

opsawg
Internet-Draft
Updates: 8782 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

S. Barguil
O. Gonzalez de Dios, Ed.
Telefonica
M. Boucadair, Ed.
Orange
Q. Wu
Huawei
July 13, 2020

A Layer 2/3 VPN Common YANG Model
draft-bgbw-opsawg-vpn-common-00

Abstract

This document defines a common YANG module that is meant to be reused by various VPN-related modules such as Layer 3 VPN Service Model, Layer 2 VPN Service Model, Layer 3 VPN Network Model, and Layer 2 VPN Network Model.

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:

- o "This version of this YANG module is part of RFC XXXX;"
- o "RFC XXXX: A Layer 2/3 VPN Common YANG Model";
- o reference: RFC XXXX

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

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 January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

- 1. Introduction 2
- 2. Terminology 5
- 3. Description of the VPN Common YANG Module 5
- 4. Layer 2/3 VPN Common Module 8
- 5. Security Considerations 31
- 6. IANA Considerations 31
- 7. Contributors 32
- 8. References 32
 - 8.1. Normative References 32
 - 8.2. Informative References 33
- Authors' Addresses 34

1. Introduction

Various VPN-related YANG data modules were specified by the IETF (e.g., Layer 3 VPN Service Model (L3SM) [RFC8299] or Layer 2 VPN Service Model (L2SM) [RFC8466]). Others are also being specified (e.g., Layer 3 VPN Network Model (L3NM) [I-D.ietf-opsawg-l3sm-l3nm] or Layer 2 VPN Network Model (L2NM) [I-D.ietf-opsawg-l2nm]). These modules have data nodes and structures that are present in almost all these models or a subset of them. An example of such data nodes is depicted in Figure 1.

```

module: ietf-l2vpn-ntw
  +--rw vpn-services
    +--rw vpn-service* [vpn-id]
      +--rw vpn-id                               svc-id
      +--rw vpn-svc-type?                       identityref
      +--rw customer-name?                     string
      +--rw svc-topo?                          identityref
      +--rw service-status
        |
        | +--rw admin
        | |
        | | +--rw status?                       operational-type
        | | +--rw timestamp?                   yang:date-and-time
        | |
        | | +--ro ops
        | | +--ro status?                       operational-type
        | | +--ro timestamp?                   yang:date-and-time
        | |
        | | ...
        |
        +--rw admin
          |
          | +--rw status?                       operational-type
          | +--rw timestamp?                   yang:date-and-time
          |
          +--ro ops
            +--ro status?                       operational-type
            +--ro timestamp?                   yang:date-and-time
            +--ro vpn-id                       l3vpn-svc:svc-id
            +--ro l3sm-vpn-id?                 l3vpn-svc:svc-id
            +--ro customer-name?              string
            +--ro vpn-service-topology?      identityref
            +--ro description?                string
            |
            ...

```

Figure 1: Example of Common Data Nodes in Both L2NM/L3NM

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

Likewise, early versions of the L2NM reused many of the data nodes that are defined in both L2SM and L3NM. An example of L3NM groupings reused in L3NM is shown in Figure 2. This data nodes reuse was

interpreted as if the deployment of L2NM requires both L3NM; which is not required.

```
ietf-l2vpn-ntw {
  ...
  import ietf-l3vpn-ntw {
    prefix l3vpn-ntw;
    reference
      "RFC NNNN: A Layer 3 VPN Network YANG Model";
  }
  ...
  container l2vpn-ntw {
    ...
    container vpn-services {
      list vpn-service {
        ...
        uses l3vpn-ntw:service-status;
        uses l3vpn-ntw:svc-transport-encapsulation;
        ...
      }
    }
    ...
  }
}
```

Figure 2: Excerpt from the L2NM YANG Module

To avoid the issues discussed above, this document defines a common YANG module that is meant to be reused by various VPN-related modules such as Layer 3 VPN Service Model (L3SM) [RFC8299], Layer 2 VPN Service Model (L2SM) [RFC8466], Layer 3 VPN Network Model (L3NM) [I-D.ietf-opsawg-l3sm-l3nm], and Layer 2 VPN Network Model (L2NM) [I-D.ietf-opsawg-l2nm]: "ietf-vpn-common" (Section 4).

The "ietf-vpn-common" module includes a set of identities, types, and groupings that are meant to be reused by other VPN-related YANG modules independently of their 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., L3SM [RFC8299] or L2SM [RFC8466]).

The approach that is followed for building the common module (Section 4) is to first extract data nodes that are common for both L3NM and L3SM; these data nodes are then filtered out against Layer 2 modules. All the common groupings are called, for example, in the L3NM module defined in [I-D.ietf-opsawg-l3sm-l3nm].

2. Terminology

The terminology for describing YANG modules is defined in [RFC7950].

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

3. Description of the VPN Common YANG Module

The `ietf-vpn-common` contains the following reusable groupings and identities:

Groupings:

- o `vpn-description`:

- A YANG grouping that provides common administrative VPN information such as name, a textual description and the customer name.

- o `vpn-profile-cfg`:

- A YANG grouping that defines the of profiles (encryption, routing, forwarding) valid for any L2/L3 VPN.

- o `status-timestamp`:

- A YANG grouping that defines operational and administrative updates of a component.

- o `service-status`:

- A YANG grouping that defines the administrative and operational status of a component. The grouping can be applied to the whole service of e.g. and end point.

- o `svc-transport-encapsulation`:

- A YANG grouping that defines the type of underlay transport for a VPN service.

- o `rt-rd`:

- A YANG grouping that defines the set of route-targets to match for import and export routes to/from VRF.

- o `vpn-node-group`:

- A YANG grouping that is used to group `vpn-network-` access.

Identities

- o bw-direction:Identity for the bandwidth direction.
- o qos-profile-direction:Base identity for QoS profile direction.
- o customer-application:Base identity for customer application.
- o ie-type:Defines Import-Export routing profiles.
- o site-network-access-type:Base identity for site-network-access type.
- o operational-status:Base identity for the operational status.
- o administrative-status:Base identity for administrative status.
- o encapsulation-type:Base identity for encapsulation type.
- o tag-type:Base identity from which all tag types are derived.
- o protocol-type:Base identity for Protocol Type.
- o vpn-topology:Base identity for VPN topology.
- o role:Base identity for site or node type.
- o vpn-signaling-type:Identity of VPN signaling types
- o service-type:Identity of service type.
- o vxlan-peer-mode:Base identity for the VXLAN peer mode.
- o multicast-gp-address-mapping:Identity for multicast group mapping type.
- o multicast-tree-type:Base identity for multicast tree type.
- o multicast-rp-discovery-type:Base identity for RP discovery type.

The tree diagram of the "ietf-vpn-common" module that depicts the common groupings is provided in Figure 3. The descriptions of these groupings are provided in the description statements in Section 4.

```
module: ietf-vpn-common
  grouping vpn-description
    +-- vpn-id?          vpn-common:vpn-id
```

```

    +-- vpn-name?          string
    +-- vpn-description?  string
    +-- customer-name?   string
grouping vpn-profile-cfg
  +-- valid-provider-identifiers
    +-- cloud-identifier* [id] {cloud-access}?
      | +-- id?  string
    +-- encryption-profile-identifier* [id]
      | +-- id?  string
    +-- qos-profile-identifier* [id]
      | +-- id?  string
    +-- bfd-profile-identifier* [id]
      | +-- id?  string
    +-- forwarding-profile-identifier* [id]
      | +-- id?  string
    +-- routing-profile-identifier* [id]
      | +-- id?  string
grouping status-timestamp
  +-- status?          identityref
  +-- last-updated?   yang:date-and-time
grouping service-status
  +-- status
    +-- admin-status
      | +-- status?          identityref
      | +-- last-updated?   yang:date-and-time
    +--ro oper-status
      +--ro status?          identityref
      +--ro last-updated?   yang:date-and-time
grouping svc-transport-encapsulation
  +-- underlay-transport
    +-- type*  identityref
grouping rt-rd
  +-- rd?          union
  +-- vpn-targets
    +-- vpn-target* [id]
      | +-- id?          int8
      | +-- route-targets* [route-target]
      | | +-- route-target?  rt-types:route-target
      | +-- route-target-type  rt-types:route-target-type
    +-- vpn-policies
      +-- import-policy?  string
      +-- export-policy?  string
grouping vpn-route-targets
  +-- vpn-target* [id]
    | +-- id?          int8
    | +-- route-targets* [route-target]
    | | +-- route-target?  rt-types:route-target
    | +-- route-target-type  rt-types:route-target-type

```

```

    +-- vpn-policies
       +-- import-policy?  string
       +-- export-policy?  string
grouping vpn-node-group
    +-- groups
       +-- group* [group-id]
          +-- group-id?  string

```

Figure 3: VPN Common Tree

4. Layer 2/3 VPN Common Module

This module uses types defined in [RFC6991] and [RFC8294].

Editor's Note: RFCs cited in the reference statements will be added to the References Section in future versions.

```

<CODE BEGINS> file "ietf-vpn-common@2020-07-13.yang"
module ietf-vpn-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-vpn-common";
  prefix vpn-common;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
}

organization
  "IETF OPSA (Operations and Management Area) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/opsawg/>";
  WG List:  <mailto:opsawg@ietf.org>
  Editor:   Samier Barguil
            <mailto:samier.barguilgiraldo.ext@telefonica.com>
  Editor:   Oscar Gonzalez de Dios
            <mailto:oscar.gonzalezdedios@telefonica.com>

```

Editor: Mohamed Boucadair
<mailto:mohamed.boucadair@orange.com>
Author: Qin Wu
<mailto:bill.wu@huawei.com>

```
";
description
  "This YANG module defines a common module that is meant
  to be reused by various VPN-related modules (e.g.,
  Layer 3 VPN Service Model (L3SM), Layer 2 VPN Service
  Model (L2SM), Layer 3 VPN Network Model (L3NM), Layer 2
  VPN Network Model (L2NM)).

  Copyright (c) 2020 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision 2020-07-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A Layer 2/3 VPN Common YANG Model";
}

/* Features */

feature cloud-access {
  description
    "Indicates support of the VPN to connect to a Cloud
    Service Provider (CSP).";
}

feature lag-interface {
  description
    "Indicates the support of Link aggregation between
    Site Network Accesses. ";
}

feature site-diversity {
```

```
    description
      "Indicates the site diversity in the customer premises.";
  }

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

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

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

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

  feature multicast {
    description
      "Indicates multicast capabilities support in a VPN.";
  }

  feature ipv4 {
    description
      "Indicates IPv4 support in a VPN.";
  }

  feature ipv6 {
    description
      "Indicates IPv6 support in a VPN.";
  }

  feature carrierscarrier {
    description
      "Indicates support of Carrier-of-Carrier VPNs.";
  }
}
```

```
feature extranet-vpn {
  description
    "Indicates support of extranet VPNs.";
}

feature fast-reroute {
  description
    "Indicates support of Fast Reroute (FRR).";
}

feature qos {
  description
    "Indicates support of classes of services (CoSes).";
}

feature encryption {
  description
    "Indicates support of encryption.";
}

feature bfd {
  description
    "Indicates support of BFD.";
}

feature bearer-reference {
  description
    "Indicates support of the 'bearer-reference' access
    constraint.";
}

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

/* Typedef */

typedef vpn-id {
  type string;
  description
    "Defines an identifier that is used as
    a service identifier, for example.";
}

typedef address-family {
  type enumeration {
```

```
    enum ipv4 {
        description
            "IPv4 address family.";
    }
    enum ipv6 {
        description
            "IPv6 address family.";
    }
}
description
    "Defines a type for the address family.";
}

/* Identities */

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

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

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

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

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

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

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

identity customer-application {
  description
    "Base identity for customer application.";
}

identity web {
  base customer-application;
  description
    "Identity for Web application (e.g., HTTP, HTTPS).";
}

identity mail {
  base customer-application;
  description
    "Identity for mail application.";
}

identity file-transfer {
  base customer-application;
  description
    "Identity for file transfer application (e.g., FTP, SFTP).";
}

identity database {
  base customer-application;
  description
    "Identity for database application.";
}

identity social {
  base customer-application;
  description
    "Identity for social-network application.";
}

identity games {
  base customer-application;
  description
    "Identity for gaming application.";
}
```

```
identity p2p {
  base customer-application;
  description
    "Identity for peer-to-peer application.";
}

identity network-management {
  base customer-application;
  description
    "Identity for management application
    (e.g., Telnet, syslog, SNMP).";
}

identity voice {
  base customer-application;
  description
    "Identity for voice application.";
}

identity video {
  base customer-application;
  description
    "Identity for video conference application.";
}

identity embb {
  base customer-application;
  description
    "Identity for an enhanced Mobile Broadband (eMBB)
    application. Note that an eMBB application demands
    network performance with a wide variety of
    characteristics, such as data rate, latency,
    loss rate, reliability, and many other parameters.";
}

identity urllic {
  base customer-application;
  description
    "Identity for an Ultra-Reliable and Low Latency
    Communications (URLLC) application. Note that a
    URLLC application demands network performance
    with a wide variety of characteristics, such as latency,
    reliability, and many other parameters.";
}

identity mmhc {
  base customer-application;
  description
```

```
    "Identity for a massive Machine Type
    Communications (mMTC) application. Note that an
    mMTC application demands network performance
    with a wide variety of characteristics, such as data
    rate, latency, loss rate, reliability, and many
    other parameters.";
}

identity ie-type {
  description
    "Defines Import-Export routing profiles.
    Those profiles can be reused between VPN nodes.";
}

identity import {
  base ie-type;
  description
    "Import a routing profile.";
}

identity export {
  base ie-type;
  description
    "Export a routing profile.";
}

identity import-export {
  base ie-type;
  description
    "Import/Export a routing profile.";
}

identity site-network-access-type {
  description
    "Base identity for site-network-access type.";
}

identity point-to-point {
  base site-network-access-type;
  description
    "Identity for point-to-point connection.";
}

identity multipoint {
  base site-network-access-type;
  description
    "Identity for multipoint connection.
    Example: Ethernet broadcast segment.";
```

```
}  
  
identity pseudowire {  
  base site-network-access-type;  
  description  
    "Identity for pseudowire connections.";  
}  
  
identity loopback {  
  base site-network-access-type;  
  description  
    "Identity for loopback connections.";  
}  
  
identity operational-status {  
  description  
    "Base identity for the operational status.";  
}  
  
identity operational-state-up {  
  base operational-status;  
  description  
    "Operational status is UP/Enabled.";  
}  
  
identity operational-state-down {  
  base operational-status;  
  description  
    "Operational status is DOWN/Disabled.";  
}  
  
identity operational-state-unknown {  
  base operational-status;  
  description  
    "Operational status is UNKNOWN.";  
}  
  
identity administrative-status {  
  description  
    "Base identity for administrative status.";  
}  
  
identity administrative-state-up {  
  base administrative-status;  
  description  
    "Administrative status is UP/Enabled.";  
}
```

```
identity administrative-state-down {
  base administrative-status;
  description
    "Administrative status is DOWN/Disabled.";
}

identity administrative-state-testing {
  base administrative-status;
  description
    "Administrative status is up for testing purposes.";
}

identity administrative-state-pre-deployment {
  base administrative-status;
  description
    "Administrative status is pre-deployment phase.";
}

identity encapsulation-type {
  description
    "Base identity for encapsulation type.";
}

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

identity dot1q {
  base encapsulation-type;
  description
    "This identity indicates the support of
    the 'dot1q' encapsulation.";
}

identity qinq {
  base encapsulation-type;
  description
    "This identity indicates the support of
    the 'qinq' encapsulation.";
}

identity qinany {
  base encapsulation-type;
  description
    "This identity indicates the support of
    the 'qinany' encapsulation.";
```

```
}  
  
identity vxlan {  
    base encapsulation-type;  
    description  
        "This identity indicates the support of  
        the 'vxlan' encapsulation.";  
}  
  
identity ethernet-type {  
    base encapsulation-type;  
    description  
        "Identity for encapsulation type.";  
}  
  
identity vlan-type {  
    base encapsulation-type;  
    description  
        "Identity for VLAN encapsulation.";  
}  
  
identity untagged-int {  
    base encapsulation-type;  
    description  
        "Identity for Ethernet type.";  
}  
  
identity tagged-int {  
    base encapsulation-type;  
    description  
        "Identity for the VLAN type.";  
}  
  
identity lag-int {  
    base encapsulation-type;  
    description  
        "Identity for the VLAN type.";  
}  
  
identity tag-type {  
    description  
        "Base identity from which all tag types are derived.";  
}  
  
identity c-vlan {  
    base tag-type;  
    description  
        "A CVLAN tag, normally using the 0x8100 Ethertype.";
```

```
}

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

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

identity protocol-type {
  description
    "Base identity for Protocol Type.";
}

identity gre {
  base protocol-type;
  description
    "GRE encapsulation.";
  reference
    "RFC 1701: Generic Routing Encapsulation (GRE)
     RFC 1702: Generic Routing Encapsulation over IPv4 networks
     RFC 7676: IPv6 Support for Generic Routing Encapsulation
     (GRE)";
}

identity ldp {
  base protocol-type;
  description
    "Transport based on LDP.";
  reference
    "RFC 3086: LDP Specification";
}

identity sr {
  base protocol-type;
  description
    "Transport based on SR.";
  reference
    "RFC 8660: Segment Routing with the MPLS Data Plane
     RFC 8663: MPLS Segment Routing over IP
     RFC 8754: IPv6 Segment Routing Header (SRH)";
}
```

```
identity sr-te {
  base protocol-type;
  description
    "Transport based on SR-TE.";
  reference
    "RFC 8426: Recommendations for RSVP-TE and Segment Routing (SR)
    Label Switched Path (LSP) Coexistence";
}

identity rsvp-te {
  base protocol-type;
  description
    "Transport based on RSVP-TE.";
  reference
    "RFC 2205: Resource ReSerVation Protocol (RSVP) --
    Version 1 Functional Specification";
}

identity bgp-lu {
  base protocol-type;
  description
    "Transport based on BGP-LU.";
}

identity unknown {
  base protocol-type;
  description
    "Not known at this stage.";
}

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

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

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

identity hub-spoke-disjoint {
```

```
    base vpn-topology;
    description
      "Identity for Hub-and-Spoke VPN topology
       where Hubs cannot communicate with each other.";
  }

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

  identity role {
    description
      "Base identity for site or node type.";
  }

  identity any-to-any-role {
    base role;
    description
      "VPN-Node in an any-to-any IP VPN.";
  }

  identity spoke-role {
    base role;
    description
      "VPN-Node acting as a Spoke IP VPN.";
  }

  identity hub-role {
    base role;
    description
      "VPN-Node acting as a Hub IP VPN.";
  }

  identity custom-role {
    base role;
    description
      "VPN-Node with custom or complex role in the VPN.";
  }

  identity vpn-signaling-type {
    description
      "Identity of VPN signaling types";
  }
}
```

```
identity l2vpn-bgp {
  base vpn-signaling-type;
  description
    "Identity of l2vpn-bgp";
}

identity evpn-bgp {
  base vpn-signaling-type;
  description
    "Identity of evpn-bgp";
}

identity t-ldp {
  base vpn-signaling-type;
  description
    "Identity of t-ldp.";
}

identity h-vpls {
  base vpn-signaling-type;
  description
    "Identity for h-vpls";
}

identity l2tp {
  base vpn-signaling-type;
  description
    "Identity of l2tp.";
}

identity service-type {
  description
    "Identity of service type.";
}

identity l3vpn {
  base service-type;
  description
    "Identity of L3VPN service.";
}

identity vpws {
  base service-type;
  description
    "Point-to-point Virtual Private Wire Service (VPWS)
    service type.";
}
```

```
identity pwe3 {
  base service-type;
  description
    "Pseudowire Emulation Edge to Edge (PWE3) service type.";
}

identity ldp-l2tp-vpls {
  base service-type;
  description
    "LDP-based or L2TP-based multipoint Virtual Private LAN
    Service (VPLS) service type. This VPLS uses LDP-signaled
    Pseudowires or L2TP-signaled Pseudowires.";
}

identity bgp-vpls {
  base service-type;
  description
    "BGP-based multipoint VPLS service type. This VPLS uses a
    BGP control plane.";
  reference
    "RFC4761: Virtual Private LAN Service (VPLS) Using
    BGP for Auto-Discovery and Signaling
    RFC 6624: Layer 2 Virtual Private Networks Using BGP for
    Auto-Discovery and Signaling";
}

identity vpws-evpn {
  base service-type;
  description
    "VPWS service type using Ethernet VPNs (EVPNs).";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN";
}

identity pbb-evpn {
  base service-type;
  description
    "PBB EVPN.";
}

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

identity static-mode {
  base vxlan-peer-mode;
  description
```

```
    "Identity for VXLAN access in the static mode.";
}

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

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

identity static-mapping {
    base multicast-gp-address-mapping;
    description
        "Identity for static mapping, i.e., attach the interface
        to the multicast group as a static member.";
}

identity dynamic-mapping {
    base multicast-gp-address-mapping;
    description
        "Identity for dynamic mapping, i.e., an interface was added
        to the multicast group as a result of snooping.";
}

identity multicast-tree-type {
    description
        "Base identity for multicast tree type.";
}

identity ssm-tree-type {
    base multicast-tree-type;
    description
        "Identity for SSM tree type.";
}

identity asm-tree-type {
    base multicast-tree-type;
    description
        "Identity for ASM tree type.";
}

identity bidir-tree-type {
    base multicast-tree-type;
    description
```

```
    "Identity for bidirectional tree type.";
}

identity multicast-rp-discovery-type {
  description
    "Base identity for RP discovery type.";
}

identity auto-rp {
  base multicast-rp-discovery-type;
  description
    "Base identity for Auto-RP discovery type.";
}

identity static-rp {
  base multicast-rp-discovery-type;
  description
    "Base identity for static type.";
}

identity bsr-rp {
  base multicast-rp-discovery-type;
  description
    "Base identity for BSR discovery type.";
}

/* Grouping */

grouping vpn-description {
  leaf vpn-id {
    type vpn-common:vpn-id;
    description
      "VPN identifier.
      This identifier has a local meaning.";
  }
  leaf vpn-name {
    type string;
    description
      "A name used to refer to the VPN.";
  }
  leaf vpn-description {
    type string;
    description
      "Textual description of a VPN service.";
  }
  leaf customer-name {
    type string;
    description

