS";
    }

    identity ecrtp {
      base iana-pw-types;
      description
        "ECRTP Transport Header-compressed Packets.";
      reference
        "RFC 3545: Enhanced Compressed RTP (CRTP) for Links with High
                   Delay, Packet Loss and Reordering
         RFC 4901: Protocol Extensions for Header Compression over
                   MPLS";
    }

    identity iphc {
      base iana-pw-types;
      description
        "IPHC Transport Header-compressed Packets.";
      reference
        "RFC 2507: IP Header Compression
         RFC 4901: Protocol Extensions for Header Compression over
                   MPLS";
    }

    identity crtp {
      base iana-pw-types;
      description
        "cRTP Transport Header-compressed Packets.";
      reference
        "RFC 2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial
                   Links
         RFC 4901: Protocol Extensions for Header Compression over
                   MPLS";
    }
```

```
    identity atm-vp-virtual-trunk {
      base iana-pw-types;
      description
        "ATM VP Virtual Trunk.";
      reference
        "MFA Forum: The Use of Virtual Trunks for ATM/MPLS
                    Control Plane Interworking Specification";
    }

    identity fc-port-mode {
      base iana-pw-types;
      description
        "FC Port Mode.";
      reference
        "RFC 6307: Encapsulation Methods for Transport of
                   Fibre Channel Traffic over MPLS Networks";
    }

    identity wildcard {
      base iana-pw-types;
      description
        "Wildcard.";
      reference
        "RFC 4863: Wildcard Pseudowire Type";
    }
  }
  <CODE ENDS>
```

8.3. Ethernet Segments

The "ietf-ethernet-segment" YANG module uses types defined in [RFC6991].

```
  <CODE BEGINS>
  file "ietf-ethernet-segment@2022-05-25.yang"
  module ietf-ethernet-segment {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-ethernet-segment";
    prefix l2vpn-es;

    import ietf-yang-types {
      prefix yang;
      reference
        "RFC 6991: Common YANG Data Types, Section 3";
    }

    organization
      "IETF OPSA (Operations and Management Area) Working Group";
```

```
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>

     Editor:    Mohamed Boucadair
                <mailto:mohamed.boucadair@orange.com>
     Editor:    Samier Barguil
                <mailto:samier.barguilgiraldo.ext@telefonica.com>
     Author:    Oscar Gonzalez de Dios
                <mailto:oscar.gonzalezdedios@telefonica.com>";
  description
    "This YANG module defines a model for Ethernet Segments.

     Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

  revision 2022-05-25 {
    description
      "Initial version.";
    reference
      "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
  }

  /* Typedefs */

  typedef es-ref {
    type leafref {
      path "/l2vpn-es:ethernet-segments/l2vpn-es:ethernet-segment"
         + "/l2vpn-es:name";
    }
    description
      "Defines a type for referencing an Ethernet segment in
       other modules.";
  }

  /* Identities */

  identity esi-type {
```

```
      description
        "T-(Ethernet Segment Identifier (ESI) Type) is a 1-octet field
         (most significant octet) that specifies the format of the
         remaining 9 octets (ESI Value).";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 5";
    }

    identity esi-type-0-operator {
      base esi-type;
      description
        "This type indicates an arbitrary 9-octet ESI value,
         which is managed and configured by the operator.";
    }

    identity esi-type-1-lacp {
      base esi-type;
      description
        "When IEEE 802.1AX Link Aggregation Control Protocol (LACP)
         is used between the Provider Edge (PE) and Customer Edge (CE)
         devices, this ESI type indicates an auto-generated ESI value
         determined from LACP.";
      reference
        "IEEE Std. 802.1AX: Link Aggregation";
    }

    identity esi-type-2-bridge {
      base esi-type;
      description
        "The ESI value is auto-generated and determined based
         on the Layer 2 bridge protocol.";
    }

    identity esi-type-3-mac {
      base esi-type;
      description
        "This type indicates a MAC-based ESI value that can be
         auto-generated or configured by the operator.";
    }

    identity esi-type-4-router-id {
      base esi-type;
      description
        "This type indicates a Router ID ESI value that can be
         auto-generated or configured by the operator.";
    }

    identity esi-type-5-asn {
```

```
    base esi-type;
    description
      "This type indicates an Autonomous System (AS)-based ESI value
       that can be auto-generated or configured by the operator.";
  }

  identity df-election-methods {
    description
      "Base Identity Designated Forwarder (DF) election method.";
  }

  identity default-7432 {
    base df-election-methods;
    description
      "The default DF election method.

       The default procedure for DF election at the granularity of
       <ES,VLAN> for VLAN-based service or <ES, VLAN bundle> for
       VLAN-(aware) bundle service is referred to as
       'service carving'.";
    reference
      "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.5";
  }

  identity highest-random-weight {
    base df-election-methods;
    description
      "The highest random weight (HRW) method.";
    reference
      "RFC 8584: Framework for Ethernet VPN Designated
                 Forwarder Election Extensibility, Section 3";
  }

  identity preference {
    base df-election-methods;
    description
      "The preference based method. PEs are assigned with
       preferences to become the DF in the Ethernet Segment (ES).
       The exact preference-based algorithm (e.g., lowest-preference
       algorithm, highest-preference algorithm) to use is
       signaled at the control plane.";
  }

  identity es-redundancy-mode {
    description
      "Base identity for ES redundancy modes.";
  }
```

```
identity single-active {
  base es-redundancy-mode;
  description
    "Indicates Single-Active redundancy mode for a given ES.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1.1";
}

identity all-active {
  base es-redundancy-mode;
  description
    "Indicates All-Active redundancy mode for a given ES.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1.2";
}

/* Main Ethernet Segment Container */

container ethernet-segments {
  description
    "Top container for the Ethernet Segment Identifier (ESI).";
  list ethernet-segment {
    key "name";
    description
      "Top list for ESIs.";
    leaf name {
      type string;
      description
        "Includes the name of the Ethernet Segment (ES) that
         is used to unambiguously identify an ES.";
    }
    leaf esi-type {
      type identityref {
        base esi-type;
      }
      default "esi-type-0-operator";
      description
        "T-(ESI Type) is a 1-octet field (most significant
         octet) that specifies the format of the remaining
         9 octets (ESI Value).";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 5";
    }
    choice esi-choice {
      description
        "Ethernet segment choice between several types.
         For ESI Type 0: The esi is directly configured by the
                         operator.
```

```
             For ESI Type 1: The auto-mode must be used.
             For ESI Type 2: The auto-mode must be used.
             For ESI Type 3: The directly-assigned or auto-mode must
                             be used.
             For ESI Type 4: The directly-assigned or auto-mode must
                             be used.
             For ESI Type 5: The directly-assigned or auto-mode must
                             be used.";
          case directly-assigned {
            description
              "Explicitly assign an ESI value.";
            leaf ethernet-segment-identifier {
              type yang:hex-string {
                length "29";
              }
              description
                "10-octet ESI.";
            }
          }
          case auto-assigned {
            description
              "The ESI is auto-assigned.";
            container esi-auto {
              description
                "The ESI is auto-assigned.";
              choice auto-mode {
                description
                  "Indicates the auto-assignment mode. ESI can be
                   automatically assigned either with or without
                   indicating a pool from which the ESI should be
                   taken.

                   For both cases, the server will auto-assign an
                   ESI value 'auto-assigned-ESI' and use that value
                   operationally.";
                case from-pool {
                  leaf esi-pool-name {
                    type string;
                    description
                      "The auto-assignment will be made from the
                       pool identified by the ESI-pool-name.";
                  }
                }
                case full-auto {
                  leaf auto {
                    type empty;
                    description
                      "Indicates an ESI is fully auto-assigned.";
```

```
                }
              }
            }
            leaf auto-ethernet-segment-identifier {
              type yang:hex-string {
                length "29";
              }
              config false;
              description
                "The value of the auto-assigned ESI.";
            }
          }
        }
        leaf esi-redundancy-mode {
          type identityref {
            base es-redundancy-mode;
          }
          description
            "Indicates the ES redundancy mode.";
          reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1";
        }
        container df-election {
          description
            "Top container for the DF election method properties.";
          leaf df-election-method {
            type identityref {
              base df-election-methods;
            }
            default "default-7432";
            description
              "Specifies the DF election method.";
            reference
              "RFC 8584: Framework for Ethernet VPN Designated
                         Forwarder Election Extensibility";
          }
          leaf revertive {
            when "derived-from-or-self(../df-election-method, "
               + "'preference')" {
              description
                "The revertive value is only applicable
                 to the preference method.";
            }
            type boolean;
            default "true";
            description
              "The default behavior is that the DF election
```

```
            procedure is triggered upon PE failures following
            configured preference values. Such a mode is called
            the revertive mode. This mode may not be suitable in
            some scenarios where, e.g., an operator may want to
            maintain the new DF even if the former DF recovers.
            Such a mode is called the 'non-revertive' mode.

            The non-revertive mode can be configured by
            setting 'revertive' leaf to 'false'.";
          reference
            "RFC 8584: Framework for Ethernet VPN Designated
                       Forwarder Election Extensibility,
                       Section 1.3.2";
        }
        leaf election-wait-time {
          type uint32;
          units "seconds";
          default "3";
          description
            "Election wait timer.";
          reference
            "RFC 8584: Framework for Ethernet VPN Designated
                       Forwarder Election Extensibility";
        }
      }
      leaf split-horizon-filtering {
        type boolean;
        description
          "Controls split-horizon filtering. It is enabled
           when set to 'true'.

           In order to achieve split-horizon filtering, every
           Broadcast, unknown unicast, or multicast (BUM)
           packet originating from a non-DF PE is encapsulated
           with an MPLS label that identifies the origin ES.";
        reference
          "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3";
      }
      container pbb {
        description
          "Provider Backbone Bridging (PBB) parameters .";
        reference
          "IEEE 802.1ah: Provider Backbone Bridge";
        leaf backbone-src-mac {
          type yang:mac-address;
          description
            "The PEs connected to the same CE must share the
             same Provider Backbone (B-MAC) address in
```

```
                All-Active mode.";
              reference
                "RFC 7623: Provider Backbone Bridging Combined with
                           Ethernet VPN (PBB-EVPN), Section 6.2.1.1";
            }
          }
          list member {
            key "ne-id interface-id";
            description
              "Includes a list of ES members.";
            leaf ne-id {
              type string;
              description
                "An identifier of the network element where the ES
                 is configured within a service provider network.";
            }
            leaf interface-id {
              type string;
              description
                "Identifier of a node interface.";
            }
          }
        }
      }
    }
    <CODE ENDS>
```

8.4. L2NM

The "ietf-l2vpn-ntw" YANG module uses types defined in [RFC6991], [RFC9181], [RFC8294], and [IEEE802.1Qcp-2018].

```
<CODE BEGINS>
file "ietf-l2vpn-ntw@2022-05-25.yang"
module ietf-l2vpn-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw";
  prefix l2vpn-ntw;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types, Section 4";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types, Section 3";
```

```
    }
    import ietf-vpn-common {
      prefix vpn-common;
      reference
        "RFC 9181: A Common YANG for Data Model for Layer 2
                   and Layer 3 VPNs";
    }
    import iana-bgp-l2-encaps {
      prefix iana-bgp-l2-encaps;
      reference
        "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
    }
    import iana-pseudowire-types {
      prefix iana-pw-types;
      reference
        "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
    }
    import ietf-ethernet-segment {
      prefix l2vpn-es;
      reference
        "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
    }
    import ietf-routing-types {
      prefix rt-types;
      reference
        "RFC 8294: Common YANG Data Types for the Routing Area";
    }
    import ieee802-dot1q-types {
      prefix dot1q-types;
      reference
        "IEEE Std 802.1Qcp-2018: Bridges and Bridged Networks -
                                 Amendment: YANG Data Model";
    }

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

       Editor:    Mohamed Boucadair
                  <mailto:mohamed.boucadair@orange.com>
       Editor:    Samier Barguil
                  <mailto:samier.barguilgiraldo.ext@telefonica.com>
       Author:    Oscar Gonzalez de Dios
                  <mailto:oscar.gonzalezdedios@telefonica.com>";
    description
      "This YANG module defines a network model for Layer 2 VPN
```

```
      services.