```

```
        "Name of the customer that actually uses the VPN service.";
    }
    description
        "Provides common VPN information.";
}

grouping vpn-profile-cfg {
    container valid-provider-identifiers {
        list cloud-identifier {
            if-feature "cloud-access";
            key "id";
            leaf id {
                type string;
                description
                    "Identification of cloud service.
                     Local administration meaning.";
            }
            description
                "List for Cloud Identifiers.";
        }
        list encryption-profile-identifier {
            key "id";
            leaf id {
                type string;
                description
                    "Identification of the SP encryption profile
                     to be used. Local administration meaning.";
            }
            description
                "List for encryption profile identifiers.";
        }
        list qos-profile-identifier {
            key "id";
            leaf id {
                type string;
                description
                    "Identification of the QoS Profile to be used.
                     Local administration meaning.";
            }
            description
                "List for QoS Profile Identifiers.";
        }
        list bfd-profile-identifier {
            key "id";
            leaf id {
                type string;
                description
                    "Identification of the SP BFD Profile to be used.
```

```
        Local administration meaning.";
    }
    description
        "List for BFD Profile identifiers.";
}
list forwarding-profile-identifier {
    key "id";
    leaf id {
        type string;
        description
            "Identification of the Forwrding Profile Filter to be used.
            Local administration meaning.";
    }
    description
        "List for Forwrding Profile identifiers.";
}
list routing-profile-identifier {
    key "id";
    leaf id {
        type string;
        description
            "Identification of the routing Profile to be used
            by the routing-protocols within sites, vpn-
            network-accesses or vpn-nodes for refering
            vrf-import/export policies.
            This identifier has a local meaning.";
    }
    description
        "List for Routing Profile Identifiers.";
}
nacm:default-deny-write;
description
    "Container for Valid Provider Identifies.";
}
description
    "Grouping for VPN Profile configuration.";
}

grouping status-timestamp {
    leaf status {
        type identityref {
            base operational-status;
        }
        description
            "Operations status";
    }
    leaf last-updated {
        type yang:date-and-time;
    }
}
```

```
        description
          "Indicates the actual date and time of the service
           status change.";
      }
    description
      "This grouping defines some operational
       parameters for the service.";
  }

  grouping service-status {
    container status {
      container admin-status {
        leaf status {
          type identityref {
            base administrative-status;
          }
          description
            "Administrative service status.";
        }
        leaf last-updated {
          type yang:date-and-time;
          description
            "Indicates the actual date and time of the service
             status change.";
        }
        description
          "Administrative service status.";
      }
      container oper-status {
        config false;
        uses status-timestamp;
        description
          "Operational service status.";
      }
      description
        "Service status.";
    }
    description
      "Service status grouping.";
  }

  grouping svc-transport-encapsulation {
    container underlay-transport {
      leaf-list type {
        type identityref {
          base protocol-type;
        }
        ordered-by user;
      }
    }
  }
}
```

```
        description
            "Protocols used to deliver a VPN service.";
    }
    description
        "Container for the Transport underlay.";
}
description
    "This grouping defines the type of underlay transport
    for VPN service.";
}

grouping rt-rd {
    leaf rd {
        type union {
            type rt-types:route-distinguisher;
            type empty;
        }
        description
            "Route distinguisher value. If this leaf has not been
            configured, the server will auto-assign a route
            distinguisher value and use that value operationally.
            This calculated value is available in the operational
            state.

            Use the empty type to indicate RD has no value and
            is not to be auto-assigned.";
    }
    container vpn-targets {
        description
            "Set of route-targets to match for import and export routes
            to/from VRF";
        uses vpn-route-targets;
    }
    description
        "Grouping for RT and RD.";
}

grouping vpn-route-targets {
    description
        "A grouping that specifies Route Target import-export rules
        used in a BGP-enabled VPN.";
    list vpn-target {
        key "id";
        leaf id {
            type int8;
            description
                "Identifies each VPN Target";
        }
    }
}
```

```
list route-targets {
  key "route-target";
  leaf route-target {
    type rt-types:route-target;
    description
      "Route Target value";
  }
  description
    "List of Route Targets.";
}
leaf route-target-type {
  type rt-types:route-target-type;
  mandatory true;
  description
    "Import/export type of the Route Target.";
}
description
  "L3VPN route targets. AND/OR Operations are available
  based on the RTs assignment.";
}
reference
  "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs)
  RFC4664: Framework for Layer 2 Virtual Private Networks
  (L2VPNs)";
container vpn-policies {
  description
    "VPN policies";
  leaf import-policy {
    type string;
    description
      "Defines the import policy.";
  }
  leaf export-policy {
    type string;
    description
      "Defines the export policy.";
  }
}
}

grouping vpn-node-group {
  container groups {
    list group {
      key "group-id";
      leaf group-id {
        type string;
        description
          "Group-id the vpn-node belongs to.";
      }
    }
  }
}
```

```
    }
    description
      "List of group-ids.";
  }
  description
    "Groups the vpn node and network access belongs to.";
}
description
  "Grouping definition to assign
  group-ids to group or network access.";
}
}
<CODE ENDS>
```

5. Security Considerations

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

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

The "ietf-vpn-common" module defines a set of identities, types, and groupings. These nodes are intended to be reused by other YANG modules. As such, the module does not expose by itself any data nodes which are writable, contain read-only state, or RPCs. As such, there are no additional security issues to be considered relating to the "ietf-vpn-common" module.

6. IANA Considerations

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

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

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

```
name: ietf-vpn-common
namespace: urn:ietf:params:xml:ns:yang:ietf-vpn-common
maintained by IANA: N
prefix: vpn-common
reference: RFC XXXX
```

7. Contributors

Italo Busi
Huawei Technologies
Email: Italo.Busi@huawei.com

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

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

8. References

8.1. Normative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

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

8.2. Informative References

- [I-D.ietf-opsawg-l2nm]
Barguil, S., Dios, O., Boucadair, M., Munoz, L., Jalil, L., and J. Ma, "A Layer 2 VPN Network YANG Model", draft-ietf-opsawg-l2nm-00 (work in progress), July 2020.
- [I-D.ietf-opsawg-l3sm-l3nm]
Barguil, S., Dios, O., Boucadair, M., Munoz, L., and A. Aguado, "A Layer 3 VPN Network YANG Model", draft-ietf-opsawg-l3sm-l3nm-03 (work in progress), April 2020.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [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>>.
- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.

Authors' Addresses

Samier Barguil
Telefonica
Madrid
ES

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
France

Email: "mohamed.boucadair@orange.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: October 25, 2021

B. Claise
Huawei
J. Quilbeuf
Independent
D. Lopez
Telefonica I+D
D. Voyer
Bell Canada
T. Arumugam
Cisco Systems, Inc.
April 23, 2021

Service Assurance for Intent-based Networking Architecture
draft-claise-opsawg-service-assurance-architecture-05

Abstract

This document describes an architecture for Service Assurance for Intent-based Networking (SAIN). This architecture aims at assuring that service instances are correctly running. As services rely on multiple sub-services by the underlying network devices, getting the assurance of a healthy service is only possible with a holistic view of network devices. This architecture not only helps to correlate the service degradation with the network root cause but also the impacted services when a network component fails or degrades.

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 October 25, 2021.

Copyright Notice

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

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

Table of Contents

1. Terminology	2
2. Introduction	5
3. Architecture	6
3.1. Decomposing a Service Instance Configuration into an Assurance Graph	9
3.2. Intent and Assurance Graph	10
3.3. Subservices	11
3.4. Building the Expression Graph from the Assurance Graph	11
3.5. Building the Expression from a Subservice	12
3.6. Open Interfaces with YANG Modules	12
3.7. Handling Maintenance Windows	13
3.8. Flexible Architecture	14
3.9. Timing	15
3.10. New Assurance Graph Generation	15
4. Security Considerations	16
5. IANA Considerations	16
6. Contributors	16
7. Open Issues	16
8. References	16
8.1. Normative References	16
8.2. Informative References	17
Appendix A. Changes between revisions	18
Acknowledgements	18
Authors' Addresses	19

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

SAIN Agent: Component that communicates with a device, a set of devices, or another agent to build an expression graph from a received assurance graph and perform the corresponding computation.

Assurance Graph: DAG representing the assurance case for one or several service instances. The nodes (also known as vertices in the context of DAG) are the service instances themselves and the subservices, the edges indicate a dependency relations.

SAIN collector: Component that fetches or receives the computer-consumable output of the agent(s) and displays it in a user friendly form or process it locally.

DAG: Directed Acyclic Graph.

ECMP: Equal Cost Multiple Paths

Expression Graph: Generic term for a DAG representing a computation in SAIN. More specific terms are:

- o **Subservice Expressions:** expression graph representing all the computations to execute for a subservice.
- o **Service Expressions:** expression graph representing all the computations to execute for a service instance, i.e. including the computations for all dependent subservices.
- o **Global Computation Graph:** expression graph representing all the computations to execute for all services instances (i.e. all computations performed).

Dependency: The directed relationship between subservice instances in the assurance graph.

Informational Dependency: Type of dependency whose score does not impact the score of its parent subservice or service instance(s) in the assurance graph. However, the symptoms should be taken into account in the parent service instance or subservice instance(s), for informational reasons.

Impacting Dependency: Type of dependency whose score impacts the score of its parent subservice or service instance(s) in the assurance graph. The symptoms are taken into account in the parent service instance or subservice instance(s), as the impacting reasons.

Metric: Information retrieved from a network device.

Metric Engine: Maps metrics to a list of candidate metric implementations depending on the target model.

Metric Implementation: Actual way of retrieving a metric from a device.

Network Service YANG Module: describes the characteristics of service, as agreed upon with consumers of that service [RFC8199].

Service Instance: A specific instance of a service.

Service configuration orchestrator: Quoting RFC8199, "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

SAIN Orchestrator: Component of SAIN in charge of fetching the configuration specific to each service instance and converting it into an assurance graph.

Health status: Score and symptoms indicating whether a service instance or a subservice is healthy. A non-maximal score **MUST** always be explained by one or more symptoms.

Health score: Integer ranging from 0 to 100 indicating the health of a subservice. A score of 0 means that the subservice is broken, a score of 100 means that the subservice is perfectly operational.

Subservice: Part of an assurance graph that assures a specific feature or subpart of the network system.

Symptom: Reason explaining why a service instance or a subservice is not completely healthy.

2. Introduction

Network Service YANG Modules [RFC8199] describe the configuration, state data, operations, and notifications of abstract representations of services implemented on one or multiple network elements.

Quoting RFC8199: "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

In other words, service configuration orchestrators deploy Network Service YANG Modules through the configuration of Network Element YANG Modules. Network configuration is based on those YANG data models, with protocol/encoding such as NETCONF/XML [RFC6241], RESTCONF/JSON [RFC8040], gNMI/gRPC/protobuf, etc. Knowing that a configuration is applied doesn't imply that the service is running correctly (for example the service might be degraded because of a failure in the network), the network operator must monitor the service operational data at the same time as the configuration. The industry has been standardizing on telemetry to push network element performance information.

A network administrator needs to monitor her network and services as a whole, independently of the use cases or the management protocols. With different protocols come different data models, and different ways to model the same type of information. When network administrators deal with multiple protocols, the network management must perform the difficult and time-consuming job of mapping data models: the model used for configuration with the model used for monitoring. This problem is compounded by a large, disparate set of data sources (MIB modules, YANG models [RFC7950], IPFIX information elements [RFC7011], syslog plain text [RFC3164], TACACS+ [RFC8907], RADIUS [RFC2865], etc.). In order to avoid this data model mapping, the industry converged on model-driven telemetry to stream the service operational data, reusing the YANG models used for configuration. Model-driven telemetry greatly facilitates the notion of closed-loop automation whereby events from the network drive remediation changes back into the network.

However, it proves difficult for network operators to correlate the service degradation with the network root cause. For example, why does my L3VPN fail to connect? Why is this specific service slow? The reverse, i.e. which services are impacted when a network component fails or degrades, is even more interesting for the operators. For example, which service(s) is(are) impacted when this specific optic dBm begins to degrade? Which application is impacted by this ECMP imbalance? Is that issue actually impacting any other customers?

Intent-based approaches are often declarative, starting from a statement of the "The service works correctly" and trying to enforce it. Such approaches are mainly suited for greenfield deployments.

Instead of approaching intent from a declarative way, this framework focuses on already defined services and tries to infer the meaning of "The service works correctly". To do so, the framework works from an assurance graph, deduced from the service definition and from the network configuration. This assurance graph is decomposed into components, which are then assured independently. The root of the assurance graph represents the service to assure, and its children represent components identified as its direct dependencies; each component can have dependencies as well. The SAIN architecture maintains the correct assurance graph when services are modified or when the network conditions change.

When a service is degraded, the framework will highlight where in the assurance service graph to look, as opposed to going hop by hop to troubleshoot the issue. Not only can this framework help to correlate service degradation with network root cause/symptoms, but it can deduce from the assurance graph the number and type of services impacted by a component degradation/failure. This added value informs the operational team where to focus its attention for maximum return.

This architecture provides the building blocks to assure both physical and virtual entities and is flexible with respect to services and subservices, of (distributed) graphs, and of components (Section 3.8).

3. Architecture

SAIN aims at assuring that service instances are correctly running. The goal of SAIN is to assure that service instances are operating correctly and if not, to pinpoint what is wrong. More precisely, SAIN computes a score for each service instance and outputs symptoms explaining that score, especially why the score is not maximal. The score augmented with the symptoms is called the health status.

The SAIN architecture is a generic architecture, applicable to multiple environments. Obviously wireline but also wireless, including 5G, virtual infrastructure manager (VIM), and even virtual functions. Thanks to the distributed graph design principle, graphs from different environments/orchestrator can be combined together.

As an example of a service, let us consider a point-to-point L2VPN connection (i.e. pseudowire). Such a service would take as parameters the two ends of the connection (device, interface or subinterface, and address of the other end) and configure both devices (and maybe more) so that a L2VPN connection is established between the two devices. Examples of symptoms might be "Interface has high error rate" or "Interface flapping", or "Device almost out of memory".

To compute the health status of such as service, the service is decomposed into an assurance graph formed by subservices linked through dependencies. Each subservice is then turned into an expression graph that details how to fetch metrics from the devices and compute the health status of the subservice. The subservice expressions are combined according to the dependencies between the subservices in order to obtain the expression graph which computes the health status of the service.

The overall architecture of our solution is presented in Figure 1. Based on the service configuration, the SAIN orchestrator deduces the assurance graph. It then sends to the SAIN agents the assurance graph along some other configuration options. The SAIN agents are responsible for building the expression graph and computing the health statuses in a distributed manner. The collector is in charge of collecting and displaying the current inferred health status of the service instances and subservices. Finally, the automation loop is closed by having the SAIN Collector providing feedback to the network orchestrator.

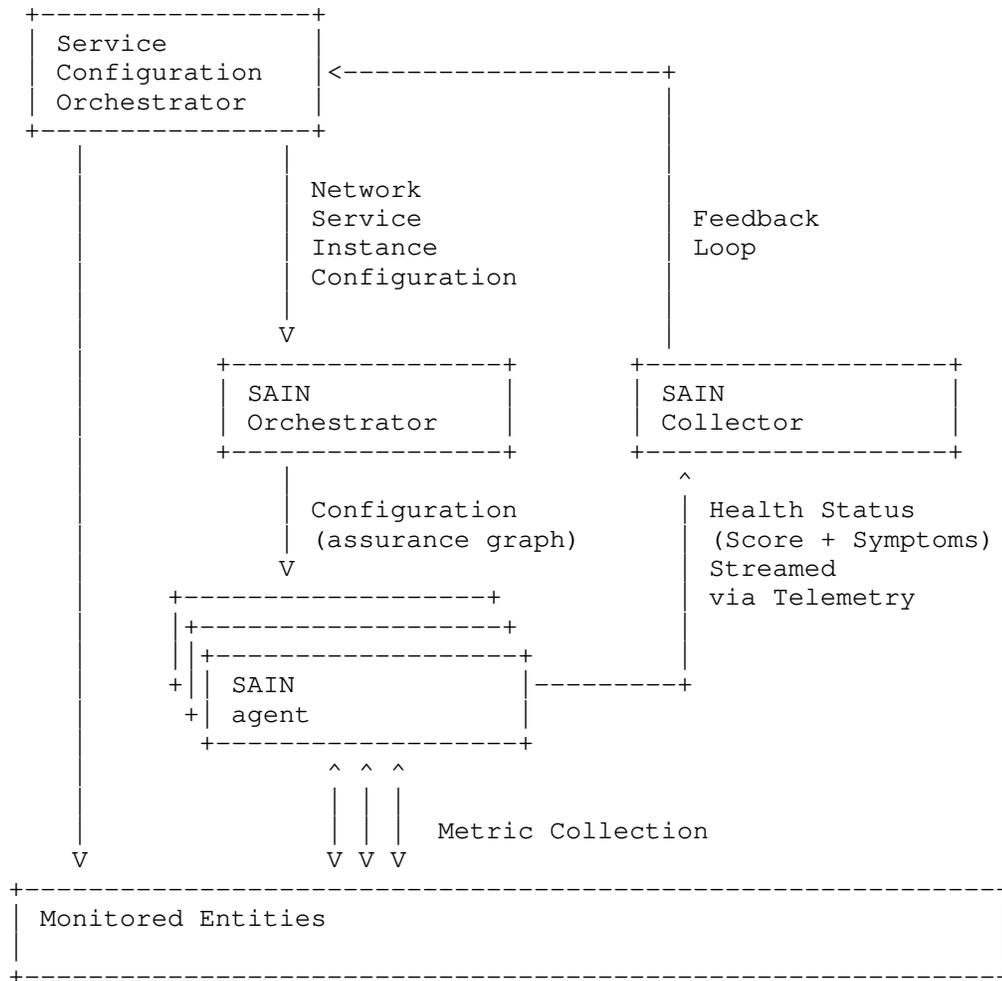


Figure 1: SAIN Architecture

In order to produce the score assigned to a service instance, the architecture performs the following tasks:

- o Analyze the configuration pushed to the network device(s) for configuring the service instance and decide: which information is needed from the device(s), such a piece of information being called a metric, which operations to apply to the metrics for computing the health status.

- o Stream (via telemetry [RFC8641]) operational and config metric values when possible, else continuously poll.
- o Continuously compute the health status of the service instances, based on the metric values.

3.1. Decomposing a Service Instance Configuration into an Assurance Graph

In order to structure the assurance of a service instance, the service instance is decomposed into so-called subservice instances. Each subservice instance focuses on a specific feature or subpart of the network system.

The decomposition into subservices is an important function of this architecture, for the following reasons.

- o The result of this decomposition provides a relational picture of a service instance, that can be represented as a graph (called assurance graph) to the operator.
- o Subservices provide a scope for particular expertise and thereby enable contribution from external experts. For instance, the subservice dealing with the optics health should be reviewed and extended by an expert in optical interfaces.
- o Subservices that are common to several service instances are reused for reducing the amount of computation needed.

The assurance graph of a service instance is a DAG representing the structure of the assurance case for the service instance. The nodes of this graph are service instances or subservice instances. Each edge of this graph indicates a dependency between the two nodes at its extremities: the service or subservice at the source of the edge depends on the service or subservice at the destination of the edge.

Figure 2 depicts a simplistic example of the assurance graph for a tunnel service. The node at the top is the service instance, the nodes below are its dependencies. In the example, the tunnel service instance depends on the peer1 and peer2 tunnel interfaces, which in turn depend on the respective physical interfaces, which finally depend on the respective peer1 and peer2 devices. The tunnel service instance also depends on the IP connectivity that depends on the IS-IS routing protocol.

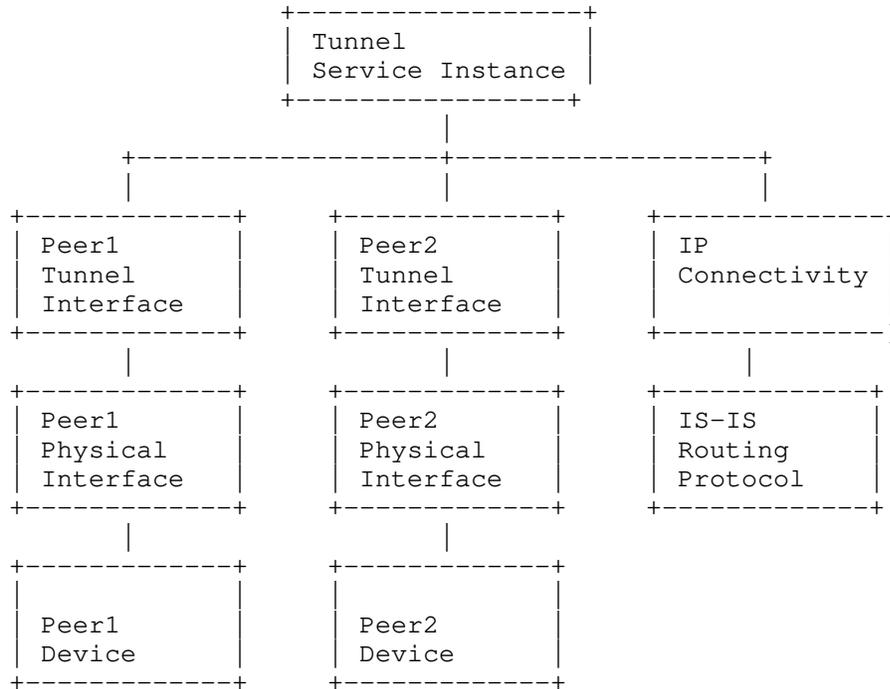


Figure 2: Assurance Graph Example

Depicting the assurance graph helps the operator to understand (and assert) the decomposition. The assurance graph shall be maintained during normal operation with addition, modification and removal of service instances. A change in the network configuration or topology shall be reflected in the assurance graph. As a first example, a change of routing protocol from IS-IS to OSPF would change the assurance graph accordingly. As a second example, assuming that ECMP is in place for the source router for that specific tunnel; in that case, multiple interfaces must now be monitored, on top of the monitoring the ECMP health itself.

3.2. Intent and Assurance Graph

The SAIN orchestrator analyzes the configuration of a service instance to:

- o Try to capture the intent of the service instance, i.e. what is the service instance trying to achieve,
- o Decompose the service instance into subservices representing the network features on which the service instance relies.

The SAIN orchestrator must be able to analyze configuration from various devices and produce the assurance graph.

To schematize what a SAIN orchestrator does, assume that the configuration for a service instance touches 2 devices and configure on each device a virtual tunnel interface. Then:

- o Capturing the intent would start by detecting that the service instance is actually a tunnel between the two devices, and stating that this tunnel must be functional. This is the current state of SAIN, however it does not completely capture the intent which might additionally include, for instance, on the latency and bandwidth requirements of this tunnel.
- o Decomposing the service instance into subservices would result in the assurance graph depicted in Figure 2, for instance.

In order for SAIN to be applied, the configuration necessary for each service instance should be identifiable and thus should come from a "service-aware" source. While the Figure 1 makes a distinction between the SAIN orchestrator and a different component providing the service instance configuration, in practice those two components are mostly likely combined. The internals of the orchestrator are currently out of scope of this document.

3.3. Subservices

A subservice corresponds to subpart or a feature of the network system that is needed for a service instance to function properly. In the context of SAIN, subservice is actually a shortcut for subservice assurance, that is the method for assuring that a subservice behaves correctly.

Subservices, just as with services, have high-level parameters that specify the type and specific instance to be assured. For example, assuring a device requires the specific deviceId as parameter. For example, assuring an interface requires the specific combination of deviceId and interfaceId.

A subservice is also characterized by a list of metrics to fetch and a list of computations to apply to these metrics in order to infer a health status.

3.4. Building the Expression Graph from the Assurance Graph

From the assurance graph is derived a so-called global computation graph. First, each subservice instance is transformed into a set of subservice expressions that take metrics and constants as input (i.e.

sources of the DAG) and produce the status of the subservice, based on some heuristics. Then for each service instance, the service expressions are constructed by combining the subservice expressions of its dependencies. The way service expressions are combined depends on the dependency types (impacting or informational). Finally, the global computation graph is built by combining the service expressions. In other words, the global computation graph encodes all the operations needed to produce health statuses from the collected metrics.

Subservices shall be device independent. To justify this, let's consider the interface operational status. Depending on the device capabilities, this status can be collected by an industry-accepted YANG module (IETF, Openconfig), by a vendor-specific YANG module, or even by a MIB module. If the subservice was dependent on the mechanism to collect the operational status, then we would need multiple subservice definitions in order to support all different mechanisms. This also implies that, while waiting for all the metrics to be available via standard YANG modules, SAIN agents might have to retrieve metric values via non-standard YANG models, via MIB modules, Command Line Interface (CLI), etc., effectively implementing a normalization layer between data models and information models.

In order to keep subservices independent from metric collection method, or, expressed differently, to support multiple combinations of platforms, OSes, and even vendors, the framework introduces the concept of "metric engine". The metric engine maps each device-independent metric used in the subservices to a list of device-specific metric implementations that precisely define how to fetch values for that metric. The mapping is parameterized by the characteristics (model, OS version, etc.) of the device from which the metrics are fetched.

3.5. Building the Expression from a Subservice

Additionally, to the list of metrics, each subservice defines a list of expressions to apply on the metrics in order to compute the health status of the subservice. The definition or the standardization of those expressions (also known as heuristic) is currently out of scope of this standardization.

3.6. Open Interfaces with YANG Modules

The interfaces between the architecture components are open thanks to the YANG modules specified in YANG Modules for Service Assurance [I-D.claise-opsawg-service-assurance-yang]; they specify objects for assuring network services based on their decomposition into so-called subservices, according to the SAIN architecture.

This module is intended for the following use cases:

- o Assurance graph configuration:
 - * Subservices: configure a set of subservices to assure, by specifying their types and parameters.
 - * Dependencies: configure the dependencies between the subservices, along with their types.
- o Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

3.7. Handling Maintenance Windows

Whenever network components are under maintenance, the operator want to inhibit the emission of symptoms from those components. A typical use case is device maintenance, during which the device is not supposed to be operational. As such, symptoms related to the device health should be ignored, as well as symptoms related to the device-specific subservices, such as the interfaces, as their state changes is probably the consequence of the maintenance.

To configure network components as "under maintenance" in the SAIN architecture, the ietf-service-assurance model proposed in [I-D.claise-opsawg-service-assurance-yang] specifies an "under-maintenance" flag per service or subservice instance. When this flag is set and only when this flag is set, the companion field "maintenance-contact" must be set to a string that identifies the person or process who requested the maintenance. Any symptom produced by a service or subservice under maintenance, or by one of its dependencies MUST NOT be reported. A service or subservice under maintenance MAY propagate a symptom "Under Maintenance" towards services or subservices that depend on it.

We illustrate this mechanism on three independent examples based on the assurance graph depicted in Figure 2:

- o Device maintenance, for instance upgrading the device OS. The operator sets the "under-maintenance" flag for the subservice "Peer1" device. This inhibits the emission of symptoms from "Peer1 Physical Interface", "Peer1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.
- o Interface maintenance, for instance replacing a broken optic. The operator sets the "under-maintenance" flag for the subservice "Peer1 Physical Interface". This inhibits the emission of

symptoms from "Peer 1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.

- o Routing protocol maintenance, for instance modifying parameters or redistribution. The operator sets the "under-maintenance" flag for the subservice "IS-IS Routing Protocol". This inhibits the emission of symptoms from "IP connectivity" and "Tunnel Service Instance". All other subservices are unaffected.

3.8. Flexible Architecture

The SAIN architecture is flexible in terms of components. While the SAIN architecture in Figure 1 makes a distinction between two components, the SAIN configuration orchestrator and the SAIN orchestrator, in practice those two components are mostly likely combined. Similarly, the SAIN agents are displayed in Figure 1 as being separate components. Practically, the SAIN agents could be either independent components or directly integrated in monitored entities. A practical example is an agent in a router.

The SAIN architecture is also flexible in terms of services and subservices. Most examples in this document deal with the notion of Network Service YANG modules, with well known service such as L2VPN or tunnels. However, the concepts of services is general enough to cross into different domains. One of them is the domain of service management on network elements, with also requires its own assurance. Examples includes a DHCP server on a linux server, a data plane, an IPFIX export, etc. The notion of "service" is generic in this architecture. Indeed, a configured service can itself be a service for someone else. Exactly like an DHCP server/ data plane/IPFIX export can be considered as services for a device, exactly like an routing instance can be considered as a service for a L3VPN, exactly like a tunnel can considered as a service for an application in the cloud. The assurance graph is created to be flexible and open, regardless of the subservice types, locations, or domains.

The SAIN architecture is also flexible in terms of distributed graphs. As shown in Figure 1, our architecture comprises several agents. Each agent is responsible for handling a subgraph of the assurance graph. The collector is responsible for fetching the subgraphs from the different agents and gluing them together. As an example, in the graph from Figure 2, the subservices relative to Peer 1 might be handled by a different agent than the subservices relative to Peer 2 and the Connectivity and IS-IS subservices might be handled by yet another agent. The agents will export their partial graph and the collector will stitch them together as dependencies of the service instance.

And finally, the SAIN architecture is flexible in terms of what it monitors. Most, if not all examples, in this document refer to physical components but this is not a constrain. Indeed, the assurance of virtual components would follow the same principles and an assurance graph composed of virtualized components (or a mix of virtualized and physical ones) is well possible within this architecture.

3.9. Timing

The SAIN architecture requires the Network Time Protocol (NTP) [RFC5905] between all elements: monitored entities, SAIN agents, Service Configuration Orchestttrator, the SAIN Collector, as well as the SAIN Orchestrator. This garantees the correlations of all symptoms in the system, correlated with the right assurance graph version.

The SAIN agent might have to remove some symptoms for specific subservice symptoms, because there are outdated and not relevant any longer, or simply because the SAIN agent needs to free up some space. Regardless of the reason, it's important for a SAIN collector (re-)connecting to a SAIN agent to understand the effect of this garbage collection. Therefore, the SAIN agent contains a YANG object specifying the date and time at which the symptoms history starts for the subservice instances.

3.10. New Assurance Graph Generation

The assurance graph will change along the time, because services and subservices come and go (changing the dependencies between subservices), or simply because a subservice is now under maintenance. Therefore an assurance graph version must be maintained, along with the date and time of its last generation. The date and time of a particular subservice instance (again dependencies or under maintenane) might be kept. From a client point of view, an assurance graph change is triggered by the value of the assurance-graph-version and assurance-graph-last-change YANG leaves. At that point in time, the client (collector) follows the following process:

- o Keep the previous assurance-graph-last-change value (let's call it time T)
- o Run through all subservice instance and process the subservice instances for which the last-change is newer that the time T
- o Keep the new assurance-graph-last-change as the new referenced date and time

4. Security Considerations

The SAIN architecture helps operators to reduce the mean time to detect and mean time to repair. As such, it should not cause any security threats. However, the SAIN agents must be secure: a compromised SAIN agents could be sending wrong root causes or symptoms to the management systems.