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

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

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

    revision 2022-05-25 {
      description
        "Initial version.";
      reference
        "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
    }

    /* Features */

    feature oam-3ah {
      description
        "Indicates the support of OAM 802.3ah.";
      reference
        "IEEE Std 802.3ah: Media Access Control Parameters, Physical
                           Layers, and  Management Parameters for
                           Subscriber Access Networks";
    }

    /* Identities */

    identity evpn-service-interface-type {
      description
        "Base identity for EVPN service interface type.";
    }

    identity vlan-based-service-interface {
      base evpn-service-interface-type;
      description
        "VLAN-Based Service Interface.";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.1";
    }
```

```
identity vlan-bundle-service-interface {
  base evpn-service-interface-type;
  description
    "VLAN Bundle Service Interface.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.2";
}

identity vlan-aware-bundle-service-interface {
  base evpn-service-interface-type;
  description
    "VLAN-Aware Bundle Service Interface.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.3";
}

identity mapping-type {
  base vpn-common:multicast-gp-address-mapping;
  description
    "Identity for multicast group mapping type.";
}

identity loop-prevention-type {
  description
    "Identity of loop prevention.";
}

identity shut {
  base loop-prevention-type;
  description
    "Shut protection type.";
}

identity trap {
  base loop-prevention-type;
  description
    "Trap protection type.";
}

identity color-type {
  description
    "Identity of color types. A type is assigned to a service frame
     to identify its QoS profile conformance.";
}

identity green {
  base color-type;
  description
```

```
        "'green' color type.  A service frame is 'green' if it is
         conformant with the committed rate of the bandwidth profile.";
    }

    identity yellow {
      base color-type;
      description
        "'yellow' color type. A service frame is 'yellow' if it exceeds
          the committed rate but is conformant with the excess rate
         of the bandwidth profile.";
    }

    identity red {
      base color-type;
      description
        "'red' color type. A service famre is 'red' if it is not
         conformant with both the committed and excess rates of the
         bandwidth profile.";
    }

    identity t-ldp-pw-type {
      description
        "Identity for t-ldp-pw-type.";
    }

    identity vpws-type {
      base t-ldp-pw-type;
      description
        "Virtual Private Wire Service (VPWS) t-ldp-pw-type.";
      reference
        "RFC 4664: Framework for Layer 2 Virtual Private Networks
                   (L2VPNs), Section 3.3";
    }

    identity vpls-type {
      base t-ldp-pw-type;
      description
        "Virtual Private LAN Service (VPLS) t-ldp-pw-type.";
      reference
        "RFC 4762: Virtual Private LAN Service (VPLS) Using
                    Label Distribution Protocol (LDP)
                    Signaling, Section 6.1";
    }

    identity hvpls {
      base t-ldp-pw-type;
      description
        "Identity for Hierarchical Virtual Private LAN Service (H-VPLS)
```

```
          t-ldp-pw-type.";
        reference
          "RFC 4762: Virtual Private LAN Service (VPLS) Using
                     Label Distribution Protocol (LDP)
                     Signaling, Section 10";
      }

      identity lacp-mode {
        description
          "Identity of the LACP mode.";
      }

      identity lacp-active {
        base lacp-mode;
        description
          "LACP active mode.

           This mode refers to the mode where auto-speed negotiation
           is initiated followed by an establishment of an
           Ethernet channel with the other end.";
      }

      identity lacp-passive {
        base lacp-mode;
        description
          "LACP passive mode.

           This mode refers to the LACP mode where an endpoint does
           not initiate the negotiation, but only responds to LACP
           packets initiated by the other end (e.g., full duplex
           or half duplex)";
      }

      identity pm-type {
        description
          "Identity for performance monitoring type.";
      }

      identity loss {
        base pm-type;
        description
          "Loss measurement is the performance monitoring type.";
      }

      identity delay {
        base pm-type;
        description
          "Delay measurement is the performance monitoring type.";
```

```
}

identity mac-learning-mode {
  description
    "Media Access Control (MAC) learning mode.";
}

identity data-plane {
  base mac-learning-mode;
  description
    "User MAC addresses are learned through ARP broadcast.";
}

identity control-plane {
  base mac-learning-mode;
  description
    "User MAC addresses are advertised through EVPN-BGP.";
}

identity mac-action {
  description
    "Base identity for a MAC action.";
}

identity drop {
  base mac-action;
  description
    "Dropping a packet as the MAC action.";
}

identity flood {
  base mac-action;
  description
    "Packet flooding as the MAC action.";
}

identity warning {
  base mac-action;
  description
    "Log a warning message as the MAC action.";
}

identity precedence-type {
  description
    "Redundancy type. The service can be created
     with primary and secondary signalization.";
}
```

```
  identity primary {
    base precedence-type;
    description
      "Identifies the main VPN network access.";
  }

  identity secondary {
    base precedence-type;
    description
      "Identifies the secondary VPN network access.";
  }

  identity ldp-pw-type {
    description
      "Identity for allowed LDP-based pseudowire (PW) type.";
    reference
      "RFC 4762: Virtual Private LAN Service (VPLS) Using
                 Label Distribution Protocol (LDP)
                 Signaling, Section 6.1.1";
  }

  identity ethernet {
    base ldp-pw-type;
    description
      "PW Ethernet type.";
  }

  identity ethernet-tagged {
    base ldp-pw-type;
    description
      "PW Ethernet tagged mode type.";
  }

  /* Typedefs */

  typedef ccm-priority-type {
    type uint8 {
      range "0..7";
    }
    description
      "A 3-bit priority value to be used in the VLAN tag,
       if present in the transmitted frame. A larger value
       indicates a higher priority.";
  }

  /* Groupings */

  grouping cfm-802 {
```

```
      description
        "Grouping for 802.1ag Connectivity Fault Management (CFM)
         attributes.";
      reference
        "IEEE Std 802-1ag: Virtual Bridged Local Area Networks
                           Amendment 5: Connectivity Fault Management";
      leaf maid {
        type string;
        description
          "Maintenance Association Identifier (MAID).";
      }
      leaf mep-id {
        type uint32;
        description
          "Local Maintenance Entity Group End Point (MEP) ID.";
      }
      leaf mep-level {
        type uint32;
        description
          "MEP level.";
      }
      leaf mep-up-down {
        type enumeration {
          enum up {
            description
              "MEP is up.";
          }
          enum down {
            description
              "MEP is down.";
          }
        }
        default "up";
        description
          "MEP up/down.";
      }
      leaf remote-mep-id {
        type uint32;
        description
          "Remote MEP ID.";
      }
      leaf cos-for-cfm-pdus {
        type uint32;
        description
          "Class of service for CFM PDUs.";
      }
      leaf ccm-interval {
        type uint32;
```

```
        units "milliseconds";
        default "10000";
        description
          "Continuity Check Message (CCM) interval.";
      }
      leaf ccm-holdtime {
        type uint32;
        units "milliseconds";
        default "35000";
        description
          "CCM hold time.";
      }
      leaf ccm-p-bits-pri {
        type ccm-priority-type;
        description
          "The priority parameter for Continuity Check Messages (CCMs)
           transmitted by the MEP.";
      }
    }

    grouping y-1731 {
      description
        "Grouping for Y-1731";
      reference
        "ITU-T Y-1731:  Operations, administration and maintenance
                        (OAM) functions and mechanisms for
                        Ethernet-based networks";
      list y-1731 {
        key "maid";
        description
          "List of configured Y-1731 instances.";
        leaf maid {
          type string;
          description
            "MAID.";
        }
        leaf mep-id {
          type uint32;
          description
            "Local MEP ID.";
        }
        leaf pm-type {
          type identityref {
            base pm-type;
          }
          default "delay";
          description
            "Performance monitor types.";
```

```
        }
        leaf remote-mep-id {
          type uint32;
          description
            "Remote MEP ID.";
        }
        leaf message-period {
          type uint32;
          units "milliseconds";
          default "10000";
          description
            "Defines the interval between OAM messages.";
        }
        leaf measurement-interval {
          type uint32;
          units "seconds";
          description
            "Specifies the measurement interval for statistics.";
        }
        leaf cos {
          type uint32;
          description
            "Identifies the Class of Service.";
        }
        leaf loss-measurement {
          type boolean;
          default "false";
          description
            "Controls whether loss measurement is (’true’) or
             disabled (’false’).";
        }
        leaf synthethic-loss-measurement {
          type boolean;
          default "false";
          description
            "Indicates whether synthetic loss measurement is enabled
             (’true’) or disabled (’false’).";
        }
        container delay-measurement {
          description
            "Container for delay measurement";
          leaf enable-dm {
            type boolean;
            default "false";
            description
              "Controls whether delay measurement is enabled (’true’)
               or disabled (’false’).";
          }
```

```
          leaf two-way {
            type boolean;
            default "false";
            description
              "Whether delay measurement is two-way ('true') of one-
               way ('false').";
          }
        }
        leaf frame-size {
          type uint32;
          units "bytes";
          description
            "Indicates the frame size.";
        }
        leaf session-type {
          type enumeration {
            enum proactive {
              description
                "Proactive mode.";
            }
            enum on-demand {
              description
                "On-demand mode.";
            }
          }
          default "on-demand";
          description
            "Specifies the session type.";
        }
      }
    }

    grouping parameters-profile {
      description
        "Container for per-service parameters.";
      leaf local-autonomous-system {
        type inet:as-number;
        description
          "Indicates a local AS Number (ASN).";
      }
      leaf svc-mtu {
        type uint32;
        units "bytes";
        description
          "Layer 2 service MTU.
           It is also known as the maximum transmission
           unit or maximum frame size.";
      }
```

```
      leaf ce-vlan-preservation {
        type boolean;
        description
          "Preserve the CE-VLAN ID from ingress to egress, i.e.,
           CE-VLAN tag of the egress frame is identical to
           that of the ingress frame that yielded this egress
           service frame. If all-to-one bundling within a site
           is enabled, then preservation applies to all ingress
           service frames. If all-to-one bundling is disabled,
           then preservation applies to tagged ingress service
           frames having CE-VLAN ID 1 through 4094.";
      }
      leaf ce-vlan-cos-preservation {
        type boolean;
        description
          "CE VLAN CoS preservation. Priority Code Point (PCP) bits
           in the CE-VLAN tag of the egress frame are identical to
           those of the ingress frame that yielded this egress
           service frame.";
      }
      leaf control-word-negotiation {
        type boolean;
        description
          "Controls whether Control-word negotiation is enabled
           (if set to true) or not (if set to false).";
        reference
          "RFC 8077: Pseudowire Setup and Maintenance
                     Using the Label Distribution Protocol (LDP),
                     Section 7";
      }
      container mac-policies {
        description
          "Container of MAC policies.";
        container mac-addr-limit {
          description
            "Container of MAC address limit configuration.";
          leaf limit-number {
            type uint16;
            description
              "Maximum number of MAC addresses learned from
               the customer for a single service instance.
               The default value is '2' when this grouping
               is used at the service level.";
          }
          leaf time-interval {
            type uint32;
            units "milliseconds";
            description
```

```
            "The aging time of the mac address.
             The default value is '300' when this grouping
             is used at the service level.";
          }
          leaf action {
            type identityref {
              base mac-action;
            }
            description
              "Specifies the action when the upper limit is
               exceeded: drop the packet, flood the packet,
               or log a warning message (without dropping
               the packet).
               The default value is 'warning' when this
               grouping is used at the service level.";
          }
        }
        container mac-loop-prevention {
          description
            "Container for MAC loop prevention.";
          leaf window {
            type uint32;
            units "seconds";
            description
              "The time interval over which a MAC mobility event
               is detected and checked.
               The default value is '180' when this grouping
               is used at the service level.";
          }
          leaf frequency {
            type uint32;
            description
              "The number of times to detect MAC duplication, where
               a 'duplicate MAC address' situation has occurred
               within the 'window' time interval and the duplicate
               MAC address has been added to a list of duplicate
               MAC addresses.
               The default value is '5' when this grouping is
               called at the service level.";
          }
          leaf retry-timer {
            type uint32;
            units "seconds";
            description
              "The retry timer. When the retry timer expires,
               the duplicate MAC address will be flushed from
               the MAC-VRF.";
          }
```

```
          leaf protection-type {
            type identityref {
              base loop-prevention-type;
            }
            description
              "Protection type.
               The default value is 'trap' when this grouping
               is used at the service level.";
          }
        }
      }
      container multicast {
        if-feature "vpn-common:multicast";
        description
          "Multicast container.";
        leaf enabled {
          type boolean;
          default "false";
          description
            "Enables multicast.";
        }
        container customer-tree-flavors {
          description
            "Type of trees used by the customer.";
          leaf-list tree-flavor {
            type identityref {
              base vpn-common:multicast-tree-type;
            }
            description
              "Type of multicast tree to be used.";
          }
        }
      }
    }

    grouping bandwidth-parameters {
      description
        "A grouping for bandwidth parameters.";
      leaf cir {
        type uint64;
        units "bps";
        description
          "Committed Information Rate. The maximum
           number of bits that a port can receive or
           send during one-second over an
           interface.";
      }
      leaf cbs {
```

```
        type uint64;
        units "bytes";
        description
          "Committed Burst Size. CBS controls the
           bursty nature of the traffic. Traffic
           that does not use the configured CIR
           accumulates credits until the credits
           reach the configured CBS.";
      }
      leaf eir {
        type uint64;
        units "bps";
        description
          "Excess Information Rate, i.e., excess
           frame delivery allowed not subject to
           SLA. The traffic rate can be limited
           by EIR.";
      }
      leaf ebs {
        type uint64;
        units "bytes";
        description
          "Excess Burst Size. 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 sum of CIR and EIR.";
      }
      leaf pbs {
        type uint64;
        units "bytes";
        description
          "Peak Burst Size.";
      }
    }

    /* Main L2NM Container */

    container l2vpn-ntw {
```

```
      description
        "Container for the L2NM.";
      container vpn-profiles {
        description
          "Container for VPN profiles.";
        uses vpn-common:vpn-profile-cfg;
      }
      container vpn-services {
        description
          "Container for L2VPN services.";
        list vpn-service {
          key "vpn-id";
          description
            "Container of a VPN service.";
          uses vpn-common:vpn-description;
          leaf parent-service-id {
            type vpn-common:vpn-id;
            description
              "Pointer to the parent service that
               triggered the L2NM.";
          }
          leaf vpn-type {
            type identityref {
              base vpn-common:service-type;
            }
            must "not(derived-from-or-self(current(), "
               + "'vpn-common:l3vpn'))" {
              error-message "L3VPN is only applicable in L3NM.";
            }
            description
              "Service type.";
          }
          leaf vpn-service-topology {
            type identityref {
              base vpn-common:vpn-topology;
            }
            description
              "Defining service topology, such as
               any-to-any, hub-spoke, etc.";
          }
          leaf bgp-ad-enabled {
            type boolean;
            description
              "Indicates whether BGP auto-discovery is enabled
               or disabled.";
          }
          leaf signaling-type {
            type identityref {
```

```
            base vpn-common:vpn-signaling-type;
          }
          description
            "VPN signaling type.";
        }
        container global-parameters-profiles {
          description
            "Container for a list of global parameters
             profiles.";
          list global-parameters-profile {
            key "profile-id";
            description
              "List of global parameters profiles.";
            leaf profile-id {
              type string;
              description
                "The identifier of the global parameters profile.";
            }
            uses vpn-common:route-distinguisher;
            uses vpn-common:vpn-route-targets;
            uses parameters-profile;
          }
        }
        container underlay-transport {
          description
            "Container for the underlay transport.";
          uses vpn-common:underlay-transport;
        }
        uses vpn-common:service-status;
        container vpn-nodes {
          description
            "Set of VPN nodes that are involved in the L2NM.";
          list vpn-node {
            key "vpn-node-id";
            description
              "Container of the VPN nodes.";
            leaf vpn-node-id {
              type vpn-common:vpn-id;
              description
                "Sets the identifier of the VPN node.";
            }
            leaf description {
              type string;
              description
                "Textual description of a VPN node.";
            }
            leaf ne-id {
              type string;
```

```
      description
        "An identifier of the network element where
         the VPN node is deployed. This identifier
         uniquely identifies the network element within
         an administrative domain.";
    }
    leaf role {
      type identityref {
        base vpn-common:role;
      }
      default "vpn-common:any-to-any-role";
      description
        "Role of the VPN node in the VPN.";
    }
    leaf router-id {
      type rt-types:router-id;
      description
        "A 32-bit number in the dotted-quad format that is
         used to uniquely identify a node within an
         autonomous system (AS).";
    }
    container active-global-parameters-profiles {
      description
        "Container for a list of global parameters
         profiles.";
      list global-parameters-profile {
        key "profile-id";
        description
          "List of active global parameters profiles.";
        leaf profile-id {
          type leafref {
            path "../../../../../global-parameters-profiles"
               + "/global-parameters-profile/profile-id";
          }
          description
            "Points to a global profile defined at the
             service level.";
        }
        uses parameters-profile;
      }
    }
    uses vpn-common:service-status;
    container bgp-auto-discovery {
       when "../../../bgp-ad-enabled = 'true'" {
        description
          "Only applies when BGP auto-discovery is enabled.";
      }
      description
```

```
        "BGP is used for auto-discovery.";
      choice bgp-type {
        description
          "Choice for the BGP type.";
        case l2vpn-bgp {
          description
            "Container for BGP L2VPN.";
          leaf vpn-id {
            type vpn-common:vpn-id;
            description
              "VPN Identifier. This identifier serves to
               unify components of a given VPN for the
               sake of auto-discovery.";
            reference
              "RFC 6624: Layer 2 Virtual Private Networks
                         Using BGP for Auto-Discovery and
                         Signaling";
          }
        }
        case evpn-bgp {
          description
            "EVPN case.";
          leaf evpn-type {
            type leafref {
              path "../../../../vpn-type";
            }
            description
              "EVPN type.";
          }
          leaf auto-rt-enable {
            type boolean;
            default "false";
            description
              "Enables/disabled RT auto-derivation based on
               the ASN and Ethernet Tag ID.";
            reference
              "RFC 7432: BGP MPLS-Based Ethernet VPN,
                         Section 7.10.1";
          }
          leaf auto-route-target {
            when "../auto-rt-enable = 'true'" {
              description
                "Can only be used when auto-RD is enabled.";
            }
            type rt-types:route-target;
            config false;
            description
              "The value of the auto-assigned RT.";
```

```
        }
      }
    }
    uses vpn-common:route-distinguisher;
    uses vpn-common:vpn-route-targets;
  }
  container signaling-option {
    description
      "Container for the L2VPN signaling.";
    leaf advertise-mtu {
      type boolean;
      description
        "Controls whether MTU is advertised.";
      reference
        "RFC 4667: Layer 2 Virtual Private Network (L2VPN)
                   Extensions for Layer 2 Tunneling
                   Protocol (L2TP), Section 4.3";
    }
    leaf mtu-allow-mismatch {
      type boolean;
      description
        "When set to true, it allows MTU mismatch.";
      reference
        "RFC 4667: Layer 2 Virtual Private Network (L2VPN)
                   Extensions for Layer 2 Tunneling
                   Protocol (L2TP), Section 4.3";
    }
    leaf signaling-type {
      type leafref {
        path "../../../../signaling-type";
      }
      description
        "VPN signaling type.";
    }
    choice signaling-option {
      description
        "Choice for the signaling-option.";
      case bgp {
        description
          "BGP is used as the signaling protocol.";
        choice bgp-type {
          description
            "Choice for the BGP type.";
          case l2vpn-bgp {
            description
              "Container for BGP L2VPN.";
            leaf ce-range {
              type uint16;
```

```
      description
        "Determines the number of remote CEs with
         which a given CE can communicate in the
          context of a VPN.";
      reference
        "RFC 6624: Layer 2 Virtual Private Networks
                   Using BGP for Auto-Discovery and
                   Signaling";
    }
    leaf pw-encapsulation-type {
      type identityref {
        base iana-bgp-l2-encaps:bgp-l2-encaps-type;
      }
      description
        "PW encapsulation type.";
    }
    container vpls-instance {
      when "derived-from-or-self(../../../../"
         + "vpn-type, 'vpn-common:vpls')" {
        description
          "Only applies for VPLS.";
      }
      description
        "VPLS instance.";
      leaf vpls-edge-id {
        type uint16;
        description
          "VPLS Edge Identifier (VE ID). This is
           used when the same VE ID is configured
           for the PE.";
        reference
          "RFC 4761: Virtual Private LAN Service
                     (VPLS) Using BGP for Auto-
                     Discovery and Signaling,
                     Section 3.5";
      }
      leaf vpls-edge-id-range {
        type uint16;
        description
          "Specifies the size of the range of
           VE ID in a VPLS service. The range
           controls the size of the label
           block advertised in the context of
           a VPLS instance.";
        reference
          "RFC 4761: Virtual Private LAN Service
                     (VPLS) Using BGP for Auto-
                     Discovery and Signaling";
```

```
        }
      }
    }
    case evpn-bgp {
      description
        "Used for EVPN.";
      leaf evpn-type {
        type leafref {
          path "../../bgp-auto-discovery/evpn-type";
        }
        description
          "EVPN type.";
      }
      leaf service-interface-type {
        type identityref {
          base evpn-service-interface-type;
        }
        description
          "EVPN service interface type.";
      }
      container evpn-policies {
        description
          "Includes a set of EVPN policies such
           as those related to handling MAC
           addresses.";
        leaf mac-learning-mode {
          type identityref {
            base mac-learning-mode;
          }
          description
            "Indicates through which plane MAC
             addresses are advertised.";
        }
        leaf ingress-replication {
          type boolean;
          description
            "Controls whether ingress replication is
             enabled ('true') or disabled ('false').";
          reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN,
                       Section 8.3.1.1";
        }
        leaf p2mp-replication {
          type boolean;
          description
            "Controles whether P2MP replication is
             enabled ('true') or disabled ('false')";
          reference
```

```
        "RFC 7432: BGP MPLS-Based Ethernet VPN,
                   Section 8.3.1.2";
    }
    container arp-proxy {
      if-feature "vpn-common:ipv4";
      description
        "Top container for the ARP proxy.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or
           disables (when set to 'false')
           ARP proxy.";
        reference
          "RFC 7432: BGP MPLS-Based Ethernet VPN,
                     Section 10";
      }
      leaf arp-suppression {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or
           disables (when set to 'false') ARP
           suppression.";
        reference
          "RFC 7432: BGP MPLS-Based Ethernet
                     VPN";
      }
      leaf ip-mobility-threshold {
        type uint16;
        description
          "It is possible for a given host (as
           defined by its IP address) to move
           from one ES  to another.
           IP mobility threshold specifies the
           number of IP mobility events
           that are detected for a given IP
           address within the
           detection-threshold before it
           is identified as a duplicate IP
           address.
           Once the detection threshold is
           reached, updates for the IP address
           are suppressed.";
      }
      leaf duplicate-ip-detection-interval {
        type uint16;
```

```
        units "seconds";
        description
          "The time interval used in detecting a
           duplicate IP address. Duplicate IP
           address detection number of host moves
           are allowed within this interval
           period.";
      }
    }
    container nd-proxy {
      if-feature "vpn-common:ipv6";
      description
        "Top container for the ND proxy.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or
           disables (when set to 'false') ND
           proxy.";
        reference
          "RFC 7432: BGP MPLS-Based Ethernet VPN,
                     Section 10";
      }
      leaf nd-suppression {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or
           disables (when set to 'false')
           Neighbor Discovery (ND) message
           suppression.
           ND suppression is a technique that
           is used to reduce the amount of ND
           packets flooding within individual
           segments, that is between hosts
           connected to the same logical
           switch.";
      }
      leaf ip-mobility-threshold {
        type uint16;
        description
          "It is possible for a given host (as
           defined by its IP address) to move
           from one ES to another.
           IP mobility threshold specifies the
           number of IP mobility events
           that are detected for a given IP
```

```
            address within the
            detection-threshold before it
            is identified as a duplicate IP
            address.
            Once the detection threshold is
            reached, updates for the IP address
            are suppressed.";
        }
        leaf duplicate-ip-detection-interval {
          type uint16;
          units "seconds";
          description
            "The time interval used in detecting a
             duplicate IP address. Duplicate IP
             address detection number of host moves
             are allowed within this interval
             period.";
        }
      }
      leaf underlay-multicast {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or disables
           (when set to 'false') underlay
           multicast.";
      }
      leaf flood-unknown-unicast-supression {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or disables
           (when set to 'false') unknown flood
           unicast suppression.";
      }
      leaf vpws-vlan-aware {
        type boolean;
        default "false";
        description
          "Enables (when set to 'true') or disables
           (when set to 'false') VPWS VLAN-aware.";
      }
      container bum-management {
        description
          "Broadcast-unknown-unicast-multicast
           management.";
        leaf discard-broadcast {
          type boolean;
```

```
                default "false";
                description
                  "Discards broadcast, when enabled.";
              }
              leaf discard-unknown-multicast {
                type boolean;
                default "false";
                description
                  "Discards unknown multicast, when
                   enabled.";
              }
              leaf discard-unknown-unicast {
                type boolean;
                default "false";
                description
                  "Discards unknown unicast, when
                   enabled.";
              }
            }
            container pbb {
              when "derived-from-or-self("
                 + "../../evpn-type, 'pbb-evpn')" {
                description
                  "Only applies for PBB EVPN.";
              }
              description
                "PBB parameters container.";
              reference
                "IEEE 802.1ah: Provider Backbone Bridge";
              leaf backbone-src-mac {
                type yang:mac-address;
                description
                  "Includes provider backbone MAC (B-MAC)
                   address.";
                reference
                  "RFC 7623: Provider Backbone Bridging
                             Combined with Ethernet VPN
                             (PBB-EVPN), Section 8.1";
              }
            }
          }
        }
      }
      container ldp-or-l2tp {
        description
          "Container for LDP or L2TP-signaled PWs
           choice.";
```

```
                leaf agi {
                  type rt-types:route-distinguisher;
                  description
                    "Attachment Group Identifier. Also, called
                     VPLS-Id.";
                  reference
                    "RFC 4667: Layer 2 Virtual Private Network
                               (L2VPN) Extensions for Layer 2
                               Tunneling Protocol (L2TP),
                               Section 4.3
                     RFC 4762: Virtual Private LAN Service (VPLS)
                               Using Label Distribution Protocol
                               (LDP) Signaling, Section 6.1.1";
                }
                leaf saii {
                  type uint32;
                  description
                    "Source Attachment Individual Identifier
                     (SAII).";
                  reference
                    "RFC 4667: Layer 2 Virtual Private Network
                               (L2VPN) Extensions for Layer 2
                               Tunneling Protocol (L2TP),
                               Section 3";
                }
                list remote-targets {
                  key "taii";
                  description
                    "List of allowed target Attachment Individual
                     Identifier (AII) and peers.";
                  reference
                    "RFC 4667: Layer 2 Virtual Private Network
                               (L2VPN) Extensions for Layer 2
                               Tunneling Protocol (L2TP),
                               Section 5";
                  leaf taii {
                    type uint32;
                    description
                      "Target Attachment Individual Identifier.";
                    reference
                      "RFC 4667: Layer 2 Virtual Private Network
                                 (L2VPN) Extensions for Layer 2
                                 Tunneling  Protocol (L2TP),
                                 Section 3";
                  }
                  leaf peer-addr {
                    type inet:ip-address;
                    description
```

```
            "Indicates the peer forwarder's IP address.";
        }
      }
      choice ldp-or-l2tp {
        description
          "Choice of LDP or L2TP-signaled PWs.";
        case ldp {
          description
            "Container for T-LDP PW configurations.";
          leaf t-ldp-pw-type {
            type identityref {
              base t-ldp-pw-type;
            }
            description
              "T-LDP PW type.";
          }
          leaf pw-type {
            type identityref {
              base ldp-pw-type;
            }
            description
              "PW encapsulation type.";
            reference
              "RFC 4762: Virtual Private LAN Service
                         (VPLS) Using Label Distribution
                         Protocol (LDP) Signaling,
                         Section 6.1.1";
          }
          leaf pw-description {
            type string;
            description
              "Includes a human-readable description
               of the interface. This may be used when
               communicating with a remote peer.";
            reference
              "RFC 4762: Virtual Private LAN Service
                         (VPLS) Using Label Distribution
                         Protocol (LDP) Signaling,
                         Section 6.1.1";
          }
          leaf mac-addr-withdraw {
            type boolean;
            description
              "If set to 'true', then MAC address
               withdrawal is enabled.  If 'false',
               then MAC address withdrawal is
               disabled.";
            reference
```