Except for the configuration of telemetry, the agents do not need "write access" to the devices they monitor. This configuration is applied with a YANG module, whose protection is covered by Secure Shell (SSH) [RFC6242] for NETCONF or TLS [RFC8446] for RESTCONF.

The data collected by SAIN could potentially be compromising to the network or provide more insight into how the network is designed. Considering the data that SAIN requires (including CLI access in some cases), one should weigh data access concerns with the impact that reduced visibility will have on being able to rapidly identify root causes.

If a closed loop system relies on this architecture then the well known issue of those system also applies, i.e., a lying device or compromised agent could trigger partial reconfiguration of the service or network. The SAIN architecture neither augments or reduces this risk.

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

- o Youssef El Fathi
- o Eric Vyncke

7. Open Issues

Refer to the Intent-based Networking NMRG documents

8. References

8.1. Normative References

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

- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [I-D.claise-opsawg-service-assurance-yang] Claise, B. and J. Quilbeuf, "Service Assurance for Intent-based Networking Architecture", February 2020.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.
- [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>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [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>>.

- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [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>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8907] Dahm, T., Ota, A., Medway Gash, D., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <<https://www.rfc-editor.org/info/rfc8907>>.

Appendix A. Changes between revisions

v02 - v03

- o Timing Concepts
- o New Assurance Graph Generation

v01 - v02

- o Handling maintenance windows
- o Flexible architecture better explained
- o Improved the terminology
- o Notion of mapping information model to data model, while waiting for YANG to be everywhere
- o Started a security considerations section

v00 - v01

- o Terminology clarifications
- o Figure 1 improved

Acknowledgements

The authors would like to thank Stephane Litkowski, Charles Eckel, Rob Wilton, Vladimir Vassiliev, Gustavo Albuquerque, Stefan Vallin, and Eric Vyncke for their reviews and feedback.

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Jean Quilbeuf
Independent

Email: jean@quilbeuf.net

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

Dan Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Thangam Arumugam
Cisco Systems, Inc.
Milpitas (California)
United States of America

Email: tarumuga@cisco.com

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: October 25, 2021

B. Claise
Huawei
J. Quilbeuf
Independent
P. Lucente
NTT
P. Fasano
TIM S.p.A
T. Arumugam
Cisco Systems, Inc.
April 23, 2021

YANG Modules for Service Assurance
draft-claise-opsawg-service-assurance-yang-07

Abstract

This document proposes YANG modules for the Service Assurance for Intent-based Networking Architecture.

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 October 25, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Introduction	3
3. YANG Models Overview	3
4. Base ietf-service-assurance YANG module	4
4.1. Tree View	4
4.2. Concepts	5
4.3. YANG Module	6
5. Subservice Extension: ietf-service-assurance-device YANG module	13
5.1. Tree View	13
5.2. Complete Tree View	13
5.3. Concepts	14
5.4. YANG Module	15
6. Subservice Extension: ietf-service-assurance-interface YANG module	16
6.1. Tree View	16
6.2. Complete Tree View	17
6.3. Concepts	18
6.4. YANG Module	18
7. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module	19
7.1. Tree View	19
7.2. Complete Tree View	20
7.3. Concepts	21
7.4. YANG Module	22
8. Security Considerations	23
9. IANA Considerations	24
9.1. The IETF XML Registry	24
9.2. The YANG Module Names Registry	25
10. Open Issues	25
11. References	25
11.1. Normative References	25
11.2. Informative References	26
Appendix A. Changes between revisions	26
Acknowledgements	27
Authors' Addresses	27

1. Terminology

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

The terms used in this document are defined in draft-claise-opsawg-service-assurance-architecture IETF draft.

2. Introduction

The "Service Assurance for Intent-based Networking Architecture" draft-claise-opsawg-service-assurance-architecture, specifies the framework and all of its components for service assurance. This document complements the architecture by providing open interfaces between components. More specifically, the goal is to provide YANG modules for the purpose of service assurance in a format that is:

- o machine readable
- o vendor independent
- o augmentable

3. YANG Models Overview

The main YANG module, `ietf-service-assurance`, defines objects for assuring network services based on their decomposition into so-called subservices. The subservices are hierarchically organised by dependencies. The subservices, along with the dependencies, constitute an assurance graph. This module should be supported by an agent, able to interact with the devices in order to produce a health status and symptoms for each subservice in the assurance graph. This module is intended for the following use cases:

- o Assurance graph configuration:
 - * Subservices: configure a set of subservices to assure, by specifying their types and parameters.
 - * Dependencies: configure the dependencies between the subservices, along with their type.
- o Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

The second YANG module, `ietf-service-assurance-device`, extends the `ietf-service-assurance` module to add support for the subservice `DeviceHealthy`. Additional subservice types might be added the same way.

The third YANG module, `example-service-assurance-device-acme`, extends the `ietf-service-assurance-device` module as an example to add support for the subservice `DeviceHealthy`, with specifics for the fictional ACME Corporation. Additional vendor-specific parameters might be added the same way.

4. Base `ietf-service-assurance` YANG module

4.1. Tree View

The following tree diagram [RFC8340] provides an overview of the `ietf-service-assurance` data model.

```

module: ietf-service-assurance
+--ro assurance-graph-version          yang:counter32
+--ro assurance-graph-last-change      yang:date-and-time
+--rw subservices
  +--rw subservice* [type id]
    +--rw type                          identityref
    +--rw id                             string
    +--ro last-change?                  yang:date-and-time
    +--ro label?                        string
    +--rw under-maintenance?            boolean
    +--rw maintenance-contact           string
    +--rw (parameter)?
      +--:(service-instance-parameter)
        +--rw service-instance-parameter
          +--rw service?                string
          +--rw instance-name?          string
    +--ro health-score?                  uint8
    +--ro symptoms-history-start?        yang:date-and-time
    +--rw symptoms
      +--ro symptom* [start-date-time id]
        +--ro id                        string
        +--ro health-score-weight?      uint8
        +--ro description?              string
        +--ro start-date-time           yang:date-and-time
        +--ro stop-date-time?           yang:date-and-time
    +--rw dependencies
      +--rw dependency* [type id]
        +--rw type                      -> /subservices/subservice/type
        +--rw id                        -> /subservices/subservice[type=current()/
../type]/id
        +--rw dependency-type?          identityref

```

4.2. Concepts

The ietf-service-assurance YANG model assumes an identified number of subservices, to be assured independently. A subservice is a feature or a subpart of the network system that a given service instance might depend on. Example of subservices include:

- o DeviceHealthy: whether a device is healthy, and if not, what are the symptoms. Potential symptoms are "CPU overloaded", "Out of RAM", or "Out of TCAM".
- o ConnectivityHealthy: given two IP addresses owned by two devices, what is the quality of the connection between them. Potential symptoms are "No route available" or "ECMP Imbalance".

The first example is a subservice representing a subpart of the network system, while the second is a subservice representing a feature of the network. In both cases, these subservices might depend on other subservices, for instance, the connectivity might depend on a subservice representing the routing mechanism and on a subservice representing ECMP.

The symptoms are listed for each subservice. Each symptom is specified by a unique id and contains a health-score-weight (the impact to the health score incurred by this symptom), a label (text describing what the symptom is), and dates and times at which the symptom was detected and stopped being detected. While the unique id is sufficient as an unique key list, the start-date-time second key help sorting and retrieving relevant symptoms.

The assurance of a given service instance can be obtained by composing the assurance of the subservices that it depends on, via the dependency relations.

In order to declare a subservice MUST provide:

- o A type: identity inheriting of the base identity for subservice,
- o An id: string uniquely identifying the subservice among those with the same identity,
- o Some parameters, which should be specified in an augmenting model, as described in the next sections.

The type and id uniquely identify a given subservice. They are used to indicate the dependencies. Dependencies have types as well. Two types are specified in the model:

- o **Impacting:** such a dependency indicates an impact on the health of the dependent,
- o **Informational:** such a dependency might explain why the dependent has issues but does not impact its health.

To illustrate the difference between "impacting" and "informational", consider the subservice `InterfaceHealthy`, representing a network interface. If the device to which the network interface belongs goes down, the network interface will transition to a down state as well. Therefore, the dependency of `InterfaceHealthy` towards `DeviceHealthy` is "impacting". On the other hand, as a the dependency towards the `ECMPLoad` subservice, which checks that the load between ECMP remains stable throughout time, is only "informational". Indeed, services might be perfectly healthy even if the load distribution between ECMP changed. However, such an instability might be a relevant symptom for diagnosing the root cause of a problem.

Service instances **MUST** be modeled as a particular type of subservice with two parameters, a type and an instance name. The type is the name of the service defined in the network orchestrator, for instance "point-to-point-l2vpn". The instance name is the name assigned to the particular instance that we are assuring, for instance the name of the customer using that instance.

The "under-maintenance" and "maintenance-contact" flags inhibit the emission of symptoms for that subservice and subservices that depend on them. See Section 3.7 of [draft-claise-opsawg-service-assurance-architecture] for a more detailed discussion.

By specifying service instances and their dependencies in terms of subservices, one defines the whole assurance to apply for them. An assurance agent supporting this model should then produce telemetry in return with, for each subservice: a health-status indicating how healthy the subservice is and when the subservice is not healthy, a list of symptoms explaining why the subservice is not healthy.

4.3. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance@2020-01-13.yang"

module ietf-service-assurance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance";
  prefix service-assurance;

  import ietf-yang-types {
```

```
    prefix yang;
  }
```

```
organization
```

```
  "IETF NETCONF (Network Configuration) Working Group";
```

```
contact
```

```
  "WG Web: <https://datatracker.ietf.org/wg/netconf/>
```

```
  WG List: <mailto:netconf@ietf.org>
```

```
  Author: Benoit Claise <mailto:bclaise@cisco.com>
```

```
  Author: Jean Quilbeuf <mailto:jquilbeu@cisco.com>";
```

```
description
```

```
"This module defines objects for assuring network services based on their decomposition into so-called subservices, according to the SAIN (Service Assurance for Intent-based Networking) architecture.
```

The subservices hierarchically organised by dependencies constitute an assurance graph. This module should be supported by an assurance agent, able to interact with the devices in order to produce a health status and symptoms for each subservice in the assurance graph.

This module is intended for the following use cases:

- * Assurance graph configuration:
 - * subservices: configure a set of subservices to assure, by specifying their types and parameters.
 - * dependencies: configure the dependencies between the subservices, along with their type.
- * Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

Copyright (c)2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

TO DO:

- Better type (IETF or OC) for device-id, interface-id, etc.
- Have a YANG module for IETF and one for OC?";

```
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: Title to be completed";
}

revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: Title to be completed";
}

identity subservice-idty {
  description
    "Root identity for all subservice types.";
}

identity service-instance-idty {
  base subservice-idty;
  description
    "Identity representing a service instance.";
}

identity dependency-type {
  description
    "Base identity for representing dependency types.";
}

identity informational-dependency {
  base dependency-type;
  description
    "Indicates that symptoms of the dependency might be of interest for the
    dependent, but the status of the dependency should not have any
    impact on the dependent.";
}

identity impacting-dependency {
  base dependency-type;
  description
    "Indicates that the status of the dependency directly impacts the status
    of the dependent.";
}
```

```
grouping symptom {
  description
    "Contains the list of symptoms for a specific subservice.";
  leaf id {
    type string;
    description
      "A unique identifier for the symptom.";
  }
  leaf health-score-weight {
    type uint8 {
      range "0 .. 100";
    }
    description
      "The weight to the health score incurred by this symptom. The higher the
      value, the more of an impact this symptom has. If a subservice health
      score is not 100, there must be at least one symptom with a health
      score weight larger than 0.";
  }
  leaf description {
    type string;
    description
      "Description of the symptom, i.e. text describing what the symptom is, to
      be computer-consumable and be displayed on a human interface. ";
  }
  leaf start-date-time {
    type yang:date-and-time;
    description
      "Date and time at which the symptom was detected.";
  }
  leaf stop-date-time {
    type yang:date-and-time;
    description
      "Date and time at which the symptom stopped being detected.";
  }
}

grouping subservice-dependency {
  description
    "Represent a dependency to another subservice.";
  leaf type {
    type leafref {
      path "/subservices/subservice/type";
    }
    description
      "The type of the subservice to refer to (e.g. DeviceHealthy).";
  }
  leaf id {
    type leafref {
```

```
    path "/subservices/subservice[type=current()/../type]/id";
  }
  description
    "The identifier of the subservice to refer to.";
}
leaf dependency-type {
  type identityref {
    base dependency-type;
  }
  description
    "Represents the type of dependency (i.e. informational, impacting).";
}
// augment here if more info are needed (i.e. a percentage) depending on the
// dependency type.
}

leaf assurance-graph-version {
  type yang:counter32;
  mandatory true;
  config false;
  description
    "The assurance graph version, which increases by 1 for each new version, af
    ter the changes
    (dependencies and/or maintenance windows parameters) are applied to the su
    bservice(s).";
}
leaf assurance-graph-last-change {
  type yang:date-and-time;
  mandatory true;
  config false;
  description
    "Date and time at which the assurance graph last changed after the changes
    (dependencies
    and/or maintenance windows parameters) are applied to the subservice(s). T
    hese date and time
    must be more recent or equal compared to the more recent value of any chan
    ged subservices
    last-change";
}
container subservices {
  description
    "Root container for the subservices.";
  list subservice {
    key "type id";
    description
      "List of subservice configured.";
    leaf type {
      type identityref {
        base subservice-idty;
      }
      description
        "Name of the subservice, e.g. DeviceHealthy.";
    }
    leaf id {
```



```
    type string;
    description
      "Unique identifier of the subservice instance, for each type.";
  }
  leaf last-change {
    type yang:date-and-time;
    config false;
    description
      "Date and time at which the assurance graph for this subservice
      instance last changed, i.e. dependencies and/or maintenance windows pa
rameters.";
  }
  leaf label {
    type string;
    config false;
    description
      "Label of the subservice, i.e. text describing what the subservice is t
o
      be displayed on a human interface.";
  }
  leaf under-maintenance {
    type boolean;
    default false;
    description
      "An optional flag indicating whether this particular subservice is unde
r
      maintenance. Under this circumstance, the subservice symptoms and the
      symptoms of its dependencies in the assurance graph should not be taken
      into account. Instead, the subservice should send a 'Under Maintenance'
      single symptom.

      The operator changing the under-maintenance value must set the
      maintenance-contact variable.

      When the subservice is not under maintenance any longer, the
      under-maintenance flag must return to its default value and
      the under-maintenance-owner variable deleted.";
  }
  leaf maintenance-contact {
    when "../under-maintenance = 'true'";
    type string;
    mandatory true;
    description
      "A string used to model an administratively assigned name of the
      resource that changed the under-maintenance value to 'true.

      It is suggested that this name contain one or more of the following:
      IP address, management station name, network manager's name, location,
      or phone number. In some cases the agent itself will be the owner of
      an entry. In these cases, this string shall be set to a string
      starting with 'monitor'.";
```

```
    }
    choice parameter {
      description
        "Specify the required parameters per subservice type.";
      container service-instance-parameter {
        when "derived-from-or-self(..../type, 'service-assurance:service-instance
-idty')";
        description
          "Specify the parameters of a service instance.";
        leaf service {
          type string;
          description "Name of the service.";
        }
        leaf instance-name{
          type string;
          description "Name of the instance for that service.";
        }
      }
      // Other modules can augment their own cases into here
    }
    leaf health-score {
      type uint8 {
        range "0 .. 100";
      }
      config false;
      description
        "Score value of the subservice health. A value of 100 means that
        subservice is healthy. A value of 0 means that the subservice is
        broken. A value between 0 and 100 means that the subservice is
        degraded.";
    }
    leaf symptoms-history-start {
      type yang:date-and-time;
      config false;
      description
        "Date and time at which the symptoms history starts for this
        subservice instance, either because the subservice instance
        started at that date and time or because the symptoms before that
        were removed due to a garbage collection process.";
    }
    container symptoms {
      description
        "Symptoms for the subservice.";
      list symptom {
        key "start-date-time id";
        config false;
        description
          "List of symptoms the subservice. While the start-date-time key is no
t
          necessary per se, this would get the entries sorted by start-date-tim
e
```



```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter32
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?                  yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact          string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service?                string
            +--rw instance-name?         string
        +--:(service-assurance-device:device-idty)
          +--rw service-assurance-device:device-idty
            +--rw service-assurance-device:device? string
      +--ro health-score?                  uint8
      +--ro symptoms-history-start?        yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                        string
          +--ro health-score-weight?      uint8
          +--ro description?              string
          +--ro start-date-time           yang:date-and-time
          +--ro stop-date-time?           yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type                      -> /subservices/subservice/type
          +--rw id                        -> /subservices/subservice[type=current()/
../type]/id
          +--rw dependency-type?          identityref

```

5.3. Concepts

As the number of subservices will grow over time, the YANG module is designed to be extensible. A new subservice type requires the precise specifications of its type and expected parameters. Let us illustrate the example of the new DeviceHealthy subservice type. As the name implies, it monitors and reports the device health, along with some symptoms in case of degradation.

For our DeviceHealthy subservice definition, the new device-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of a device.

The typical parameter for the configuration of the DeviceHealthy subservice is the name of the device that we want to assure. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the device-idty subservice type, this new parameter is specified.

5.4. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance-device@2020-01-13.yang"

module ietf-service-assurance-device {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device";
  prefix service-assurance-device;

  import ietf-service-assurance {
    prefix "service-assurance";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/netconf/>
    WG List: <mailto:netconf@ietf.org>
    Author: Benoit Claise <mailto:bclaise@cisco.com>
    Author: Jean Quilbeuf <mailto:jquilbeuf@cisco.com>";
  description
    "This module extends the ietf-service-assurance module to add
    support for the subservice DeviceHealthy.

    Checks whether a network device is healthy.

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.

    Copyright (c) 2020 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

```
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: Title to be completed";
}

revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: Title to be completed";
}

identity device-idty {
  base service-assurance:subservice-idty;
  description "Network Device is healthy.";
}

augment /service-assurance:subservices/service-assurance:subservice/service-assurance:parameter {
  description
    "Specify the required parameters for a new subservice type";
  container device-idty{
    when "derived-from-or-self(..service-assurance:type, 'device-idty')";
    description
      "Specify the required parameters for the device-idty subservice type";
  }

  leaf device {
    type string;
    description "The device to monitor.";
  }
}
}

<CODE ENDS>
```

6. Subservice Extension: ietf-service-assurance-interface YANG module

6.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance-interface data model.