```
      "RFC 4762: Virtual Private LAN Service
                 (VPLS) Using Label Distribution
                 Protocol (LDP) Signaling,
                 Section 6.2";
    }
    list pw-peer-list {
      key "peer-addr vc-id";
      description
        "List of AC and PW bindings.";
      leaf peer-addr {
        type inet:ip-address;
        description
          "Indicates the peer's IP address.";
      }
      leaf vc-id {
        type string;
        description
          "VC label used to identify a PW.";
      }
      leaf pw-priority {
        type uint32;
        description
          "Defines the priority for the PW.
           The higher the pw-priority value, the
           higher the preference of the PW will
           be.";
      }
    }
    container qinq {
      when "derived-from-or-self("
         + "../t-ldp-pw-type, 'hvpls')" {
        description
          "Only applies when t-ldp pw type
           is h-vpls.";
      }
      description
        "Container for QinQ.";
      leaf s-tag {
        type dot1q-types:vlanid;
        mandatory true;
        description
          "S-TAG.";
      }
      leaf c-tag {
        type dot1q-types:vlanid;
        mandatory true;
        description
          "C-TAG.";
```

```
                        }
                      }
                    }
                    case l2tp {
                      description
                        "Container for L2TP PWs.";
                      leaf router-id {
                        type rt-types:router-id;
                        description
                          "A 32-bit number in the dotted-quad format
                           that is used to uniquely identify a node
                           within a service provider network.";
                        reference
                          "RFC 4667: Layer 2 Virtual Private Network
                                     (L2VPN) Extensions for Layer 2
                                     Tunneling Protocol (L2TP),
                                     Section 4.2";
                      }
                      leaf pseudowire-type {
                        type identityref {
                          base iana-pw-types:iana-pw-types;
                        }
                        description
                          "Encapsulation type.";
                        reference
                          "RFC 4667: Layer 2 Virtual Private Network
                                     (L2VPN) Extensions for Layer 2
                                     Tunneling Protocol (L2TP),
                                     Section 4.2";
                      }
                    }
                  }
                }
              }
            }
            container vpn-network-accesses {
              description
                "Main container for VPN network accesses.";
              list vpn-network-access {
                key "id";
                description
                  "List of VPN network accesses.";
                leaf id {
                  type vpn-common:vpn-id;
                  description
                    "Identifier of the network access";
                }
                leaf description {
```

```
        type string;
        description
          "A textual description of the VPN network
           access.";
      }
      leaf interface-id {
        type string;
        description
          "Refers to a physical or logical interface.";
      }
      leaf active-vpn-node-profile {
        type leafref {
            path "../../.."
             + "/active-global-parameters-profiles"
             + "/global-parameters-profile/profile-id";
        }
        description
          "An identifier of an active VPN instance
           profile.";
      }
      uses vpn-common:service-status;
      container connection {
        description
          "Container for the bearer and AC.";
        leaf l2-termination-point {
          type string;
          description
            "Specifies a reference to a local Layer 2
             termination point such as a Layer 2
             sub-interface.";
        }
        leaf local-bridge-reference {
          type string;
          description
            "Specifies a local bridge reference to
             accommodate, for example, implementations
             that require internal bridging.
             A reference may be a local bridge domain.";
        }
        leaf bearer-reference {
          if-feature "vpn-common:bearer-reference";
          type string;
          description
            "This is an internal reference for the service
             provider to identify the bearer associated
             with this VPN.";
        }
        container encapsulation {
```

```
                description
                  "Container for Layer 2 encapsulation.";
                leaf encap-type {
                  type identityref {
                    base vpn-common:encapsulation-type;
                  }
                  default "vpn-common:priority-tagged";
                  description
                    "Tagged interface type. By default, the
                     type of the tagged interface is
                     'priority-tagged'.";
                }
                container dot1q {
                  when "derived-from-or-self(../encap-type, "
                     + "'vpn-common:dot1q')" {
                    description
                      "Only applies when the type of the
                       tagged interface is 'dot1q'.";
                  }
                  description
                    "Tagged interface.";
                  leaf tag-type {
                    type identityref {
                      base vpn-common:tag-type;
                    }
                    default "vpn-common:c-vlan";
                    description
                      "Tag type. By default, the tag type is
                       'c-vlan'.";
                  }
                  leaf cvlan-id {
                    type dot1q-types:vlanid;
                    description
                      "VLAN identifier.";
                  }
                  container tag-operations {
                    description
                      "Sets the tag manipulation policy for this
                       VPN network access. It defines a set of
                       tag manipulations that allow for the
                       insertion, removal, or rewriting
                       of 802.1Q VLAN tags. These operations are
                       indicated for the CE-PE direction.
                       By default, tag operations are symmetric.
                       As such, the reverse tag operation is
                       assumed on the PE-CE direction.";
                    choice op-choice {
                      description
```

```
        "Selects the tag rewriting policy for a
         VPN network access.";
      leaf pop {
        type empty;
        description
          "Pop the outer tag.";
      }
      leaf push {
        type empty;
        description
          "Push one or two tags defined by the
           tag-1 and tag-2 leaves. It is
           assumed that, absent any policy, the
           default value of 0 will be used for
           PCP setting.";
      }
      leaf translate {
        type empty;
        description
          "Translate the outer tag to one or two
           tags. PCP bits are preserved.";
      }
    }
    leaf tag-1 {
      when 'not(../pop)';
      type dot1q-types:vlanid;
      description
        "A first tag to be used for push or
         translate operations. This tag will be
         used as the outermost tag as a result
         of the tag operation.";
    }
    leaf tag-1-type {
      type dot1q-types:dot1q-tag-type;
      default "dot1q-types:s-vlan";
      description
        "Specifies a specific 802.1Q tag type
         of tag-1.";
    }
    leaf tag-2 {
      when '(../translate)';
      type dot1q-types:vlanid;
      description
        "A second tag to be used for
         translation.";
    }
    leaf tag-2-type {
      type dot1q-types:dot1q-tag-type;
```

```
            default "dot1q-types:c-vlan";
            description
              "Specifies a specific 802.1Q tag type
               of tag-2.";
          }
        }
      }
      container priority-tagged {
        when "derived-from-or-self(../encap-type, "
           + "'vpn-common:priority-tagged')" {
          description
            "Only applies when the type of the
             tagged interface is 'priority-tagged'.";
        }
        description
          "Priority tagged container.";
        leaf tag-type {
          type identityref {
            base vpn-common:tag-type;
          }
          default "vpn-common:c-vlan";
          description
            "Tag type. By default, the tag type is
             'c-vlan'.";
        }
      }
      container qinq {
        when "derived-from-or-self(../encap-type, "
           + "'vpn-common:qinq')" {
          description
            "Only applies when the type of the tagged
             interface is QinQ.";
        }
        description
          "Includes QinQ parameters.";
        leaf tag-type {
          type identityref {
            base vpn-common:tag-type;
          }
          default "vpn-common:s-c-vlan";
          description
            "Tag type. By default, the tag type is
             's-c-vlan'.";
        }
        leaf svlan-id {
          type dot1q-types:vlanid;
          mandatory true;
          description
```

```
        "S-VLAN identifier.";
      }
      leaf cvlan-id {
        type dot1q-types:vlanid;
        mandatory true;
        description
          "C-VLAN identifier.";
      }
      container tag-operations {
        description
          "Sets the tag manipulation policy for this
           VPN network access. It defines a set of
           tag manipulations that allow for the
           insertion, removal, or rewriting
           of 802.1Q VLAN tags. These operations are
           indicated for the CE-PE direction.
           By default, tag operations are symmetric.
           As such, the reverse tag operation is
           assumed on the PE-CE direction.";
        choice op-choice {
          description
            "Selects the tag rewriting policy for a
             VPN network access.";
          leaf pop {
            type uint8 {
              range "1|2";
            }
            description
              "Pop one or two tags as a function
               of the indicated pop value.";
          }
          leaf push {
            type empty;
            description
              "Push one or two tags defined by the
               tag-1 and tag-2 leaves. It is
               assumed that, absent any policy, the
               default value of 0 will be used for
               PCP setting.";
          }
          leaf translate {
            type uint8 {
              range "1|2";
            }
            description
              "Translate one or two outer tags. PCP
               bits are preserved.
```

```
            The following operations are
            supported:

            - translate 1 with tag-1 leaf is
              provided: only the outermost tag is
              translated to the value in tag-1.

            - translate 2 with both tag-1 and
              tag-2 leaves are provided: both
              outer and inner tags are translated
              to the values in tag-1 and tag-2,
              respectively.

            - translate 2 with tag-1 leaf is
              provided: the outer tag is popped
              while the inner tag is translated
              to the value in tag-1.";
      }
    }
    leaf tag-1 {
      when 'not(../pop)';
      type dot1q-types:vlanid;
      description
        "A first tag to be used for push or
         translate operations. This tag will be
         used as the outermost tag as a result
         of the tag operation.";
    }
    leaf tag-1-type {
      type dot1q-types:dot1q-tag-type;
      default "dot1q-types:s-vlan";
      description
        "Specifies a specific 802.1Q tag type
         of tag-1.";
    }
    leaf tag-2 {
      when 'not(../pop)';
      type dot1q-types:vlanid;
      description
        "A second tag to be used for push or
         translate operations.";
    }
    leaf tag-2-type {
      type dot1q-types:dot1q-tag-type;
      default "dot1q-types:c-vlan";
      description
        "Specifies a specific 802.1Q tag type
         of tag-2.";
```

```
          }
        }
      }
    }
    container lag-interface {
      if-feature "vpn-common:lag-interface";
      description
        "Container of LAG interface attributes
         configuration.";
      leaf lag-interface-id {
        type string;
        description
          "LAG interface identifier.";
      }
      container lacp {
        description
          "Container for LACP.";
        leaf lacp-state {
          type boolean;
          default "false";
          description
            "Controls whether LACP is enabled.";
        }
        leaf mode {
          type identityref {
            base lacp-mode;
          }
          description
            "Indicates the LACP mode.";
        }
        leaf speed {
          type uint32;
          units "mbps";
          default "10";
          description
            "LACP speed. This low default value
             is inherited from the L2SM.";
        }
        leaf mini-link-num {
          type uint32;
          description
            "Defines the minimum number of links that
             must be active before the aggregating
             link is put into service.";
        }
        leaf system-id {
          type yang:mac-address;
          description
```

```
        "Indicates the System ID used by LACP.";
    }
    leaf admin-key {
      type uint16;
      description
        "Indicates the value of the key used for
         the aggregate interface.";
    }
    leaf system-priority {
      type uint16 {
        range "0..65535";
      }
      default "32768";
      description
        "Indicates the LACP priority for the
         system.";
    }
    container member-link-list {
      description
        "Container of Member link list.";
      list member-link {
        key "name";
        description
          "Member link.";
        leaf name {
          type string;
          description
            "Member link name.";
        }
        leaf speed {
          type uint32;
          units "mbps";
          default "10";
          description
            "Port speed.";
        }
        leaf mode {
          type identityref {
            base vpn-common:neg-mode;
          }
          description
            "Negotiation mode.";
        }
        leaf link-mtu {
          type uint32;
          units "bytes";
          description
            "Link MTU size.";
```

```
                        }
                        container oam-802.3ah-link {
                          if-feature "oam-3ah";
                          description
                            "Container for oam 802.3ah link.";
                          leaf enable {
                            type boolean;
                            default "false";
                            description
                              "Indicates support of OAM 802.3ah
                               link.";
                          }
                        }
                      }
                    }
                    leaf flow-control {
                      type boolean;
                      default "false";
                      description
                        "Indicates whether flow control is
                         supported.";
                    }
                    leaf lldp {
                      type boolean;
                      default "false";
                      description
                        "Indicates whether Link Layer Discovery
                         Protocol (LLDP) is supported.";
                    }
                  }
                  container split-horizon {
                    description
                      "Configuration with split horizon enabled.";
                    leaf group-name {
                      type string;
                      description
                        "Group name of the Split Horizon.";
                    }
                  }
                }
              }
              choice signaling-option {
                description
                  "Choice for the signaling-option.";
                case bgp {
                  description
                    "BGP is used as the signaling protocol.";
                  choice bgp-type {
```

```
                        description
                          "Choice for the BGP type.";
                        case l2vpn-bgp {
                          description
                            "Container for BGP L2VPN.";
                          leaf ce-id {
                            type uint16;
                            description
                              "Identifies the CE within the VPN.";
                            reference
                              "RFC 6624: Layer 2 Virtual Private
                                         Networks Using BGP for
                                         Auto-Discovery and
                                         Signaling";
                          }
                          leaf remote-ce-id {
                            type uint16;
                            description
                              "Indicates the identifier of the remote
                               CE.";
                          }
                          container vpls-instance {
                            when "derived-from-or-self(../../../../../"
                               + "vpn-type, 'vpn-common:vpls')" {
                              description
                                "Only applies for VPLS.";
                            }
                            description
                              "VPLS instance.";
                            leaf vpls-edge-id {
                              type uint16;
                              description
                                "VPLS Edge Identifier (VE ID).";
                              reference
                                "RFC 4761: Virtual Private LAN Service
                                           (VPLS) Using BGP for Auto-
                                           Discovery and Signaling,
                                           Section 3.2.1";
                            }
                          }
                        }
                        case evpn-bgp {
                          description
                            "Used for EVPN.";
                          leaf df-preference {
                            type uint16;
                            default "32767";
                            description
```

```
          "Defines a 2-octet value that indicates
           the PE preference to become the DF in
           the ES.

           The preference value is only applicable
           to the preference based method.";
        reference
          "RFC 8584: Framework for Ethernet VPN
                     Designated Forwarder Election
                     Extensibility";
      }
      container vpws-service-instance {
        when "derived-from-or-self(../../../../../"
           + "vpn-type, 'vpn-common:vpws-evpn')" {
          description
            "Only applies for EVPN-VPWS.";
        }
        description
          "Local and remote VPWS Service Instance
           (VSI)";
        reference
          "RFC 8214: Virtual Private Wire Service
                     Support in Ethernet VPN";
        choice local-vsi-choice {
          description
            "Choices for assigning local VSI.";
          case directly-assigned {
            description
              "Explicitly assign a local VSI.";
            leaf local-vpws-service-instance {
              type uint32 {
                range "1..16777215";
              }
              description
                "Indicates the assigned local
                 VSI.";
            }
          }
          case auto-assigned {
            description
              "The local VSI is auto-assigned.";
            container local-vsi-auto {
              description
                "The local VSI is auto-assigned.";
              choice auto-mode {
                description
                  "Indicates the auto-assignment
                   mode of local VSI. VSI can be
```

```
                automatically assigned either
                with or without indicating a
                pool from which the VSI
                should be taken.

                For both cases, the server
                will auto-assign a local VSI
                value and use that value.";
              case from-pool {
                leaf vsi-pool-name {
                  type string;
                  description
                    "The auto-assignment will be
                     made from this pool.";
                }
              }
              case full-auto {
                leaf auto {
                  type empty;
                  description
                    "Indicates that a local VSI
                     is fully auto-assigned.";
                }
              }
            }
            leaf auto-local-vsi {
              type uint32 {
                range "1..16777215";
              }
              config false;
              description
                "The value of the auto-assigned
                 local VSI.";
            }
          }
        }
      }
      choice remote-vsi-choice {
        description
          "Choice for assigning the remote VSI.";
        case directly-assigned {
          description
            "Explicitly assign a remote VSI.";
          leaf remote-vpws-service-instance {
            type uint32 {
              range "1..16777215";
            }
            description
```

```
          "Indicates the value of the remote
           VSI.";
      }
    }
    case auto-assigned {
      description
        "The remote VSI is auto-assigned.";
      container remote-vsi-auto {
        description
          "The remote VSI is auto-assigned.";
        choice auto-mode {
          description
            "Indicates the auto-assignment
             mode of remote VSI. VSI can be
             automatically assigned either
             with or without indicating a
             pool from which the VSI
             should be taken.

             For both cases, the server
             will auto-assign a remote VSI
             value and use that value.";
          case from-pool {
            leaf vsi-pool-name {
              type string;
              description
                "The auto-assignment will be
                 made from this pool.";
            }
          }
          case full-auto {
            leaf auto {
              type empty;
              description
                "Indicates that a remote VSI
                 is fully auto-assigned.";
            }
          }
        }
        leaf auto-remote-vsi {
          type uint32 {
            range "1..16777215";
          }
          config false;
          description
            "The value of the auto-assigned
             remote VSI.";
        }
```

```
                    }
                  }
                }
              }
            }
          }
        }
        list group {
          key "group-id";
          description
            "List of group-ids.";
          leaf group-id {
            type string;
            description
              "Indicates the group-id to which the network
               access belongs to.";
          }
          leaf precedence {
            type identityref {
              base precedence-type;
            }
            description
              "Defining service redundancy in transport
               network.";
          }
          leaf ethernet-segment-identifier {
            type l2vpn-es:es-ref;
            description
              "Reference to the ESI associated with the VPN
               network access.";
          }
        }
        container ethernet-service-oam {
          description
            "Container for Ethernet service OAM.";
          leaf md-name {
            type string;
            description
              "Maintenance domain name.";
          }
          leaf md-level {
            type uint8;
            description
              "Maintenance domain level.";
          }
          container cfm-802.1-ag {
            description
```

```
            "Container of 802.1ag CFM configurations.";
          list n2-uni-c {
            key "maid";
            description
              "List of UNI-N to UNI-C.";
            uses cfm-802;
          }
          list n2-uni-n {
            key "maid";
            description
              "List of UNI-N to UNI-N.";
            uses cfm-802;
          }
        }
        uses y-1731;
      }
      container service {
        description
          "Container for service";
        leaf mtu {
          type uint32;
          units "bytes";
          description
            "Layer 2 MTU, it is also known as the maximum
             transmission unit or maximum frame size.";
        }
        container svc-pe-to-ce-bandwidth {
          if-feature "vpn-common:inbound-bw";
          description
            "From the customer site's perspective, the
             service inbound bandwidth of the connection
             or download bandwidth from the service
             provider the site. Note that the L2SM uses
             'input-bandwidth' to refer to the same
             concept.";
          list pe-to-ce-bandwidth {
            key "bw-type";
            description
              "List for PE-to-CE bandwidth data nodes.";
            leaf bw-type {
              type identityref {
                base vpn-common:bw-type;
              }
              description
                "Indicates the bandwidth type.";
            }
            choice type {
              description
```

```
                "Choice based upon bandwidth type.";
              case per-cos {
                description
                  "Bandwidth per CoS.";
                list cos {
                  key "cos-id";
                  description
                    "List of class of services.";
                  leaf cos-id {
                    type uint8;
                    description
                      "Identifier of the CoS, indicated by
                       DSCP or a CE-CLAN CoS (802.1p) value
                       in the service frame.";
                    reference
                      "IEEE Std 802.1Q: Bridges and Bridged
                                        Networks";
                  }
                  uses bandwidth-parameters;
                }
              }
              case other {
                description
                  "Other bandwidth types.";
                uses bandwidth-parameters;
              }
            }
          }
        }
        container svc-ce-to-pe-bandwidth {
          if-feature "vpn-common:outbound-bw";
          description
            "From the customer site's perspective,
             the service outbound bandwidth of the
             connection or upload bandwidth from
             the CE to the PE. Note that the L2SM uses
             'output-bandwidth' to refer to the same
             concept.";
          list ce-to-pe-bandwidth {
            key "bw-type";
            description
              "List for CE-to-PE bandwidth.";
            leaf bw-type {
              type identityref {
                base vpn-common:bw-type;
              }
              description
                "Indicates the bandwidth type.";
```

```
          }
          choice type {
            description
              "Choice based upon bandwidth type.";
            case per-cos {
              description
                "Bandwidth per CoS.";
              list cos {
                key "cos-id";
                description
                  "List of class of services.";
                leaf cos-id {
                  type uint8;
                  description
                    "Identifier of the CoS, indicated by
                     DSCP or a CE-CLAN CoS (802.1p) value
                     in the service frame.";
                  reference
                    "IEEE Std 802.1Q: Bridges and Bridged
                                      Networks";
                }
                uses bandwidth-parameters;
              }
            }
            case other {
              description
                "Other non CoS-aware bandwidth types.";
              uses bandwidth-parameters;
            }
          }
        }
      }
      container qos {
        if-feature "vpn-common:qos";
        description
          "QoS configuration.";
        container qos-classification-policy {
          description
            "Configuration of the traffic classification
             policy.";
          list rule {
            key "id";
            ordered-by user;
            description
              "List of classification rules.";
            leaf id {
              type string;
              description
```

```
        "A description identifying the QoS
         classification policy rule.";
      }
      choice match-type {
        default "match-flow";
        description
          "Choice for classification.";
        case match-flow {
          container match-flow {
            description
              "Describes flow-matching criteria.";
            leaf dscp {
              type inet:dscp;
              description
                "DSCP value.";
            }
            leaf dot1q {
              type uint16;
              description
                "802.1Q matching. It is a VLAN tag
                 added into a frame.";
              reference
                "IEEE Std 802.1Q: Bridges and
                                  Bridged
                                  Networks";
            }
            leaf pcp {
              type uint8 {
                range "0..7";
              }
              description
                "Priority Code Point (PCP) value.";
            }
            leaf src-mac-address {
              type yang:mac-address;
              description
                "Source MAC address.";
            }
            leaf dst-mac-address {
              type yang:mac-address;
              description
                "Destination MAC address.";
            }
            leaf color-type {
              type identityref {
                base color-type;
              }
              description
```

```
                    "Color type.";
                }
                leaf any {
                  type empty;
                  description
                    "Allows all.";
                }
              }
            }
            case match-application {
              leaf match-application {
                type identityref {
                  base vpn-common:customer-application;
                }
                description
                  "Defines the application to match.";
              }
            }
          }
          leaf target-class-id {
            type string;
            description
              "Identification of the CoS.
               This identifier is internal to the
               administration.";
          }
        }
      }
      container qos-profile {
        description
          "QoS profile configuration.";
        list qos-profile {
          key "profile";
          description
            "QoS profile.
             Can be standard profile or customized
             profile.";
          leaf profile {
            type leafref {
              path "/l2vpn-ntw/vpn-profiles"
                 + "/valid-provider-identifiers"
                 + "/qos-profile-identifier/id";
            }
            description
              "QoS profile to be used.";
          }
          leaf direction {
            type identityref {
```

```
                    base vpn-common:qos-profile-direction;
                  }
                  default "vpn-common:both";
                  description
                    "The direction to which the QoS profile
                     is applied.";
                }
              }
            }
          }
          container mac-policies {
            description
              "Container for MAC-related policies.";
            list access-control-list {
              key "name";
              description
                "Container for access control List.";
              leaf name {
                type string;
                description
                  "Specifies the name of the ACL.";
              }
              leaf-list src-mac-address {
                type yang:mac-address;
                description
                  "Specifies the source MAC address.";
              }
              leaf-list src-mac-address-mask {
                type yang:mac-address;
                description
                  "Specifies the source MAC address mask.";
              }
              leaf-list dst-mac-address {
                type yang:mac-address;
                description
                  "Specifies the destination MAC address.";
              }
              leaf-list dst-mac-address-mask {
                type yang:mac-address;
                description
                  "Specifies the destination MAC address
                   mask.";
              }
              leaf action {
                type identityref {
                  base mac-action;
                }
                default "drop";
```

```
        description
          "Specifies the filtering action.";
        }
        leaf rate-limit {
          when "derived-from-or-self(../action, "
             + "'flood')" {
            description
              "Rate-limit is valid only when the action
               is to accept the matching frame.";
          }
          type decimal64 {
            fraction-digits 2;
          }
          units "bytes per second";
          description
            "Specifies how to rate-limit the traffic.";
        }
      }
      container mac-loop-prevention {
        description
          "Container of MAC loop prevention.";
        leaf window {
          type uint32;
          units "seconds";
          default "180";
          description
            "The timer when a MAC mobility event is
             detected.";
        }
        leaf frequency {
          type uint32;
          default "5";
          description
            "The number of times to detect MAC
             duplication, where a 'duplicate MAC
             address' situation has occurred and
             the duplicate MAC address has been
             added to a list of duplicate MAC
             addresses.";
        }
        leaf retry-timer {
          type uint32;
          units "seconds";
          description
            "The retry timer. When the retry timer
             expires, the duplicate MAC address will
             be flushed from the MAC-VRF.";
        }
```

```
        leaf protection-type {
          type identityref {
            base loop-prevention-type;
          }
          default "trap";
          description
            "Protection type";
        }
      }
      container mac-addr-limit {
        description
          "Container of MAC-Addr limit configurations";
        leaf limit-number {
          type uint16;
          default "2";
          description
            "Maximum number of MAC addresses learned
             from the subscriber for a single service
             instance.";
        }
        leaf time-interval {
          type uint32;
          units "milliseconds";
          default "300";
          description
            "The aging time of the mac address.";
        }
        leaf action {
          type identityref {
            base mac-action;
          }
          default "warning";
          description
            "Specifies the action when the upper limit
             is exceeded: drop the packet, flood the
             packet, or log a warning message (without
             dropping the packet).";
        }
      }
    }
    container broadcast-unknown-unicast-multicast {
      description
        "Container of broadcast, unknown unicast, and
         multicast configurations";
      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 the multicast site.";
        }
        list multicast-gp-address-mapping {
          key "id";
          description
            "List of Port to group mappings.";
          leaf id {
            type uint16;
            description
              "Unique identifier for the mapping.";
          }
          leaf vlan-id {
            type uint32;
            mandatory true;
            description
              "The VLAN ID of the multicast group.";
          }
          leaf mac-gp-address {
            type yang:mac-address;
            mandatory true;
            description
              "The MAC address of the multicast group.";
          }
          leaf port-lag-number {
            type uint32;
            description
              "The port/LAG belonging to the multicast
               group.";
          }
        }
        leaf bum-overall-rate {
          type uint64;
          units "bps";
          description
```

```
                            "Overall rate for BUM.";
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

9. Security Considerations

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

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

There are a number of data nodes defined in "ietf-l2vpn-ntw" and "ietf-ethernet-segment" YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability in the "ietf-l2vpn-ntw" and "ietf-ethernet-segment" modules:

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

* 'ethernet-segments' and 'vpn-services': An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L2VPN service, interrupting all the traffic of a client. In addition, an attacker may modify the attributes of a running service (e.g., QoS, bandwidth) or an ES, leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create an L2VPN service, add a new network access, or intercept/redirect the traffic to a non-authorized node. In addition to using NACM to prevent authorized access, such activity can be detected by adequately monitoring and tracking network configuration changes.