```

module: ietf-service-assurance-interface
  augment /service-assurance:subservices/service-assurance:parameter:
    +--rw device?      string
    +--rw interface?  string

```

6.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the `ietf-service-assurance`, `ietf-service-assurance-device`, and `ietf-service-assurance-interface` data models.

```

module: ietf-service-assurance
  +--ro assurance-graph-version      yang:counter32
  +--ro assurance-graph-last-change  yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                    identityref
      +--rw id                      string
      +--ro last-change?            yang:date-and-time
      +--ro label?                  string
      +--rw under-maintenance?      boolean
      +--rw maintenance-contact     string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service?          string
            +--rw instance-name?   string
        +--:(service-assurance-device:device-idty)
          +--rw service-assurance-device:device-idty
            +--rw service-assurance-device:device? string
        +--:(service-assurance-interface:device)
          +--rw service-assurance-interface:device? string
        +--:(service-assurance-interface:interface)
          +--rw service-assurance-interface:interface? string
      +--ro health-score?            uint8
      +--ro symptoms-history-start?  yang:date-and-time
      +--rw symptoms
        +--rw symptom* [start-date-time id]
          +--ro id                  string
          +--ro health-score-weight? uint8
          +--ro description?        string
          +--ro start-date-time     yang:date-and-time
          +--ro stop-date-time?     yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type                -> /subservices/subservice/type
          +--rw id                  -> /subservices/subservice[type=current()/
.. /type]/id
          +--rw dependency-type?    identityref

```

6.3. Concepts

For our InterfaceHealthy subservice definition, the new interface-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of an interface.

The typical parameters for the configuration of the InterfaceHealthy subservice are the name of the device and, on that specific device, a specific interface. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the interface-idty subservice type, those two new parameter are specified.

6.4. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance-interface@2020-01-13.yang"

module ietf-service-assurance-interface {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface";
  prefix service-assurance-interface;

  import ietf-service-assurance {
    prefix "service-assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
    WG List: <mailto:opsawg@ietf.org>
    Author: Benoit Claise <mailto:bclaise@cisco.com>
    Author: Jean Quilbeuf <mailto:jquilbeu@cisco.com>";
  description
    "This module extends the ietf-service-assurance module to add
    support for the subservice InterfaceHealthy.

    Checks whether an interface is healthy.

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.

    Copyright (c) 2020 IETF Trust and the persons identified as
    authors of the code. All rights reserved.
```

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

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

```
revision 2020-01-13 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: Title to be completed";
}

identity interface-idty {
  base service-assurance:subservice-idty;
  description "Checks whether an interface is healthy.";
}

augment /service-assurance:subservices/service-assurance:subservice/service-assurance:parameter {
  when "derived-from-or-self(service-assurance:type, 'interface-idty')";
  description
    "Specify the required parameters for the interface-idty subservice type"
;

  leaf device {
    type string;
    description "Device supporting the interface.";
  }
  leaf interface {
    type string;
    description "Name of the interface.";
  }
}
}
```

<CODE ENDS>

7. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module

7.1. Tree View

The following tree diagram [RFC8340] provides an overview of the example-service-assurance-device-acme data model.

```
module: example-service-assurance-device-acme
  augment /service-assurance:subservices/service-assurance:subservice/service-assurance:parameter:
    +--rw acme-device-idty
      +--rw device?                string
      +--rw acme-specific-parameter? string
```

7.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the `ietf-service-assurance`, `ietf-service-assurance-device`, and `example-service-assurance-device-acme` data models.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter32
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          ide
  entityref
    +--rw id                              str
  ing
    +--ro last-change?                   yan
  g:date-and-time
    +--ro label?                         str
  ing
    +--rw under-maintenance?             boo
  lean
    +--rw maintenance-contact            str
  ing
    +--rw (parameter)?
      +--:(service-instance-parameter)
        +--rw service-instance-parameter
          +--rw service?                  string
          +--rw instance-name?           string
      +--:(service-assurance-device:device-idty)
        +--rw service-assurance-device:device-idty
          +--rw service-assurance-device:device? string
      +--:(service-assurance-interface:device)
        +--rw service-assurance-interface:device? str
  ing
      +--:(service-assurance-interface:interface)
        +--rw service-assurance-interface:interface? str
  ing
      +--:(example-service-assurance-device-acme:acme-device-idty)
        +--rw example-service-assurance-device-acme:acme-device-idty
          +--rw example-service-assurance-device-acme:device?
  string
      +--rw example-service-assurance-device-acme:acme-specific-parame
  ter? string
      +--ro health-score?                 uin
  t8
    +--ro symptoms-history-start?        yan
  g:date-and-time
    +--rw symptoms
      +--ro symptom* [start-date-time id]
        +--ro id                          string
        +--ro health-score-weight?        uint8
        +--ro description?                 string
        +--ro start-date-time             yang:date-and-time
        +--ro stop-date-time?             yang:date-and-time
    +--rw dependencies
      +--rw dependency* [type id]
        +--rw type                        -> /subservices/subservice/type
        +--rw id                          -> /subservices/subservice[type=current()/
  ../type]/id
      +--rw dependency-type?             identityref

```

7.3. Concepts

Under some circumstances, vendor-specific subservice types might be

required. As an example of this vendor-specific implementation, this section shows how to augment the ietf-service-assurance-device module

to add support for the subservice DeviceHealthy, specific to the ACME Corporation. The new parameter is acme-specific-parameter.

7.4. YANG Module

```
module example-service-assurance-device-acme {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-device-acme";
  prefix example-service-assurance-device-acme;

  import ietf-service-assurance {
    prefix "service-assurance";
  }

  import ietf-service-assurance-device {
    prefix "service-assurance-device";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/netconf/>
    WG List: <mailto:netconf@ietf.org>
    Author: Benoit Claise <mailto:bclaise@cisco.com>
    Author: Jean Quilbeuf <mailto:jquilbeuf@cisco.com>";
  description
    "This module extends the ietf-service-assurance-device module to add
    support for the subservice DeviceHealthy, specific to the ACME Corporation.

    ACME Network Device is healthy.

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.

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

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

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

```
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: Title to be completed";
}

revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: Title to be completed";
}

identity device-acme-idty {
  base service-assurance-device:device-idty;
  description "Network Device is healthy.";
}

augment /service-assurance:subservices/service-assurance:subservice/service-assurance:parameter {
  description
    "Specify the required parameters for a new subservice type";
  container acme-device-idty{
    when "derived-from-or-self(..service-assurance:type, 'device-acme-idty')";
    description
      "Specify the required parameters for the device-acme-idty subservice type";

    leaf device {
      type string;
      description "The device to monitor.";
    }

    leaf acme-specific-parameter {
      type string;
      description "The ACME Corporation sepcific parameter.";
    }
  }
}
}
```

8. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer

is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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

- o /subservices/subservice/type
- o /subservices/subservice/id
- o /subservices/subservice/under-maintenance
- o /subservices/subservice/maintenance-contact

9. IANA Considerations

9.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

9.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

```
name:      ietf-service-assurance
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance
prefix:    inc
reference:  RFC XXXX

name:      ietf-service-assurance-device
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device
prefix:    inc
reference:  RFC XXXX

name:      ietf-service-assurance-interface
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface
prefix:    inc
reference:  RFC XXXX
```

10. Open Issues

-None

11. References

11.1. Normative References

- [draft-claise-opsawg-service-assurance-architecture]
Claise, B. and J. Quilbeuf, "draft-claise-opsawg-service-assurance-architecture", 2020.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

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

11.2. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.

Appendix A. Changes between revisions

v04 - v05

- o Added the concept of symptoms-history-start
- o Changed label to description, under symptoms. This was confusing as there was two labels in the models

v03 - v04

- o Add the interface subservice, with two parameters

v02 - v03

- o Added the maintenace window concepts

v01 - v02

- o Improved leaf naming
- o Clarified some concepts: symptoms, dependency

v00 - v01

- o Terminology clarifications
- o Provide example of impacting versus impacted dependencies

Acknowledgements

The authors would like to thank Jan Lindblad for his help during the design of these YANG modules. The authors would like to thank Stephane Litkowski and Charles Eckel for their reviews.

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Jean Quilbeuf
Independent

Email: jean@quilbeuf.net

Paolo Lucente
NTT
Siriusdreef 70-72
Hoofddorp, WT 2132
Netherlands

Email: paolo@ntt.net

Paolo Fasano
TIM S.p.A
via G. Reiss Romoli, 274
10148 Torino
Italy

Email: paolo2.fasano@telecomitalia.it

Thangam Arumugam
Cisco Systems, Inc.
Milpitas (California)
United States

Email: tarumuga@cisco.com

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:

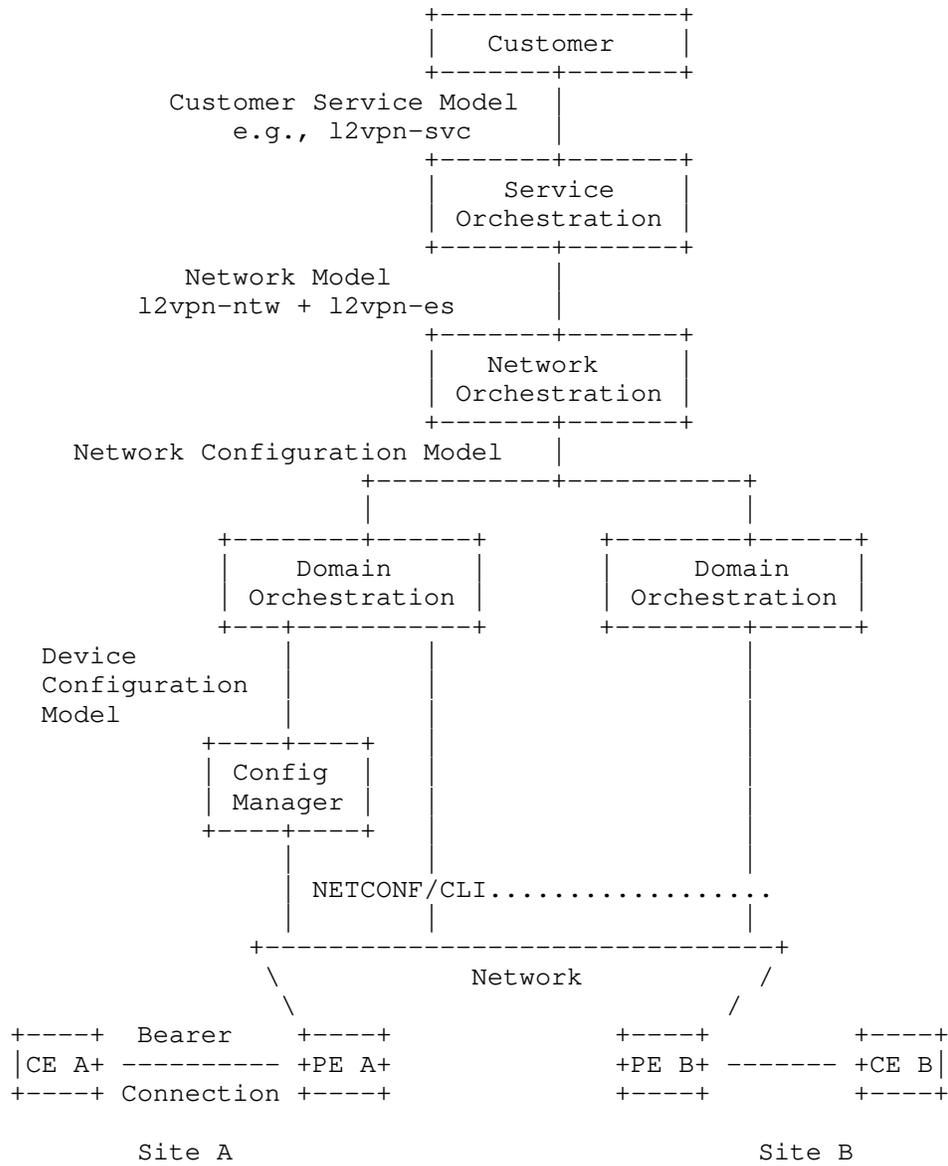
ACL	Access Control List
BGP	Border Gateway Protocol
BUM	Broadcast, unknown unicast, or multicast
CE	Customer Edge
ES	Ethernet Segment
ESI	Ethernet Segment Identifier
EVPN	Ethernet VPN
L2VPN	Layer 2 Virtual Private Network
L2SM	L2VPN Service Model
L2NM	L2VPN Network Model
MAC	Media Access Control
PBB	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).

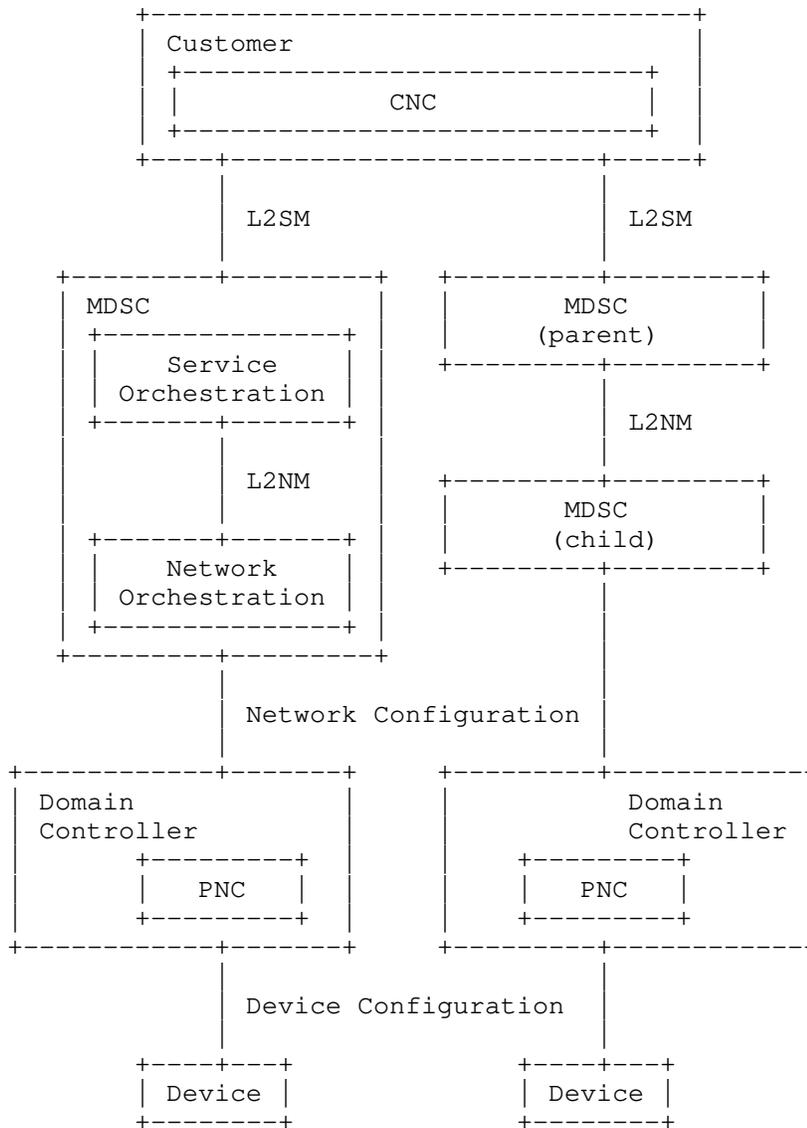


Figure 2: L2SM and L2NM in the Context of ACTN

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
          |
          | +--rw 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
vppls	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 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?

```


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 TAIID are encoded following the types defined in Section 3.4 of [RFC4446].

```

...
+--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)
  |   |   |   +--rw t-ldp-pw-type?
  |   |   |   |       identityref
  |   |   |   +--rw pw-type?                         identityref
  |   |   |   +--rw pw-description?                  string
  |   |   |   +--rw mac-addr-withdraw?              boolean
  |   |   |   +--rw pw-peer-list*
  |   |   |   |   [peer-addr vc-id]
  |   |   |   |   +--rw peer-addr
  |   |   |   |   |       inet:ip-address
  |   |   |   |   +--rw vc-id                       string
  |   |   |   |   +--rw pw-priority?                uint32
  |   |   |   +--rw qinq
  |   |   |   |   +--rw s-tag   dot1q-types:vlanid
  |   |   |   |   +--rw c-tag   dot1q-types:vlanid
  |   |   +--:(l2tp)
  |   ...
  ...

```

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 SAI, and a list of peers that are associated with a TAI (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 sai?                               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?
          +--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 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-lag] 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.