Some of the readable data nodes in the "ietf-l2vpn-ntw" YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

Both "iana-bgp-l2-encaps" and "iana-pseudowire-types" modules define YANG identities for encapsulation/pseudowires types. These identities are intended to be referenced by other YANG modules, and by themselves do not expose any nodes which are writable, contain read-only state, or RPCs.

10. IANA Considerations

10.1. Registering YANG Modules

This document requests IANA to register the following URIs in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

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

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

URI: urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

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

name: iana-bgp-l2-encaps
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps
maintained by IANA: Y
prefix: iana-bgp-l2-encaps
reference: RFC XXXX

name: iana-pseudowire-types
namespace: urn:ietf:params:xml:ns:yang:iana-pseudowire-types
maintained by IANA: Y
prefix: iana-pw-types
reference: RFC XXXX

name: ietf-ethernet-segment
namespace: urn:ietf:params:xml:ns:yang:ietf-ethernet-segment
maintained by IANA: N
prefix: l2vpn-es
reference: RFC XXXX

name: ietf-l2vpn-ntw
namespace: urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw
maintained by IANA: N
prefix: l2vpn-ntw
reference: RFC XXXX

10.2. BGP Layer 2 Encapsulation Types

This document defines the initial version of the IANA-maintained "iana-bgp-l2-encaps" YANG module (Section 8.1). IANA is requested to add this note to the registry:

> BGP Layer 2 encapsulation types must not be directly added to the "iana-bgp-l2-encaps" YANG module. They must instead be added to the "BGP Layer 2 Encapsulation Types" registry [IANA-BGP-L2].

When a Layer 2 encapsulation type is added to the "BGP Layer 2 Encapsulation Types" registry, a new "identity" statement must be added to the "iana-bgp-l2-encaps" YANG module. The name of the "identity" is a lower-case version of the encapsulation name provided in the description. The "identity" statement should have the following sub-statements defined:

"base": Contains 'bgp-l2-encaps-type'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry with the title of the document added.

Unassigned or reserved values are not present in the module.

When the "iana-bgp-l2-encaps" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to [IANA-BGP-L2]:

> When this registry is modified, the YANG module "iana-bgp-l2-encaps" must be updated as defined in RFCXXXX.

10.3. Pseudowire Types

This document defines the initial version of the IANA-maintained "iana-pseudowire-types" YANG module (Section 8.2). IANA is requested to add this note to the registry:

> MPLS pseudowire types must not be directly added to the "iana-bgp-l2-encaps" YANG module. They must instead be added to the "MPLS Pseudowire Types" registry [IANA-PW-Types].

When a pseudowire type is added to the "iana-pseudowire-types" registry, a new "identity" statement must be added to the "iana-pseudowire-types" YANG module. The name of the "identity" is a

lower-case version of the encapsulation name provided in the description. The "identity" statement should have the following sub-statements defined:

"base": Contains 'iana-pw-types'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry with the title of the document added

Unassigned or reserved values are not present in the module.

When the "iana-pseudowire-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to [IANA-PW-Types]:

> When this registry is modified, the YANG module "iana-pseudowire-types" must be updated as defined in RFCXXXX.

11. References

11.1. Normative References

[IANA-BGP-L2]
IANA, "BGP Layer 2 Encapsulation Types", <https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-l2-encapsulation-types-registry>.

[IANA-PW-Types]
IANA, "MPLS Pseudowire Types Registry", <http://www.iana.org/assignments/pwe3-parameters/pwe3-parameters.xhtml#pwe3-parameters-2>.

[IEEE-802-1ag]
IEEE, "802.1ag - 2007 - IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management", 2007, <DOI 10.1109/IEEESTD.2007.4431836>.

[IEEE802.1Qcp-2018]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model", September 2018, <https://ieeexplore.ieee.org/document/8467507>.

[ITU-T-Y-1731]
Union, I. T., "Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet-based networks", August 2015, <https://www.itu.int/rec/T-REC-Y.1731/en>.

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

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

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <https://www.rfc-editor.org/info/rfc4446>.

[RFC4667] Luo, W., "Layer 2 Virtual Private Network (L2VPN) Extensions for Layer 2 Tunneling Protocol (L2TP)", RFC 4667, DOI 10.17487/RFC4667, September 2006, <https://www.rfc-editor.org/info/rfc4667>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <https://www.rfc-editor.org/info/rfc4762>.

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

[RFC6074] Rosen, E., Davie, B., Radoaca, V., and W. Luo, "Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)", RFC 6074, DOI 10.17487/RFC6074, January 2011, <https://www.rfc-editor.org/info/rfc6074>.

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

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

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <https://www.rfc-editor.org/info/rfc6624>.

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

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

[RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <https://www.rfc-editor.org/info/rfc7623>.

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

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

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

[RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <https://www.rfc-editor.org/info/rfc8214>.

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

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

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

[RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <https://www.rfc-editor.org/info/rfc8365>.

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

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

[RFC8584] Rabadan, J., Ed., Mohanty, S., Ed., Sajassi, A., Drake, J., Nagaraj, K., and S. Sathappan, "Framework for Ethernet VPN Designated Forwarder Election Extensibility", RFC 8584, DOI 10.17487/RFC8584, April 2019, <https://www.rfc-editor.org/info/rfc8584>.

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

11.2. Informative References

[I-D.ietf-bess-evpn-pref-df]
Rabadan, J., Sathappan, S., Przygienda, T., Lin, W., Drake, J., Sajassi, A., and S. Mohanty, "Preference-based EVPN DF Election", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-pref-df-08, 23 September 2021, <https://www.ietf.org/archive/id/draft-ietf-bess-evpn-pref-df-08.txt>.

[I-D.ietf-bess-evpn-yang]
Brissette, P., Shah, H., Hussain, I., Tiruveedhula, K., and J. Rabadan, "Yang Data Model for EVPN", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-yang-07, 11 March 2019, <https://www.ietf.org/archive/id/draft-ietf-bess-evpn-yang-07.txt>.

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-13, 6 March 2022, <https://www.ietf.org/archive/id/draft-ietf-idr-bgp-model-13.txt>.

[I-D.ietf-opsawg-sap]
Boucadair, M., Dios, O. G. D., Barguil, S., Wu, Q., and V. Lopez, "A Network YANG Model for Service Attachment Points (SAPs)", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-07, 20 May 2022, <https://www.ietf.org/archive/id/draft-ietf-opsawg-sap-07.txt>.

[I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-10, 6 March 2022, <https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-10.txt>.

[I-D.ietf-teas-ietf-network-slices]
Farrel, A., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slices-10, 27 March 2022, <https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slices-10.txt>.

[I-D.ietf-teas-te-service-mapping-yang]
Lee, Y., Dhody, D., Fioccola, G., Wu, Q., Ceccarelli, D., and J. Tantsura, "Traffic Engineering (TE) and Service Mapping YANG Model", Work in Progress, Internet-Draft, draft-ietf-teas-te-service-mapping-yang-10, 7 March 2022, <https://www.ietf.org/archive/id/draft-ietf-teas-te-service-mapping-yang-10.txt>.

[IEEE-802-1ah]
IEEE, "IEEE Standard for Local and metropolitan area networks -- Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", IEEE Std 801.3AH-2008, 2008, <https://standards.ieee.org/standard/802_1ah-2008.html>.

[IEEE-802-3ah]
IEEE, "802.3ah - 2004 - IEEE Standard for Information technology-- Local and metropolitan area networks-- Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment: Media Access Control Parameters, Physical Layers, and Management Parameters for Subscriber Access Networks", IEEE Std 802.3AH-2004, 2004, <DOI 10.1109/IEEESTD.2004.94617>.

[IEEE802.1AX]
"Link Aggregation", IEEE Std 802.1AX-2020, 2020.

[IEEE802.1Q]
"Bridges and Bridged Networks", IEEE Std 802.1Q-2018, 6 July 2018, <https://ieeexplore.ieee.org/document/8403927>.

[MFA] "The Use of Virtual Trunks for ATM/MPLS Control Plane Interworking Specification", MFA Forum 9.0.0 , February 2006.

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

[RFC2507] Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression", RFC 2507, DOI 10.17487/RFC2507, February 1999, <https://www.rfc-editor.org/info/rfc2507>.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, DOI 10.17487/RFC2508, February 1999, <https://www.rfc-editor.org/info/rfc2508>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC3545] Koren, T., Casner, S., Geevarghese, J., Thompson, B., and P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering", RFC 3545, DOI 10.17487/RFC3545, July 2003, <https://www.rfc-editor.org/info/rfc3545>.

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

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <https://www.rfc-editor.org/info/rfc4448>.

[RFC4553] Vainshtein, A., Ed. and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, DOI 10.17487/RFC4553, June 2006, <https://www.rfc-editor.org/info/rfc4553>.

[RFC4618] Martini, L., Rosen, E., Heron, G., and A. Malis, "Encapsulation Methods for Transport of PPP/High-Level Data Link Control (HDLC) over MPLS Networks", RFC 4618, DOI 10.17487/RFC4618, September 2006, <https://www.rfc-editor.org/info/rfc4618>.

[RFC4619] Martini, L., Ed., Kawa, C., Ed., and A. Malis, Ed., "Encapsulation Methods for Transport of Frame Relay over Multiprotocol Label Switching (MPLS) Networks", RFC 4619, DOI 10.17487/RFC4619, September 2006, <https://www.rfc-editor.org/info/rfc4619>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[RFC4717] Martini, L., Jayakumar, J., Bocci, M., El-Aawar, N., Brayley, J., and G. Koleyni, "Encapsulation Methods for Transport of Asynchronous Transfer Mode (ATM) over MPLS Networks", RFC 4717, DOI 10.17487/RFC4717, December 2006, <https://www.rfc-editor.org/info/rfc4717>.

[RFC4816] Malis, A., Martini, L., Brayley, J., and T. Walsh, "Pseudowire Emulation Edge-to-Edge (PWE3) Asynchronous Transfer Mode (ATM) Transparent Cell Transport Service", RFC 4816, DOI 10.17487/RFC4816, February 2007, <https://www.rfc-editor.org/info/rfc4816>.

[RFC4842] Malis, A., Pate, P., Cohen, R., Ed., and D. Zelig, "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation over Packet (CEP)", RFC 4842, DOI 10.17487/RFC4842, April 2007, <https://www.rfc-editor.org/info/rfc4842>.

[RFC4863] Martini, L. and G. Swallow, "Wildcard Pseudowire Type", RFC 4863, DOI 10.17487/RFC4863, May 2007, <https://www.rfc-editor.org/info/rfc4863>.

[RFC4901] Ash, J., Ed., Hand, J., Ed., and A. Malis, Ed., "Protocol Extensions for Header Compression over MPLS", RFC 4901, DOI 10.17487/RFC4901, June 2007, <https://www.rfc-editor.org/info/rfc4901>.

[RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, DOI 10.17487/RFC5086, December 2007, <https://www.rfc-editor.org/info/rfc5086>.

[RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, DOI 10.17487/RFC5087, December 2007, <https://www.rfc-editor.org/info/rfc5087>.

[RFC5143] Malis, A., Brayley, J., Shirron, J., Martini, L., and S. Vogelsang, "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation Service over MPLS (CEM) Encapsulation", RFC 5143, DOI 10.17487/RFC5143, February 2008, <https://www.rfc-editor.org/info/rfc5143>.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <https://www.rfc-editor.org/info/rfc5795>.

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

[RFC6307] Black, D., Ed., Dunbar, L., Ed., Roth, M., and R. Solomon, "Encapsulation Methods for Transport of Fibre Channel Traffic over MPLS Networks", RFC 6307, DOI 10.17487/RFC6307, April 2012, <https://www.rfc-editor.org/info/rfc6307>.

[RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <https://www.rfc-editor.org/info/rfc7209>.

[RFC7267] Martini, L., Ed., Bocci, M., Ed., and F. Balus, Ed., "Dynamic Placement of Multi-Segment Pseudowires", RFC 7267, DOI 10.17487/RFC7267, June 2014, <https://www.rfc-editor.org/info/rfc7267>.

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

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[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>.

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

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

[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>.

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

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

[RFC8960] Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A YANG Data Model for MPLS Base", RFC 8960, DOI 10.17487/RFC8960, December 2020, <https://www.rfc-editor.org/info/rfc8960>.

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

Appendix A. Examples

This section includes a non-exhaustive list of examples to illustrate the use of the L2NM.

In the following subsections, only the content of the message bodies is shown using JSON notations [RFC7951].

The examples use the folding defined in [RFC8792] for long lines.

A.1. BGP-based VPLS

This section provides an example to illustrate how the L2NM can be used to manage BGP-based VPLS. We consider the sample VPLS service delivered using the architecture depicted in Figure 23. In accordance with [RFC4761], we assume that a full mesh is established between all PEs. The details about such full mesh are not detailed here.