```

+--rw service
...
+--rw qos {vpn-common:qos}?
|
| +--rw qos-classification-policy
| |
| | +--rw rule* [id]
| | |
| | | +--rw id string
| | |
| | | +--rw (match-type)?
| | | |
| | | | +--:(match-flow)
| | | | |
| | | | | +--rw match-flow
| | | | | |
| | | | | | +--rw dscp? inet:dscp
| | | | | | +--rw dot1q? uint16
| | | | | | +--rw pcp? uint8
| | | | | | +--rw src-mac-address?
| | | | | | | yang:mac-address
| | | | | | +--rw dst-mac-address?
| | | | | | | yang:mac-address
| | | | | | +--rw color-type?
| | | | | | | identityref
| | | | | | +--rw any? empty
| | | | | +--:(match-application)
| | | | | +--rw match-application?
| | | | | | identityref
| | | | +--rw target-class-id? string
| | +--rw qos-profile
| | |
| | | +--rw qos-profile* [profile]
| | | +--rw profile leafref
| | | +--rw direction? identityref
|
...

```

Figure 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 included 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 ecrtmp {
  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
```



```
}
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 frame 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
      "Controlles 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;
    }
}
```



```
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 tair {
    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.";
```



```
    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
```


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.";
}
```



```
        "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 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
```


- * '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](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.

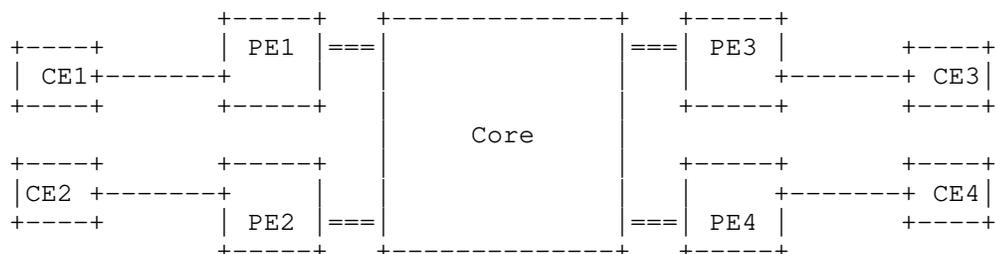


Figure 23: An Example of VPLS

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.

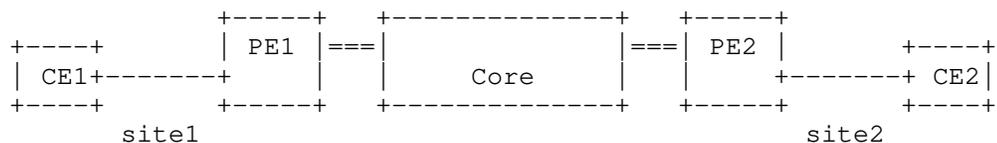


Figure 25: An Example of VPLS

```

{
  "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": {
          "sai": 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.

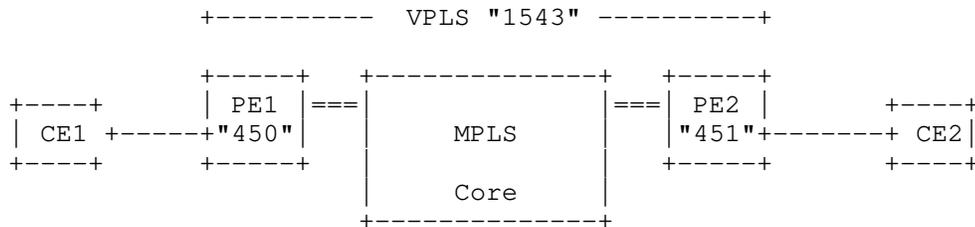


Figure 27: An Example of VPLS topology

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": {
```



```
    "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": [
      {

```

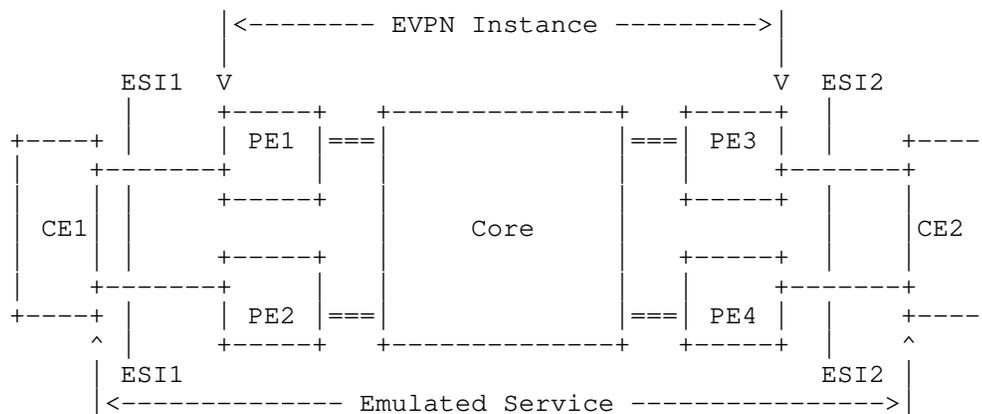



Figure 29: An Example of VPWS-EVPN

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": "es1"
        }
      ]
    }
  ],
},
{
  "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": "es1"
      }
    ]
  }
}
},
{
  "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
            }
          }
        }
      }
    ]
  }
}

```


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": "es1"
      }
    ]
  }
},
{
  "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": "es1"
    }
  ]
}
```

```

    }
  ]
}

```

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.

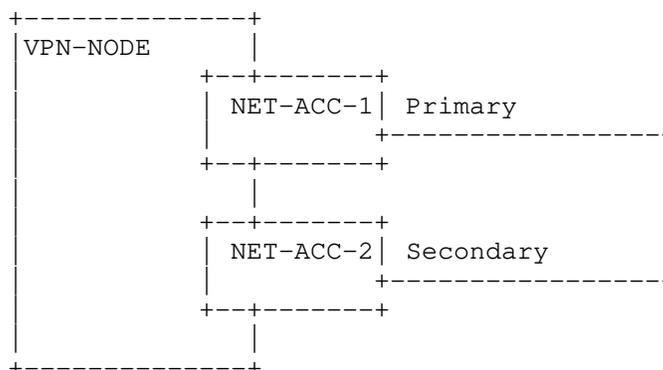


Figure 36: Example of Multiple VPN Network Accesses

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

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 11 April 2022

S. Barguil
O. Gonzalez de Dios, Ed.
Telefonica
M. Boucadair, Ed.
Orange
L. Munoz
Vodafone
A. Aguado
Nokia
8 October 2021

A Layer 3 VPN Network YANG Model
draft-ietf-opsawg-l3sm-l3nm-18

Abstract

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

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

Editorial Note (To be removed by RFC Editor)

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

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

Please update "RFC UUUU" to the RFC number to be assigned to I-D.ietf-opsawg-vpn-common.

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

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 11 April 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	5
3. Acronyms	6
4. L3NM Reference Architecture	7
5. Relation with other YANG Models	11
6. Sample Uses of the L3NM Data Model	12
6.1. Enterprise Layer 3 VPN Services	12
6.2. Multi-Domain Resource Management	13
6.3. Management of Multicast Services	13
7. Description of the L3NM YANG Module	13
7.1. Overall Structure of the Module	14
7.2. VPN Profiles	15
7.3. VPN Services	16
7.4. VPN Instance Profiles	20
7.5. VPN Nodes	22

7.6.	VPN Network Accesses	25
7.6.1.	Connection	28
7.6.2.	IP Connection	30
7.6.3.	CE-PE Routing Protocols	33
7.6.3.1.	Static Routing	35
7.6.3.2.	BGP	37
7.6.3.3.	OSPF	40
7.6.3.4.	IS-IS	42
7.6.3.5.	RIP	44
7.6.3.6.	VRRP	45
7.6.4.	OAM	47
7.6.5.	Security	48
7.6.6.	Services	49
7.6.6.1.	Overview	49
7.6.6.2.	QoS	50
7.7.	Multicast	55
8.	L3NM YANG Module	59
9.	Security Considerations	121
10.	IANA Considerations	122
11.	References	123
11.1.	Normative References	123
11.2.	Informative References	127
Appendix A.	L3VPN Examples	132
A.1.	4G VPN Provisioning Example	132
A.2.	Loopback Interface	137
A.3.	Overriding VPN Instance Profile Parameters	138
A.4.	Multicast VPN Provisioning Example	141
Appendix B.	Implementation Status	145
B.1.	Nokia Implementation	145
B.2.	Huawei Implementation	145
B.3.	Infinera Implementation	145
B.4.	Ribbon-ECI Implementation	145
B.5.	Juniper Implementation	146
Acknowledgements		146
Contributors		146
Authors' Addresses		147

1. Introduction

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

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

This document uses the common VPN YANG module defined in [I-D.ietf-opsawg-vpn-common].

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

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

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

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

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

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

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

2. Terminology

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

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

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

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

This document makes use of the following terms:

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

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

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

Network orchestrator: A functional entity that is hierarchically

intermediate between a service orchestrator and network controllers. A network orchestrator can manage one or several network controllers.

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

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

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

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

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

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

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

3. Acronyms

The following acronyms are used in the document:

ACL	Access Control List
AS	Autonomous System
ASM	Any-Source Multicast
ASN	AS Number
BSR	Bootstrap Router
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
CE	Customer Edge

CsC	Carriers' Carriers
IGMP	Internet Group Management Protocol
L3VPN	Layer 3 Virtual Private Network
L3SM	L3VPN Service Model
L3NM	L3VPN Network Model
MLD	Multicast Listener Discovery
MSDP	Multicast Source Discovery Protocol
MVPN	Multicast VPN
NAT	Network Address Translation
OAM	Operations, Administration, and Maintenance
OSPF	Open Shortest Path First
PE	Provider Edge
PIM	Protocol Independent Multicast
QoS	Quality of Service
RD	Route Distinguisher
RP	Rendezvous Point
RT	Route Target
SA	Security Association
SSM	Source-Specific Multicast
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding

4. L3NM Reference Architecture

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

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

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

The terminology from [RFC8309] is introduced to show the distinction between the customer service model, the service delivery model, the network configuration model, and the device configuration model. In that context, the "Domain Orchestration" and "Config Manager" roles may be performed by "Controllers".

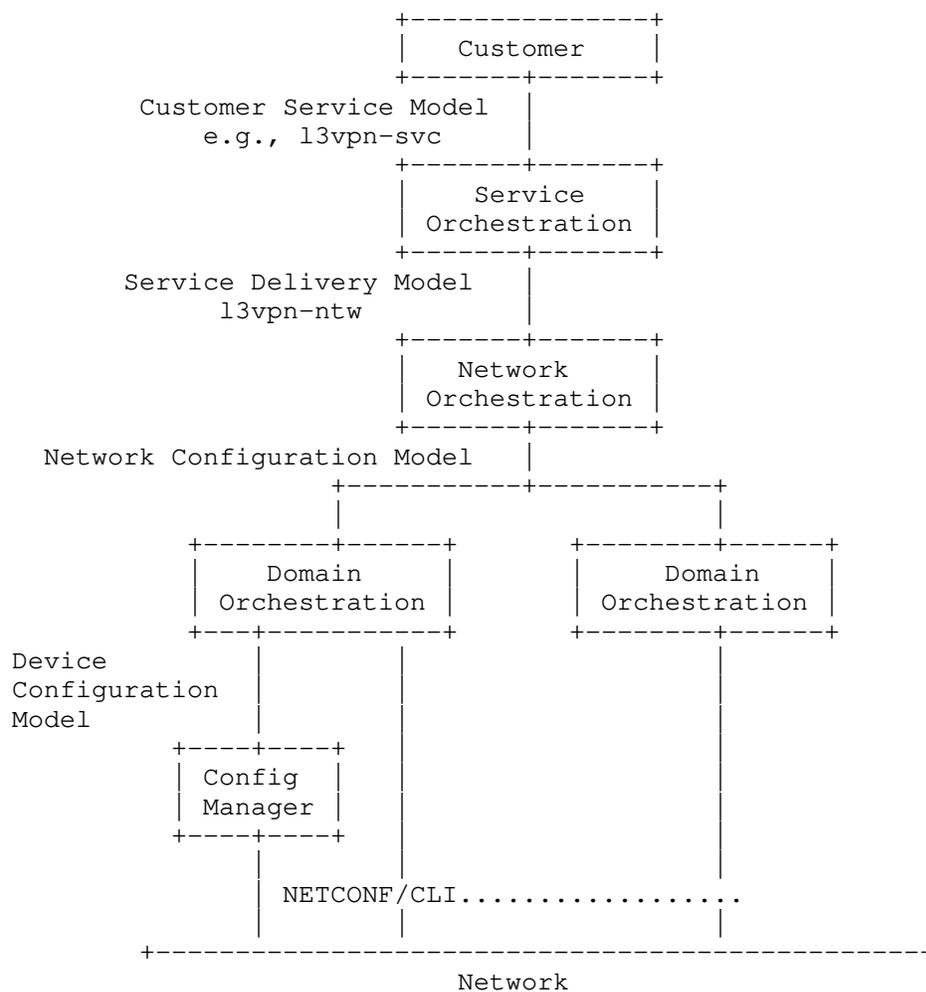


Figure 1: L3NM Reference Architecture

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

requested service [RFC7297], an enhanced VPN (VPN+) service [I-D.ietf-teas-enhanced-vpn], or an IETF network slice service [I-D.ietf-teas-ietf-network-slices].

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

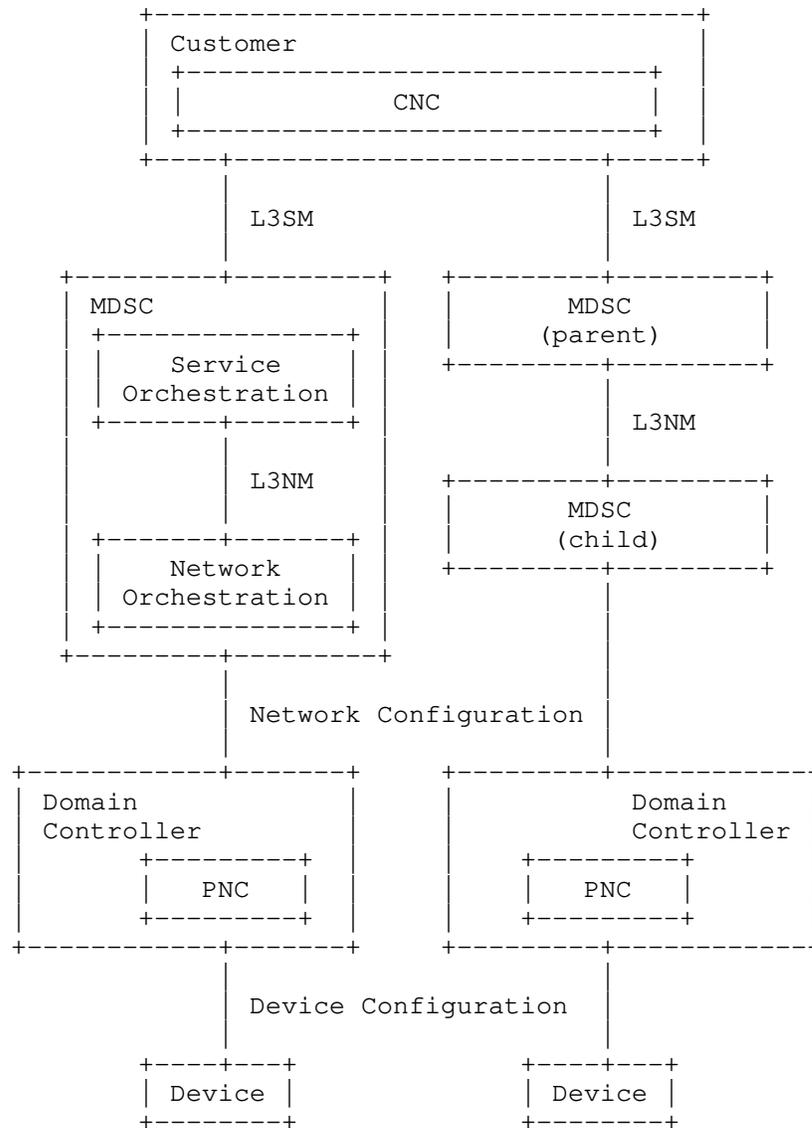


Figure 2: L3SM and L3NM in the Context of ACTN

5. Relation with other YANG Models

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

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

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

L3SM: L3NM is not a customer service model.

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

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

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

Network Topology Modules: An L3VPN involves nodes that are part of a

topology managed by the service provider network. The topology can be represented using the network topology YANG module defined in [RFC8345] or its extension such as a User-Network Interface (UNI) topology module (e.g., [I-D.ogondio-opsawg-uni-topology]).

Device Modules: L3NM is not a device model.

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

- * Routing management [RFC8349].
- * BGP [I-D.ietf-idr-bgp-model].
- * PIM [I-D.ietf-pim-yang].
- * NAT management [RFC8512].
- * QoS management [I-D.ietf-rtgwg-qos-model].
- * ACLs [RFC8519].

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

6. Sample Uses of the L3NM Data Model

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

6.1. Enterprise Layer 3 VPN Services

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

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

6.2. Multi-Domain Resource Management

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

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

6.3. Management of Multicast Services

Multicast services over L3VPN can be implemented using dual PIM MVPNs (also known as, Draft Rosen model) [RFC6037] or Multiprotocol BGP (MP-BGP)-based MVPNs [RFC6513][RFC6514]. Both methods are supported and equally effective, but the main difference is that MBGP-based MVPN does not require multicast configuration on the service provider network. MBGP MVPNs employ the intra-autonomous system BGP control plane and PIM sparse mode as the data plane. The PIM state information is maintained between PEs using the same architecture that is used for unicast VPNs.

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

7. Description of the L3NM YANG Module

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

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

7.1. Overall Structure of the Module

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

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

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

```

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

```

Figure 3: Overall L3NM Tree Structure

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

7.2. VPN Profiles

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

```

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

```

Figure 4: VPN Profiles Subtree Structure

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

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

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

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

- 'bfd-profile-identifier': A Bidirectional Forwarding Detection (BFD) profile refers to a set of BFD [RFC5880] policies that can be invoked when building a VPN service.
- 'forwarding-profile-identifier': A forwarding profile refers to the policies that apply to the forwarding of packets conveyed within a VPN. Such policies may consist, for example, of applying Access Control Lists (ACLs).
- 'routing-profile-identifier': A routing profile refers to a set of routing policies that will be invoked (e.g., BGP policies) when delivering the VPN service.

7.3. VPN Services

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