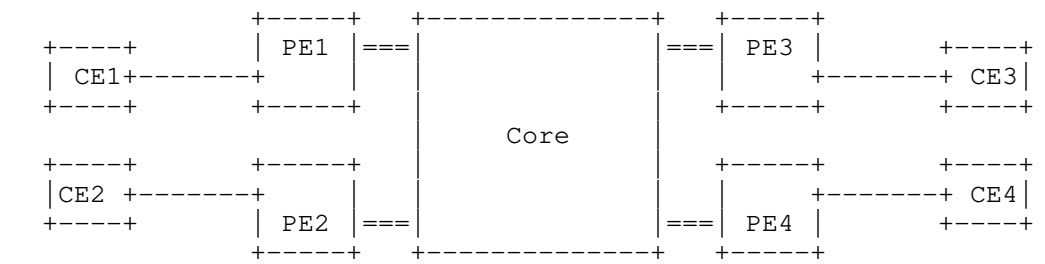


Figure 23: An Example of VPLS

Figure 24 show an example of a message body used to configure a VPLS instance using the L2NM. In this example, BGP is used for both auto-discovery and signaling. The 'signaling-type' data node is set to 'vpn-common:bgp-signaling'.


```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpls7714825356",
          "vpn-description": "Sample BGP-based VPLS",
          "customer-name": "customer-7714825356",
          "vpn-type": "ietf-vpn-common:vpls",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:bgp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65535,
                "svc-mtu": 1518,
                "rd-suffix": 1,
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
```

```
              }
            ]
          },
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "pe1",
                "ne-id": "198.51.100.1",
                "active-global-parameters-profiles": {
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "bgp-auto-discovery": {
                  "vpn-id": "1"
                },
                "signaling-option": {
                  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
                  "vpls-instance": {
                    "vpls-edge-id": 1,
                    "vpls-edge-id-range": 100
                  }
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE1",
                      "active-vpn-node-profile": "simple-profile",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
                      "connection": {
                        "encapsulation": {
                          "encap-type": "ietf-vpn-common:dot1q",
                          "dot1q": {
                            "cvlan-id": 1
                          }
                        }
                      }
                    }
                  ]
```

```
                }
              },
              {
                "vpn-node-id": "pe2",
                "ne-id": "198.51.100.2",
                "active-global-parameters-profiles": {
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "bgp-auto-discovery": {
                  "vpn-id": "1"
                },
                "signaling-option": {
                  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
                  "vpls-instance": {
                    "vpls-edge-id": 2,
                    "vpls-edge-id-range": 100
                  }
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE2",
                      "active-vpn-node-profile": "simple-profile",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
                      "connection": {
                        "encapsulation": {
                          "encap-type": "ietf-vpn-common:dot1q",
                          "dot1q": {
                            "cvlan-id": 1
                          }
                        }
                      }
                    }
                  ]
                }
              },
              {
```

```
                "vpn-node-id": "pe3",
                "ne-id": "198.51.100.3",
                "active-global-parameters-profiles": {
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "bgp-auto-discovery": {
                  "vpn-id": "1"
                },
                "signaling-option": {
                  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
                  "vpls-instance": {
                    "vpls-edge-id": 3,
                    "vpls-edge-id-range": 100
                  }
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE3",
                      "active-vpn-node-profile": "simple-profile",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
                      "connection": {
                        "encapsulation": {
                          "encap-type": "ietf-vpn-common:dot1q",
                          "dot1q": {
                            "cvlan-id": 1
                          }
                        }
                      }
                    }
                  ]
                }
              },
              {
                "vpn-node-id": "pe4",
                "ne-id": "198.51.100.4",
                "active-global-parameters-profiles": {
```

```
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "bgp-auto-discovery": {
                  "vpn-id": "1"
                },
                "signaling-option": {
                  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
                  "vpls-instance": {
                    "vpls-edge-id": 4,
                    "vpls-edge-id-range": 100
                  }
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE4",
                      "active-vpn-node-profile": "simple-profile",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
                      "connection": {
                        "encapsulation": {
                          "encap-type": "ietf-vpn-common:dot1q",
                          "dot1q": {
                            "cvlan-id": 1
                          }
                        }
                      }
                    }
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

Figure 24: Example of L2NM Message Body to Configure a BGP-based VPLS

A.2. BGP-based VPWS with LDP Signaling

Let's consider the simple architecture depicted in Figure 25 to offer a VPWS between CE1 and CE2. The service uses BGP for auto-discovery and LDP for signaling.

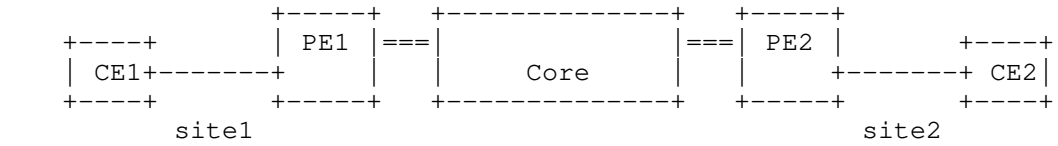


Figure 25: An Example of VPLS

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpws12345",
          "vpn-description": "Sample VPWS",
          "customer-name": "customer-12345",
          "vpn-type": "ietf-vpn-common:vpws",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65550,
                "rd-auto": {
                  "auto": [
                    null
                  ]
                },
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
```

```
                    }
                  ]
                }
              ]
            },
            "vpn-nodes": {
              "vpn-node": [
                {
                  "vpn-node-id": "pe1",
                  "ne-id": "2001:db8:100::1",
                  "active-global-parameters-profiles": {
                    "global-parameters-profile": [
                      {
                        "profile-id": "simple-profile"
                      }
                    ]
                  },
                  "bgp-auto-discovery": {
                    "vpn-id": "587"
                  },
                  "signaling-option": {
                    "advertise-mtu": true,
                    "ldp-or-l2tp": {
                      "saii": 1,
                      "remote-targets": [
                        {
                          "taii": 2
                        }
                      ],
                      "t-ldp-pw-type": "ethernet"
                    }
                  },
                  "vpn-network-accesses": {
                    "vpn-network-access": [
                      {
                        "id": "1/1/1.1",
                        "interface-id": "1/1/1",
                        "description": "Interface to CE1",
                        "active-vpn-node-profile": "simple-profile",
                        "status": {
                          "admin-status": {
                            "status": "ietf-vpn-common:admin-up"
                          }
                        }
                      }
                    ]
                  }
                },
```

```
                {
                  "vpn-node-id": "pe2",
                  "ne-id": "2001:db8:200::1",
                  "active-global-parameters-profiles": {
                    "global-parameters-profile": [
                      {
                        "profile-id": "simple-profile"
                      }
                    ]
                  },
                  "bgp-auto-discovery": {
                    "vpn-id": "587"
                  },
                  "signaling-option": {
                    "advertise-mtu": true,
                    "ldp-or-l2tp": {
                      "saii": 2,
                      "remote-targets": [
                        {
                          "taii": 1
                        }
                      ],
                      "t-ldp-pw-type": "ethernet"
                    }
                  },
                  "vpn-network-accesses": {
                    "vpn-network-access": [
                      {
                        "id": "5/1/1.1",
                        "interface-id": "5/1/1",
                        "description": "Interface to CE2",
                        "active-vpn-node-profile": "simple-profile",
                        "status": {
                          "admin-status": {
                            "status": "ietf-vpn-common:admin-up"
                          }
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }
        ]
      }
    }
  }
```

Figure 26: Example of L2NM Message Body to Configure a BGP-based VPWS with LDP Signaling

A.3. LDP-based VPLS

This section provides an example to illustrate how the L2NM can be used to manage a VPLS with LDP signaling. The connectivity between the CE and the PE is direct using Dot1q encapsulation [IEEE802.1Q]. We consider the sample service delivered using the architecture depicted in Figure 27.

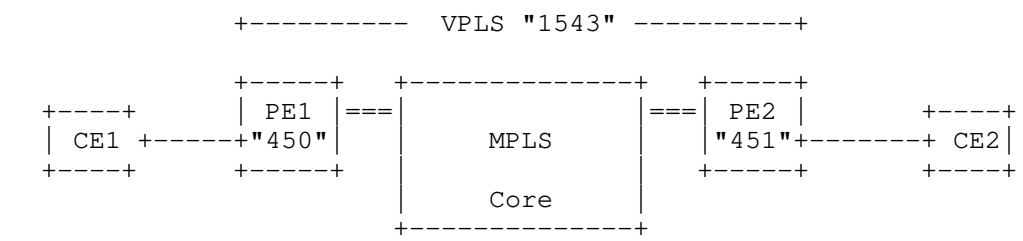


Figure 27: An Example of VPLS topology

Figure 28 shows how the L2NM is used to instruct both PE1 and PE2 to use the targeted LDP session between them to establish the VPLS "1543" between the ends. A single VPN service is created for this purpose. Additionally, two VPN Nodes and each with a corresponding VPN network access is also created.


```
================ NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "450",
          "vpn-name": "CORPO-EXAMPLE",
          "vpn-description": "SEDE_CENTRO_450",
          "customer-name": "EXAMPLE",
          "vpn-type": "ietf-vpn-common:vpls",
          "vpn-service-topology": "ietf-vpn-common:hub-spoke",
          "bgp-ad-enabled": false,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
```

```
                "profile-id": "simple-profile",
                "ce-vlan-preservation": true,
                "ce-vlan-cos-preservation": true
              }
            ]
          },
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "450",
                "description": "SEDE_CENTRO_450",
                "ne-id": "2001:db8:5::1",
                "role": "ietf-vpn-common:hub-role",
                "status": {
                  "admin-status": {
                    "status": "ietf-vpn-common:admin-up"
                  }
                },
                "active-global-parameters-profiles": {
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "signaling-option": {
                  "ldp-or-l2tp": {
                    "t-ldp-pw-type": "vpls-type",
                    "pw-peer-list": [
                      {
                        "peer-addr": "2001:db8:50::1",
                        "vc-id": "1543"
                      }
                    ]
                  }
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "4508671287",
                      "description": "VPN_450_SNA",
                      "interface-id": "gigabithethernet0/0/1",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
                      "connection": {
```

```
                    "l2-termination-point": "550",
                    "encapsulation": {
                      "encap-type": "ietf-vpn-common:dot1q",
                      "dot1q": {
                        "tag-type": "ietf-vpn-common:c-vlan",
                        "cvlan-id": 550
                      }
                    }
                  },
                  "service": {
                    "mtu": 1550,
                    "svc-pe-to-ce-bandwidth": {
                      "pe-to-ce-bandwidth": [
                        {
                          "bw-type": "ietf-vpn-common:bw-per-port",
                          "cir": "20480000"
                        }
                      ]
                    },
                    "svc-ce-to-pe-bandwidth": {
                      "ce-to-pe-bandwidth": [
                        {
                          "bw-type": "ietf-vpn-common:bw-per-port",
                          "cir": "20480000"
                        }
                      ]
                    },
                    "qos": {
                      "qos-profile": {
                        "qos-profile": [
                          {
                            "profile": "QoS_Profile_A",
                            "direction": "ietf-vpn-common:both"
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            },
            {
              "vpn-node-id": "451",
              "description": "SEDE_CHAPINERO_451",
              "ne-id": "2001:db8:50::1",
              "role": "ietf-vpn-common:spoke-role",
              "status": {
```

```
      "admin-status": {
        "status": "ietf-vpn-common:admin-up"
      }
    },
    "active-global-parameters-profiles": {
      "global-parameters-profile": [
        {
          "profile-id": "simple-profile"
        }
      ]
    },
    "signaling-option": {
      "ldp-or-l2tp": {
        "t-ldp-pw-type": "vpls-type",
        "pw-peer-list": [
          {
            "peer-addr": "2001:db8:5::1",
            "vc-id": "1543"
          }
        ]
      }
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "4508671288",
          "description": "VPN_450_SNA",
          "interface-id": "gigabithethernet0/0/1",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          },
          "connection": {
            "l2-termination-point": "550",
            "encapsulation": {
              "encap-type": "ietf-vpn-common:dot1q",
              "dot1q": {
                "tag-type": "ietf-vpn-common:c-vlan",
                "cvlan-id": 550
              }
            }
          },
          "service": {
            "mtu": 1550,
            "svc-pe-to-ce-bandwidth": {
              "pe-to-ce-bandwidth": [
                {
```

```
                                "bw-type": "ietf-vpn-common:bw-per-port",
                                "cir": "20480000"
                              }
                            ]
                          },
                          "svc-ce-to-pe-bandwidth": {
                            "ce-to-pe-bandwidth": [
                              {
                                "bw-type": "ietf-vpn-common:bw-per-port",
                                "cir": "20480000"
                              }
                            ]
                          },
                          "qos": {
                            "qos-profile": {
                              "qos-profile": [
                                {
                                  "profile": "QoS_Profile_A",
                                  "direction": "ietf-vpn-common:both"
                                }
                              ]
                            }
                          }
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }
        ]
      }
    }
  }
```


Figure 28: Example of L2NM Message Body for LDP-based VPLS

A.4. VPWS-EVPN Service Instance

Figure 29 depicts a sample architecture to offer VPWS-EVPN service between CE1 and CE2. Both CEs are multi-homed. BGP sessions are maintained between these PEs as per [RFC8214]. In this EVPN instance, an All-Active redundancy mode is used.

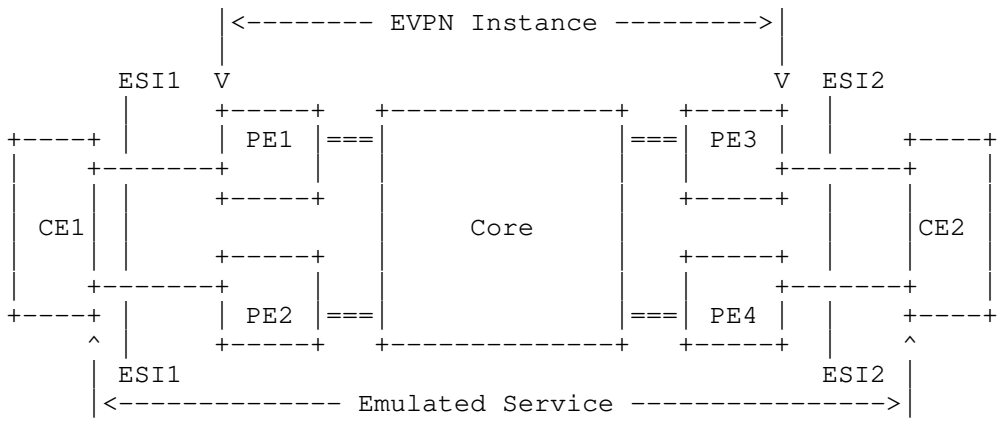


Figure 29: An Example of VPWS-EVPN

Let's first suppose that the following ES was created (Figure 30).


```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "esi1",
        "ethernet-segment-identifier": "00:11:11:11:11:11:11:\
11:11:11",
        "esi-redundancy-mode": "all-active"
      },
      {
        "name": "esi2",
        "ethernet-segment-identifier": "00:22:22:22:22:22:22:\
22:22:22",
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```


Figure 30: Example of L2NM Message Body to Configure an Ethernet Segment

Figure 29 shows a simplified configuration to illustrate the use of the L2NM to configured VPWS-EVPN instance.

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpws15432855",
          "vpn-description": "Sample VPWS-EVPN",
          "customer-name": "customer_15432855",
          "vpn-type": "ietf-vpn-common:vpws-evpn",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:bgp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65535,
                "rd-suffix": 1,
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
              }
            ]
          },
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "pe1",
                "ne-id": "198.51.100.1",
                "active-global-parameters-profiles": {
                  "global-parameters-profile": [
                    {
                      "profile-id": "simple-profile"
                    }
                  ]
                },
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
```

```
                    "description": "Interface to CE1",
                    "active-vpn-node-profile": "simple-profile",
                    "status": {
                      "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                      }
                    },
                    "connection": {
                      "encapsulation": {
                        "encap-type": "ietf-vpn-common:dot1q",
                        "dot1q": {
                          "cvlan-id": 1
                        }
                      }
                    },
                    "vpws-service-instance": {
                      "local-vpws-service-instance": 1111,
                      "remote-vpws-service-instance": 1112
                    },
                    "group": [
                      {
                        "group-id": "gr1",
                        "ethernet-segment-identifier": "esi1"
                      }
                    ]
                  }
                ]
              }
            },
            {
              "vpn-node-id": "pe2",
              "ne-id": "198.51.100.2",
              "active-global-parameters-profiles": {
                "global-parameters-profile": [
                  {
                    "profile-id": "simple-profile"
                  }
                ]
              },
              "vpn-network-accesses": {
                "vpn-network-access": [
                  {
                    "id": "1/1/1.1",
                    "interface-id": "1/1/1",
                    "description": "Interface to CE1",
                    "active-vpn-node-profile": "simple-profile",
                    "status": {
                      "admin-status": {
```

```
                        "status": "ietf-vpn-common:admin-up"
                      }
                    },
                    "connection": {
                      "encapsulation": {
                        "encap-type": "ietf-vpn-common:dot1q",
                        "dot1q": {
                          "cvlan-id": 1
                        }
                      }
                    },
                    "vpws-service-instance": {
                      "local-vpws-service-instance": 1111,
                      "remote-vpws-service-instance": 1112
                    },
                    "group": [
                      {
                        "group-id": "gr1",
                        "ethernet-segment-identifier": "esi1"
                      }
                    ]
                  }
                ]
              }
            },
            {
              "vpn-node-id": "pe3",
              "ne-id": "198.51.100.3",
              "active-global-parameters-profiles": {
                "global-parameters-profile": [
                  {
                    "profile-id": "simple-profile"
                  }
                ]
              },
              "vpn-network-accesses": {
                "vpn-network-access": [
                  {
                    "id": "1/1/1.1",
                    "interface-id": "1/1/1",
                    "description": "Interface to CE2",
                    "active-vpn-node-profile": "simple-profile",
                    "status": {
                      "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                      }
                    },
                    "connection": {
```

```
                    "encapsulation": {
                      "encap-type": "ietf-vpn-common:dot1q",
                      "dot1q": {
                        "cvlan-id": 1
                      }
                    }
                  },
                  "vpws-service-instance": {
                    "local-vpws-service-instance": 1112,
                    "remote-vpws-service-instance": 1111
                  },
                  "group": [
                    {
                      "group-id": "gr1",
                      "ethernet-segment-identifier": "esi2"
                    }
                  ]
                }
              ]
            }
          },
          {
            "vpn-node-id": "pe4",
            "ne-id": "198.51.100.4",
            "active-global-parameters-profiles": {
              "global-parameters-profile": [
                {
                  "profile-id": "simple-profile"
                }
              ]
            },
            "vpn-network-accesses": {
              "vpn-network-access": [
                {
                  "id": "1/1/1.1",
                  "interface-id": "1/1/1",
                  "description": "Interface to CE2",
                  "active-vpn-node-profile": "simple-profile",
                  "status": {
                    "admin-status": {
                      "status": "ietf-vpn-common:admin-up"
                    }
                  },
                  "connection": {
                    "encapsulation": {
                      "encap-type": "ietf-vpn-common:dot1q",
                      "dot1q": {
                        "cvlan-id": 1
```

```
                            }
                          }
                        },
                        "vpws-service-instance": {
                          "local-vpws-service-instance": 1112,
                          "remote-vpws-service-instance": 1111
                        },
                        "group": [
                          {
                            "group-id": "gr1",
                            "ethernet-segment-identifier": "esi2"
                          }
                        ]
                      }
                    ]
                  }
                }
              ]
            }
          }
        ]
      }
    }
  }
```

Figure 31: Example of L2NM Message Body to Configure a VPWS-EVPN Instance

A.5. Automatic ESI Assignment

This section provides an example to illustrate how the L2NM can be used to manage ESI auto-assignment. We consider the sample EVPN service delivered using the architecture depicted in Figure 32.

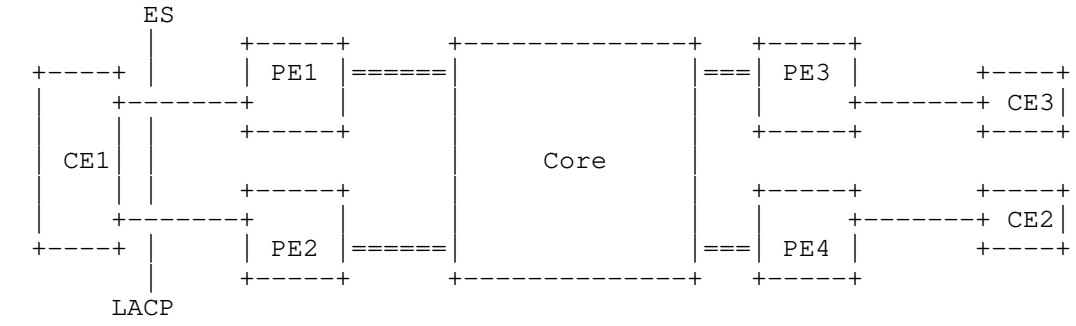


Figure 32: An Example of Automatic ESI Assignment

Figure 33 and Figure 34 show how the L2NM is used to instruct both PE1 and PE2 to auto-assign the ESI to identify the ES used with CE1. In this example, we suppose that LACP is enabled and that a Type 1 (T=0x01) is used as per Section 5 of [RFC7432]. Note that this example does not include all the details to configure the EVPN service, but focuses only on the ESI management part.

```
{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "esi1",
        "esi-type": "esi-type-1-lacp",
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```

Figure 33: Example of L2NM Message Body to Auto-Assign Ethernet Segment Identifiers

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "ietf-l2vpn-ntw:vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "auto-esi-lacp",
          "vpn-description": "Sample to illustrate auto-ESI",
          "vpn-type": "ietf-vpn-common:vpws-evpn",
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "pe1",
                "ne-id": "198.51.100.1",
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE1",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      },
```

```
                    "connection": {
                      "lag-interface": {
                        "lag-interface-id": "1",
                        "lacp": {
                          "lacp-state": true,
                          "system-id": "11:00:11:00:11:11",
                          "admin-key": 154
                        }
                      }
                    },
                    "group": [
                      {
                        "group-id": "gr1",
                        "ethernet-segment-identifier": "esi1"
                      }
                    ]
                  }
                ]
              }
            },
            {
              "vpn-node-id": "pe2",
              "ne-id": "198.51.100.2",
              "vpn-network-accesses": {
                "vpn-network-access": [
                  {
                    "id": "2/2/2.5",
                    "interface-id": "2/2/2",
                    "description": "Interface to CE1",
                    "status": {
                      "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                      }
                    },
                    "connection": {
                      "lag-interface": {
                        "lag-interface-id": "1",
                        "lacp": {
                          "lacp-state": true,
                          "system-id": "11:00:11:00:11:11",
                          "admin-key": 154
                        }
                      }
                    },
                    "group": [
                      {
                        "group-id": "gr1",
                        "ethernet-segment-identifier": "esi1"
```

```
                            }
                          ]
                        }
                      ]
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    }
  }
```

Figure 34: An Example of L2NM Message Body for ESI Auto-Assignment

The auto-assigned ESI can be retrieved using, e.g., a GET RESTCONF method. The assigned value will be then returned as shown in the 'esi-auto' data node in Figure 35.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "esi1",
        "ethernet-segment-identifier": "esi-type-1-lacp",
        "esi-auto": {
          "auto-ethernet-segment-identifier": "01:11:00:11:00:11:\
11:9a:00:00"
        },
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```

Figure 35: An Example of L2NM Message Body to Retrieve the Assigned ESI

A.6. VPN Network Access Precedence

In reference to the example depicted in Figure 36, an L2VPN service involves two VPN network accesses to sites that belong to the same customer.

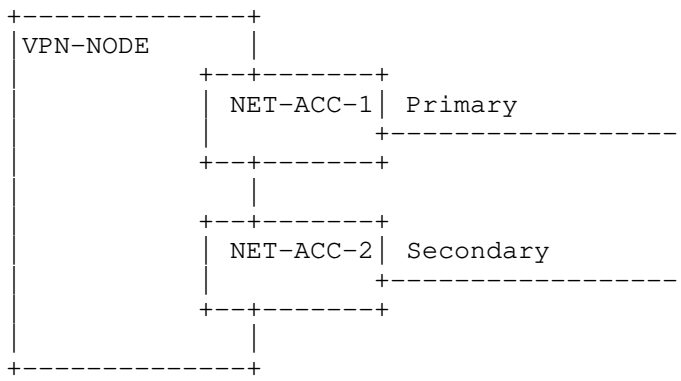


Figure 36: Example of Multiple VPN Network Accesses

In order to tag one of these VPN network accesses as "primary" and the other one as "secondary", Figure 37 shows an excerpt of the corresponding L2NM configuration. In such a configuration, both accesses are bound to the same "group-id" and the "precedence" data node set as function of the intended role of each access (primary or secondary).

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "Sample-Service",
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "VPN-NODE",
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "NET-ACC-1",
                      "connection": {
                        "bearer-reference": "br1"
                      },
                      "group": [
                        {
                          "group-id": "1",
                          "precedence": "primary"
                        }
                      ]
                    },
                    {
                      "id": "NET-ACC-2",
                      "connection": {
                        "bearer-reference": "br2"
                      },
                      "group": [
                        {
                          "group-id": "1",
                          "precedence": "secondary"
                        }
                      ]
                    }
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

Figure 37: Example of Message Body to Associate Priority Levels with VPN Network Accesses

Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from: Sergio Belotti, Italo Busi, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Moti Morgenstern, Erez Segev, and Tom Petch. Many thanks to them.

Luay Jalil, Jichun Ma, Daniel King, and Zhang Guiyu contributed to an early version of this document.

Thanks to Yingzhen Qu and Himanshu Shah for the rtgdir reviews, Ladislav Lhotka for the yangdoctors review, Chris Lonvick for the secdir review, and Dale Worley for the gen-art review. Special thanks to Adrian Farrel for the careful Shepherd review.

Thanks to Robert Wilton for the careful AD review and various suggestions to enhance the model.

Thanks to Lars Eggert, Erik Kline, Roman Danyliw, Francesca Palombini, Zaheduzzaman Sarker, and Eric Vyncke for the IESG review.

A YANG module for Ethernet segments was first defined in the context of the EVPN device module [I-D.ietf-bess-evpn-yang].

This work is partially supported by the European Commission under Horizon 2020 grant agreement number 101015857 Secured autonomic traffic management for a Tera of SDN flows (Teraflow).

Contributors

Victor Lopez
Nokia
Email: victor.lopez@nokia.com

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

Raul Arco
Nokia
Email: raul.arco@nokia.com

Authors' Addresses

Mohamed Boucadair (editor)
Orange
Rennes
France
Email: mohamed.boucadair@orange.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
Spain
Email: oscar.gonzalezdedios@telefonica.com

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

Luis Angel Munoz
Vodafone
Spain
Email: luis-angel.munoz@vodafone.com