```

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

```

Figure 5: VPN Services Subtree Structure

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

- 'vpn-id': Is an identifier that is used to uniquely identify the L3VPN service within L3NM scope.
- 'vpn-name': Associates a name with the service in order to facilitate the identification of the service.
- 'vpn-description': Includes a textual description of the service.

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

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

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

'vpn-type': Indicates the VPN type. The values are taken from [I-D.ietf-opsawg-vpn-common]. For the L3NM, this is typically set to BGP/MPLS L3VPN, but other values may be defined in the future to support specific Layer 3 VPN capabilities (e.g., [I-D.ietf-bess-evpn-prefix-advertisement]).

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

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

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

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

More details are provided in Section 7.4.

'underlay-transport': Describes the preference for the transport

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

A rich set of protocol identifiers that can be used to refer to an underlay transport are defined in [I-D.ietf-opsawg-vpn-common].

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

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

'vpn-node': Is an abstraction that represents a set of policies applied to a network node and that belong to a single 'vpn-service'. A VPN service is typically built by adding instances of 'vpn-node' to the 'vpn-nodes' container.

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

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

More details are provided in Section 7.5.

7.4. VPN Instance Profiles

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

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

```

+--rw l3vpn-ntw
  +--rw vpn-profiles
  |   ...
  +--rw vpn-services
    +--rw vpn-service* [vpn-id]
      +--rw vpn-id                               vpn-common:vpn-id
      ...
      +--rw vpn-instance-profiles
        +--rw vpn-instance-profile* [profile-id]
          +--rw profile-id                       string
          +--rw role?                            identityref
          +--rw local-as?                        inet:as-number
          |   {vpn-common:rtg-bgp}?
          +--rw (rd-choice)?
          |   +--:(directly-assigned)
          |   |   +--rw rd?
          |   |   |   rt-types:route-distinguisher
          |   +--:(directly-assigned-suffix)
          |   |   +--rw rd-suffix?                uint16
          |   +--:(auto-assigned)
          |   |   +--rw rd-auto
          |   |   |   +--rw (auto-mode)?
          |   |   |   |   +--:(from-pool)
          |   |   |   |   |   +--rw rd-pool-name?  string
          |   |   |   |   +--:(full-auto)
          |   |   |   |   |   +--rw auto?          empty
          |   |   |   +--ro auto-assigned-rd?
          |   |   |   |   rt-types:route-distinguisher
          |   +--:(auto-assigned-suffix)
          |   |   +--rw rd-auto-suffix
          |   |   |   +--rw (auto-mode)?
          |   |   |   |   +--:(from-pool)
          |   |   |   |   |   +--rw rd-pool-name?  string
          |   |   |   |   +--:(full-auto)
          |   |   |   |   |   +--rw auto?          empty

```

```

|         +--ro auto-assigned-rd-suffix?  uint16
+---:(no-rd)
|         +--rw no-rd?                      empty
+---rw address-family* [address-family]
|         +--rw address-family              identityref
+---rw vpn-targets
|         +--rw vpn-target* [id]
|         |         +--rw id                  uint8
|         |         +--rw route-targets* [route-target]
|         |         |         +--rw route-target
|         |         |         |         rt-types:route-target
|         |         |         +--rw route-target-type
|         |         |         |         rt-types:route-target-type
|         |         +--rw vpn-policies
|         |         |         +--rw import-policy?  string
|         |         |         +--rw export-policy?  string
+---rw maximum-routes* [protocol]
|         +--rw protocol                      identityref
|         +--rw maximum-routes?  uint32
+---rw multicast {vpn-common:multicast}?
...

```

Figure 6: Subtree Structure of VPN Instance Profiles

The description of the listed data nodes is as follows:

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

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

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

'rd': As defined in [I-D.ietf-opsawg-vpn-common], the following RD assignment modes are supported: direct assignment, automatic assignment from a given pool, automatic assignment, and no assignment. For illustration purposes, the following modes can be used in the deployment cases:

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

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

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

Also, the module accommodates deployments where only the Assigned Number subfield of RDs (Section 4.2 of [RFC4364]) is assigned from a pool while the Administrator subfield is set to, 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.

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

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

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

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

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

7.5. VPN Nodes

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

```
+--rw l3vpn-ntw
  +--rw vpn-profiles
  |   ...
  +--rw vpn-services
    +--rw vpn-service* [vpn-id]
    ...
    +--rw vpn-nodes
      +--rw vpn-node* [vpn-node-id]
```

```

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

```

Figure 7: VPN Node Subtree Structure

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

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

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

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

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

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

'active-vpn-instance-profiles': Lists the set of active VPN instance profiles for this VPN node. Concretely, one or more VPN instance profiles that are defined at the VPN service level can be enabled at the VPN node level; each of these profiles is uniquely identified by means of 'profile-id'. The structure of 'active-vpn-instance-profiles' is the same as the one discussed in Section 7.4 except 'router-id'. The value of 'router-id' indicated under 'active-vpn-instance-profiles' takes precedence over the 'router-id' under the 'vpn-node' for the indicated address family. For example, Router IDs can be configured per address family. This capability can be used, for example, to configure an IPv6 address as a Router ID when such capability is supported by involved routers.

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

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

'groups': Lists the groups to which a VPN node belongs to

[I-D.ietf-opsawg-vpn-common]. The 'group-id' is used to associate, e.g., redundancy or protection constraints with VPN nodes.

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

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

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

See Section 7.6 for more details.

7.6. VPN Network Accesses

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

```

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

```

Figure 8: VPN Network Access Subtree Structure

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

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

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

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

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

- 'point-to-point': Represents a direct connection between the endpoints. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.
- 'multipoint': Represents a multipoint connection between the customer site and the PEs. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.
- 'irb': Represents a connection coming from an L2VPN service. An identifier of such service ('l2vpn-id') may be included in the 'connection' container as depicted in Figure 9. The controller must keep the relationship between the logical tunnels or bridges on the devices with the 'id' of the 'vpn-network-access'.
- 'loopback': Represents the creation of a logical interface on a device. An example to illustrate how a loopback interface can be used in the L3NM is provided in Appendix A.2.
- 'vpn-instance-profile': Provides a pointer to an active VPN instance profile at the VPN node level. Referencing an active VPN instance profile implies that all associated data nodes will be inherited by the VPN network access. However, some inherited data nodes (e.g., multicast) can be overridden at the VPN network access level. In such case, adjusted values take precedence over inherited ones.
- 'status': Indicates both operational and administrative status of a VPN network access.
- 'connection': Represents and groups the set of Layer 2 connectivity from where the traffic of the L3VPN in a particular VPN Network access is coming. See Section 7.6.1.
- 'ip-connection': Contains Layer 3 connectivity information of a VPN network access (e.g., IP addressing). See Section 7.6.2.
- 'routing-protocols': Includes the CE-PE routing configuration information. See Section 7.6.3.
- 'oam': Specifies the Operations, Administration, and Maintenance (OAM) mechanisms used for a VPN network access. See Section 7.6.4.
- 'security': Specifies the authentication and the encryption to be applied for a given VPN network access. See Section 7.6.5.

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

7.6.1. Connection

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

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

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

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

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

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

A site, as per [RFC4176] represents a VPN customer's location that is connected to the service provider network via a CE-PE link, which can access at least one VPN. The connection from the site to the service provider network is the bearer. Every site is associated with a list of bearers. A bearer is the layer two connection with the site. In the L3NM, it is assumed that the bearer has been allocated by the service provider at the service orchestration stage. The bearer is associated to a network element and a port. Hence, a bearer is just a 'bearer-reference' to allow the association between a service request (e.g., L3SM) and L3NM.

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

```

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

```

Figure 9: Connection Subtree Structure

7.6.2. IP Connection

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

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

```

...
+--rw vpn-network-accesses
  +--rw vpn-network-access* [id]
    ...
    +--rw ip-connection
      +--rw l3-termination-point?      string
      +--rw ipv4 {vpn-common:ipv4}?
      |   ...
      +--rw ipv6 {vpn-common:ipv6}?
      |   ...
      ...
    ...

```

Figure 10: IP Connection Subtree Structure

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

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

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

```

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

```

Figure 11: IP Connection Subtree Structure (IPv4)

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

```

...
+--rw ip-connection
|
|  +--rw l3-termination-point?      string
|  +--rw ipv4 {vpn-common:ipv4}?
|  |
|  |  ...
|  +--rw ipv6 {vpn-common:ipv6}?
|  |
|  |  +--rw local-address?          inet:ipv6-address
|  |  +--rw prefix-length?         uint8
|  |  +--rw address-allocation-type? identityref
|  |  +--rw (allocation-type)?
|  |  |
|  |  |  +--:(provider-dhcp)
|  |  |  |
|  |  |  |  +--rw provider-dhcp
|  |  |  |  |
|  |  |  |  |  +--rw dhcp-service-type?
|  |  |  |  |  |
|  |  |  |  |  |  enumeration
|  |  |  |  |  +--rw (service-type)?
|  |  |  |  |  |
|  |  |  |  |  |  +--:(relay)
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw server-ip-address*
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  inet:ipv6-address
|  |  |  |  |  |  +--:(server)
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw (address-assign)?
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  +--:(number)
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  +--rw number-of-dynamic-address?
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  uint16
|  |  |  |  |  |  |  |  +--:(explicit)
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  +--rw customer-addresses
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  +--rw address-pool* [pool-id]
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  +--rw pool-id      string
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  +--rw start-address
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  inet:ipv6-address
|  |  |  |  |  |  |  |  |  |  |  |  |  |  +--rw end-address?
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  inet:ipv6-address
|  |  |  |  |  |  +--:(dhcp-relay)
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw customer-dhcp-servers
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  +--rw server-ip-address*
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  inet:ipv6-address
|  |  |  |  |  |  +--:(static-addresses)
|  |  |  |  |  |  |
|  |  |  |  |  |  |  ...

```

Figure 12: IP Connection Subtree Structure (IPv6)

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

```

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

```

Figure 13: IP Connection Subtree Structure (Static Mode)

7.6.3. CE-PE Routing Protocols

A VPN service provider can configure one or more routing protocols associated with a particular 'vpn-network-access'. Such routing protocols are enabled between the PE and the CE. Each instance is uniquely identified to accommodate scenarios where multiple instances of the same routing protocol have to be configured on the same link.

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

```

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

```

Figure 14: Routing Subtree Structure

Multiple routing instances can be defined; each uniquely identified by an 'id'. The type of routing instance is indicated in 'type'. The values of these attributes are those defined in [I-D.ietf-opsawg-vpn-common] ('routing-protocol-type' identity).

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

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

* A combination thereof.

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

7.6.3.1. Static Routing

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

- 'next-hop': Indicates the next-hop to be used for the static route. It can be identified by an IP address, an interface, etc.
- 'bfd-enable': Indicates whether BFD is enabled or disabled for this static route entry.
- 'metric': Indicates the metric associated with the static route entry.
- 'preference': Indicates the preference associated with the static route entry. This preference is used to selecting a preferred route among routes to the same destination prefix.
- 'status': Used to convey the status of a static route entry. This data node can also be used to control the (de)activation of individual static route entries.

7.6.3.2. BGP

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

```

...
+--rw routing-protocols
|
|  +--rw routing-protocol* [id]
|  |
|  |  ...
|  |  +--rw bgp
|  |  |
|  |  |  +--rw description?          string
|  |  |  +--rw local-as?             inet:as-number
|  |  |  +--rw peer-as               inet:as-number
|  |  |  +--rw address-family?       identityref
|  |  |  +--rw local-address?        union
|  |  |  +--rw neighbor*             inet:ip-address
|  |  |  +--rw multihop?              uint8
|  |  |  +--rw as-override?          boolean
|  |  |  +--rw allow-own-as?          uint8
|  |  |  +--rw prepend-global-as?    boolean
|  |  |  +--rw send-default-route?   boolean
|  |  |  +--rw site-of-origin?       rt-types:route-origin
|  |  |  +--rw ipv6-site-of-origin?  rt-types:ipv6-route-origin
|  |  |  +--rw redistribute-connected* [address-family]
|  |  |  |  +--rw address-family      identityref
|  |  |  |  +--rw enable?            boolean
|  |  |  +--rw bgp-max-prefix

```

```

|
| | +--rw max-prefix?          uint32
| | +--rw warning-threshold?  decimal64
| | +--rw violate-action?     enumeration
| | +--rw restart-timer?     uint32
| +--rw bgp-timers
| | +--rw keepalive?         uint16
| | +--rw hold-time?        uint16
| +--rw authentication
| | +--rw enable?           boolean
| | +--rw keying-material
| | | +--rw (option)?
| | | | +--:(ao)
| | | | | +--rw enable-ao?         boolean
| | | | | +--rw ao-keychain?      key-chain:key-chain-ref
| | | | +--:(md5)
| | | | | +--rw md5-keychain?    key-chain:key-chain-ref
| | | | +--:(explicit)
| | | | | +--rw key-id?          uint32
| | | | | +--rw key?            string
| | | | | +--rw crypto-algorithm? identityref
| | | | +--:(ipsec)
| | | | | +--rw sa?             string
| +--rw status
| | +--rw admin-status
| | | +--rw status?           identityref
| | | +--rw last-change?     yang:date-and-time
| +--ro oper-status
| | +--ro status?           identityref
| | +--ro last-change?     yang:date-and-time
...

```

Figure 16: BGP Routing Subtree Structure

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

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

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

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

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

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

- 'local-address': Specifies an address or a reference to an interface to use when establishing the BGP transport session.
- 'neighbor': Can indicate two neighbors (each for a given address-family) or one neighbor (if 'address-family' attribute is set to dual-stack). A list of IP address(es) of the BGP neighbors can be then conveyed in this data node.
- 'multihop': Indicates the number of allowed IP hops between a PE and its BGP peer.
- 'as-override': If set, this parameter indicates whether ASN override is enabled, i.e., replace the ASN of the customer specified in the AS_PATH BGP attribute with the ASN identified in the 'local-as' attribute.
- 'allow-own-as': Is used in some topologies (e.g., hub-and-spoke) to allow the provider's ASN to be included in the AS_PATH BGP attribute received from a CE. Loops are prevented by setting 'allow-own-as' to a maximum number of provider's ASN occurrences. This parameter is set by default to '0' (that is, reject any AS_PATH attribute that includes the provider's ASN).
- 'prepend-global-as': When distinct ASNs are configured in the VPN node and network access levels, this parameter controls whether the ASN provided at the VPN node level is prepended to the AS_PATH attribute.
- 'send-default-route': Controls whether default routes can be advertised to the peer.
- 'site-of-origin': Is meant to uniquely identify the set of routes learned from a site via a particular CE/PE connection and is used to prevent routing loops (Section 7 of [RFC4364]). The Site of Origin attribute is encoded as a Route Origin Extended Community.
- 'ipv6-site-of-origin': Carries an IPv6 Address Specific BGP Extended Community that is used to indicate the Site of Origin for VRF information [RFC5701]. It is used to prevent routing loops.
- 'redistribute-connected': Controls whether the PE-CE link is advertised to other PEs.

- '**bgp-max-prefix**': Controls the behavior when a prefix maximum is reached.
- '**max-prefix**': Indicates the maximum number of BGP prefixes allowed in the BGP session. If the limit is reached, the action indicated in '**violate-action**' will be followed.
- '**warning-threshold**': A warning notification is triggered when this limit is reached.
- '**violate-action**': Indicates which action to execute when the maximum number of BGP prefixes is reached. Examples of such actions are: send a warning message, discard extra paths from the peer, or restart the session.
- '**restart-timer**': Indicates, in seconds, the time interval after which the BGP session will be reestablished.
- '**bgp-timers**': Two timers can be captured in this container: (1) '**hold-time**' which is the time interval that will be used for the HoldTimer (Section 4.2 of [RFC4271]) when establishing a BGP session. (2) '**keepalive**' which is the time interval for the KeepAlive timer between a PE and a BGP peer (Section 4.4 of [RFC4271]). Both timers are expressed in seconds.
- '**authentication**': The module adheres to the recommendations in Section 13.2 of [RFC4364] as it allows enabling TCP-AO [RFC5925] and accommodates the installed base that makes use of MD5. In addition, the module includes a provision for the use of IPsec.
- This version of the L3NM assumes that TCP-AO specific parameters are preconfigured as part of the key-chain that is referenced in the L3NM. No assumption is made about how such a key-chain is pre-configured. However, the structure of the key-chain should cover data nodes beyond those in [RFC8177], mainly SendID and RecvID (Section 3.1 of [RFC5925]).
- '**status**': Indicates the status of the BGP routing instance.

7.6.3.3. OSPF

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

```

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

```

Figure 17: OPSF Routing Subtree Structure

The following data nodes are captured in Figure 17:

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

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

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

'metric': Associates a metric with OSPF routes.

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

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

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

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

7.6.3.4. IS-IS

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

```

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

```

Figure 18: IS-IS Routing Subtree Structure

The following IS-IS data nodes are supported:

- 'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.
- 'area-address': Indicates the IS-IS area address.
- 'level': Indicates the IS-IS level: Level 1, Level 2, or both.
- 'metric': Associates a metric with IS-IS routes.
- 'mode': Indicates the IS-IS interface mode type. It can be set to 'active' (that is, send or receive IS-IS protocol control packets) or 'passive' (that is, suppress the sending of IS-IS updates through the interface).

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

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

7.6.3.5. RIP

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

```

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

```

Figure 19: RIP Subtree Structure

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

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. This parameter is used to determine whether RIPv2 [RFC2453] and/or RIPv6 [RFC2080] are to be enabled.

'timers': Indicates the following timers:

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

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

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

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

These timers are expressed in seconds.

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

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

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

7.6.3.6. VRRP

The model (Figure 20) allows enabling VRRP on the 'vpn-network-access' interface.

```

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

```

Figure 20: VRRP Subtree Structure

The following data nodes are supported:

- 'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. Note that VRRP version 3 [RFC5798] supports both IPv4 and IPv6.
- 'vrrp-group': Is used to identify the VRRP group.
- 'backup-peer': Carries the IP address of the peer.
- 'virtual-ip-address': Includes virtual IP addresses for a single VRRP group.
- 'priority': Assigns the VRRP election priority for the backup virtual router.
- 'ping-reply': Controls whether ping requests can be replied to.
- 'status': Indicates the status of the VRRP instance.

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

7.6.4. OAM

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

```

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

```

Figure 21: IP Connection Subtree Structure (OAM)

The following OAM data nodes can be specified:

- 'session-type': Indicates which BFD flavor is used to set up the session (e.g., classic BFD [RFC5880], Seamless BFD [RFC7880]). By default, the BFD session is assumed to follow the behavior specified in [RFC5880].
- 'desired-min-tx-interval': Is the minimum interval, in microseconds, that a PE would like to use when transmitting BFD Control packets less any jitter applied.
- 'required-min-rx-interval': Is the minimum interval, in microseconds, between received BFD Control packets that a PE is capable of supporting, less any jitter applied by the sender.
- 'local-multiplier': The negotiated transmit interval, multiplied by this value, provides the detection time for the peer.
- 'holdtime': Is used to indicate the expected BFD holddown time, in

milliseconds. This value may be inherited from the service request (see Section 6.3.2.2.2 of [RFC8299]).

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

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

'status': Indicates the status of BFD.

7.6.5. Security

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

```

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

```

Figure 22: Security Subtree Structure

7.6.6. Services

7.6.6.1. Overview

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

```

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

```

Figure 23: Services Subtree Structure

The following data nodes are defined:

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

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

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

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

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

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

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

7.6.6.2. QoS

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

```

...
+--rw qos {vpn-common:qos}?
  +--rw qos-classification-policy
    +--rw rule* [id]
      +--rw id string
      +--rw (match-type)?
        +--:(match-flow)
          +--rw (13)?
            +--:(ipv4)
              ...
            +--:(ipv6)
              ...
          +--rw (14)?
            +--:(tcp)
              ...
            +--:(udp)
              ...
          +--:(match-application)
            +--rw match-application?
              identityref
        +--rw target-class-id?
          string
    +--rw qos-action
      +--rw rule* [id]
        +--rw id string
        +--rw target-class-id? string
        +--rw inbound-rate-limit? decimal64
        +--rw outbound-rate-limit? decimal64
    +--rw qos-profile
      +--rw qos-profile* [profile]
        +--rw profile leafref
        +--rw direction? identityref
...

```

Figure 24: Services Subtree Structure

QoS classification can be based on many criteria such as:

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

```

+--rw qos {vpn-common:qos}?
  +--rw qos-classification-policy
    +--rw rule* [id]
      +--rw id string
      +--rw (match-type)?
        +--:(match-flow)
          +--rw (l3)?
            +--:(ipv4)
              +--rw ipv4
                +--rw dscp? inet:dscp
                +--rw ecn? uint8
                +--rw length? uint16
                +--rw ttl? uint8
                +--rw protocol? uint8
                +--rw ihl? uint8
                +--rw flags? bits
                +--rw offset? uint16
                +--rw identification? uint16
                +--rw (destination-network)?
                  +--:(destination-ipv4-network)
                    +--rw destination-ipv4-network?
                      inet:ipv4-prefix
                +--rw (source-network)?
                  +--:(source-ipv4-network)
                    +--rw source-ipv4-network?
                      inet:ipv4-prefix
            +--:(ipv6)
              +--rw ipv6
                +--rw dscp? inet:dscp
                +--rw ecn? uint8
                +--rw length? uint16
                +--rw ttl? uint8
                +--rw protocol? uint8
                +--rw (destination-network)?
                  +--:(destination-ipv6-network)
                    +--rw destination-ipv6-network?
                      inet:ipv6-prefix
                +--rw (source-network)?
                  +--:(source-ipv6-network)
                    +--rw source-ipv6-network?
                      inet:ipv6-prefix
              +--rw flow-label?
                inet:ipv6-flow-label
          ...

```

Figure 25: QoS Subtree Structure (L3)

Layer 4: As discussed in [I-D.ietf-opsawg-vpn-common], any layer 4

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

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

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

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


Application match: Relies upon application-specific classification.

7.7. Multicast

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

```

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

```

Figure 27: Overall Multicast Subtree Structure

Multicast-related data nodes at the VPN instance profile level has the structure that is shown in Figure 30.

```

...
+--rw vpn-services
  +--rw vpn-service* [vpn-id]
    ...
    +--rw vpn-instance-profiles
      | +--rw vpn-instance-profile* [profile-id]
      |   ....
      |   +--rw multicast {vpn-common:multicast}?
      |   +--rw tree-flavor? identityref

```

```

+--rw rp
|
|  +--rw rp-group-mappings
|  |
|  |  +--rw rp-group-mapping* [id]
|  |  |
|  |  |  +--rw id                               uint16
|  |  |  +--rw provider-managed
|  |  |  |
|  |  |  |  +--rw enabled?                       boolean
|  |  |  |  +--rw rp-redundancy?                 boolean
|  |  |  |  +--rw optimal-traffic-delivery?     boolean
|  |  |  |  +--rw anycast
|  |  |  |  |
|  |  |  |  |  +--rw local-address?             inet:ip-address
|  |  |  |  |  +--rw rp-set-address*           inet:ip-address
|  |  |  |  +--rw rp-address                   inet:ip-address
|  |  |  +--rw groups
|  |  |  |
|  |  |  |  +--rw group* [id]
|  |  |  |  |
|  |  |  |  |  +--rw id                               uint16
|  |  |  |  |  +--rw (group-format)
|  |  |  |  |  |
|  |  |  |  |  |  +--:(group-prefix)
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +--rw group-address?         inet:ip-prefix
|  |  |  |  |  |  |  +--:(startend)
|  |  |  |  |  |  |  +--rw group-start?          inet:ip-address
|  |  |  |  |  |  |  +--rw group-end?           inet:ip-address
|  |  |  +--rw rp-discovery
|  |  |  |
|  |  |  |  +--rw rp-discovery-type?             identityref
|  |  |  |  +--rw bsr-candidates
|  |  |  |  |
|  |  |  |  |  +--rw bsr-candidate-address*       inet:ip-address
|  |  |  +--rw igmp {vpn-common:igmp and vpn-common:ipv4}?
|  |  |  |
|  |  |  |  +--rw static-group* [group-addr]
|  |  |  |  |
|  |  |  |  |  +--rw group-addr
|  |  |  |  |  |
|  |  |  |  |  |  rt-types:ipv4-multicast-group-address
|  |  |  |  |  +--rw source-addr?
|  |  |  |  |  |
|  |  |  |  |  |  rt-types:ipv4-multicast-source-address
|  |  |  |  +--rw max-groups?                   uint32
|  |  |  |  +--rw max-entries?                   uint32
|  |  |  |  +--rw version?                       identityref
|  |  |  +--rw mld {vpn-common:mld and vpn-common:ipv6}?
|  |  |  |
|  |  |  |  +--rw static-group* [group-addr]
|  |  |  |  |
|  |  |  |  |  +--rw group-addr
|  |  |  |  |  |
|  |  |  |  |  |  rt-types:ipv6-multicast-group-address
|  |  |  |  |  +--rw source-addr?
|  |  |  |  |  |
|  |  |  |  |  |  rt-types:ipv6-multicast-source-address
|  |  |  |  +--rw max-groups?                   uint32
|  |  |  |  +--rw max-entries?                   uint32
|  |  |  |  +--rw version?                       identityref
|  |  |  +--rw pim {vpn-common:pim}?
|  |  |  |
|  |  |  |  +--rw hello-interval?               rt-types:timer-value-seconds16
|  |  |  |  +--rw dr-priority?                   uint32

```

...

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

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

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

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

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

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

```

...
+--rw vpn-nodes
  +--rw vpn-node* [vpn-node-id]
    ...
    +--rw active-vpn-instance-profiles
      +--rw vpn-instance-profile* [profile-id]
        ...
        +--rw multicast {vpn-common:multicast}?
          +--rw tree-flavor* identityref
          +--rw rp
            | ...
          +--rw igmp {vpn-common:igmp and vpn-common:ipv4}?
            | ...
          +--rw mld {vpn-common:mld and vpn-common:ipv6}?
            | ...
          +--rw pim {vpn-common:pim}?
            | ...
          ...

```

Figure 29: Multicast Subtree Structure (VPN Node Level)

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

```

...
+--rw vpn-network-accesses
  +--rw vpn-network-access* [id]
    ...
    +--rw service
      ...
      +--rw multicast {vpn-common:multicast}?
        +--rw access-type? enumeration
        +--rw address-family? identityref
        +--rw protocol-type? enumeration
        +--rw remote-source? boolean
        +--rw igmp {vpn-common:igmp}?
          +--rw static-group* [group-addr]
            +--rw group-addr
              | rt-types:ipv4-multicast-group-address
            +--rw source-addr?
              | rt-types:ipv4-multicast-source-address
          +--rw max-groups? uint32
          +--rw max-entries? uint32
          +--rw max-group-sources? uint32
          +--rw version? identityref
          +--rw status
            +--rw admin-status

```

```

    |         |   +--rw status?           identityref
    |         |   +--rw last-change?     yang:date-and-time
    |         +--ro oper-status
    |         |   +--ro status?          identityref
    |         |   +--ro last-change?     yang:date-and-time
+--rw mld {vpn-common:mld}?
  +--rw static-group* [group-addr]
    |   +--rw group-addr
    |   |   rt-types:ipv6-multicast-group-address
    |   +--rw source-addr?
    |   |   rt-types:ipv6-multicast-source-address
  +--rw max-groups?           uint32
  +--rw max-entries?          uint32
  +--rw max-group-sources?    uint32
  +--rw version?              identityref
  +--rw status
    +--rw admin-status
      |   +--rw status?              identityref
      |   +--rw last-change?         yang:date-and-time
    +--ro oper-status
      |   +--ro status?              identityref
      |   +--ro last-change?         yang:date-and-time
+--rw pim {vpn-common:pim}?
  +--rw hello-interval?      rt-types:timer-value-seconds16
  +--rw dr-priority?          uint32
  +--rw status
    +--rw admin-status
      |   +--rw status?              identityref
      |   +--rw last-change?         yang:date-and-time
    +--ro oper-status
      |   +--ro status?              identityref
      |   +--ro last-change?         yang:date-and-time

```

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

8. L3NM YANG Module

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

```

<CODE BEGINS> file "ietf-l3vpn-ntw@2021-09-28.yang"
module ietf-l3vpn-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw";
  prefix l3nm;

  import ietf-vpn-common {
    prefix vpn-common;

```

```
    reference
      "RFC UUUU: A Layer 2/3 VPN Common YANG Model";
  }
import ietf-inet-types {
  prefix inet;
  reference
    "RFC 6991: Common YANG Data Types, Section 4";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types, Section 3";
}
import ietf-key-chain {
  prefix key-chain;
  reference
    "RFC 8177: YANG Key Chain.";
}
import ietf-routing-types {
  prefix rt-types;
  reference
    "RFC 8294: Common YANG Data Types for the Routing Area";
}
import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343: A YANG Data Model for Interface Management";
}

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

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

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

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://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 2021-09-28 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A Layer 3 VPN Network YANG Model";
}

/* Features */

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

/* Identities */

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

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

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

```
        customer.";
    }

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

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

    identity slaac {
        if-feature "vpn-common:ipv6";
        base address-allocation-type;
        description
            "The Provider's network uses IPv6 SLAAC to provide addressing
            to the customer.";
        reference
            "RFC 4862: IPv6 Stateless Address Autoconfiguration";
    }

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

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

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

```

```
    }

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

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

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

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

    /* Typedefs */

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

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

    /* Groupings */
```

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

```

```
        "Indicates the maximum number of prefixes that the
        VRF can accept for this address family and protocol.";
    }
}
}
container multicast {
  if-feature "vpn-common:multicast";
  description
    "Global multicast parameters.";
  leaf tree-flavor {
    type identityref {
      base vpn-common:multicast-tree-type;
    }
    description
      "Type of the multicast tree to be used.";
  }
}
container rp {
  description
    "Rendezvous Point (RP) parameters.";
  container rp-group-mappings {
    description
      "RP-to-group mappings parameters.";
    list rp-group-mapping {
      key "id";
      description
        "List of RP-to-group mappings.";
      leaf id {
        type uint16;
        description
          "Unique identifier for the mapping.";
      }
    }
  }
  container provider-managed {
    description
      "Parameters for a provider-managed RP.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "Set to true if the Rendezvous Point (RP)
        must be a provider-managed node. Set to
        false if it is a customer-managed node.";
    }
  }
  leaf rp-redundancy {
    type boolean;
    default "false";
    description
      "If set to true, it indicates that a redundancy
      mechanism for the RP is required.";
  }
}
```

```
}
leaf optimal-traffic-delivery {
  type boolean;
  default "false";
  description
    "If set to true, the service provider (SP) must
     ensure that the traffic uses an optimal path.
     An SP may use Anycast RP or RP-tree-to-SPT
     switchover architectures.";
}
container anycast {
  when "../rp-redundancy = 'true' and
        ../optimal-traffic-delivery = 'true'" {
    description
      "Only applicable if both RP redundancy and
       delivery through optimal path are
       activated.";
  }
  description
    "PIM Anycast-RP parameters.";
  leaf local-address {
    type inet:ip-address;
    description
      "IP local address for PIM RP. Usually, it
       corresponds to the Router ID or the
       primary address.";
  }
  leaf-list rp-set-address {
    type inet:ip-address;
    description
      "Specifies the IP address of other RP routers
       that share the same RP IP address.";
  }
}
}
leaf rp-address {
  when "../provider-managed/enabled = 'false'" {
    description
      "Relevant when the RP is not
       provider-managed.";
  }
  type inet:ip-address;
  mandatory true;
  description
    "Defines the address of the RP.
     Used if the RP is customer-managed.";
}
container groups {
```



```
    default "vpn-common:static-rp";
    description
      "Type of RP discovery used.";
  }
  container bsr-candidates {
    when "derived-from-or-self(../rp-discovery-type, "
      + "'vpn-common:bsr-rp')" {
      description
        "Only applicable if discovery type is BSR-RP.";
    }
    description
      "Container for the customer Bootstrap Router (BSR)
      candidate's addresses.";
    leaf-list bsr-candidate-address {
      type inet:ip-address;
      description
        "Specifies the address of candidate BSR.";
    }
  }
}
}
container igmp {
  if-feature "vpn-common:igmp and vpn-common:ipv4";
  description
    "Includes IGMP-related parameters.";
  list static-group {
    key "group-addr";
    description
      "Multicast static source/group associated to the
      IGMP session.";
    leaf group-addr {
      type rt-types:ipv4-multicast-group-address;
      description
        "Multicast group IPv4 address.";
    }
    leaf source-addr {
      type rt-types:ipv4-multicast-source-address;
      description
        "Multicast source IPv4 address.";
    }
  }
  leaf max-groups {
    type uint32;
    description
      "Indicates the maximum number of groups.";
  }
  leaf max-entries {
    type uint32;
  }
}
```

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

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

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

container l3vpn-ntw {
    description
```

```
    "Main container for L3VPN services management.";
container vpn-profiles {
  description
    "Contains a set of valid VPN profiles to reference in the VPN
    service.";
  uses vpn-common:vpn-profile-cfg;
}
container vpn-services {
  description
    "Container for the VPN services.";
  list vpn-service {
    key "vpn-id";
    description
      "List of VPN services.";
    uses vpn-common:vpn-description;
    leaf parent-service-id {
      type vpn-common:vpn-id;
      description
        "Pointer to the parent service, if any.
        A parent service can be an L3SM, a slice request, a VPN+
        service, etc.";
    }
    leaf vpn-type {
      type identityref {
        base vpn-common:service-type;
      }
      description
        "Indicates the service type.";
    }
    leaf vpn-service-topology {
      type identityref {
        base vpn-common:vpn-topology;
      }
      default "vpn-common:any-to-any";
      description
        "VPN service topology.";
    }
  }
  uses vpn-common:service-status;
  container vpn-instance-profiles {
    description
      "Container for a list of VPN instance profiles.";
    list vpn-instance-profile {
      key "profile-id";
      description
        "List of VPN instance profiles.";
      leaf profile-id {
        type string;
        description

```

```
        "VPN instance profile identifier.";
    }
    leaf role {
        type identityref {
            base vpn-common:role;
        }
        default "vpn-common:any-to-any-role";
        description
            "Role of the VPN node in the VPN.";
    }
    uses vpn-instance-profile;
}
}
container underlay-transport {
    description
        "Container for underlay transport.";
    uses vpn-common:underlay-transport;
}
container external-connectivity {
    if-feature "vpn-common:external-connectivity";
    description
        "Container for external connectivity.";
    choice profile {
        description
            "Choice for the external connectivity profile.";
        case profile {
            leaf profile-name {
                type leafref {
                    path "/l3vpn-ntw/vpn-profiles"
                        + "/valid-provider-identifiers"
                        + "/external-connectivity-identifier/id";
                }
                description
                    "Name of the service provider's profile to be applied
                    at the VPN service level.";
            }
        }
    }
}
}
container vpn-nodes {
    description
        "Container for VPN nodes.";
    list vpn-node {
        key "vpn-node-id";
        description
            "Includes a list of VPN nodes.";
        leaf vpn-node-id {
            type vpn-common:vpn-id;
        }
    }
}
```

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

```

```
        "Router-id per address family.";
    leaf address-family {
        type identityref {
            base vpn-common:address-family;
        }
        description
            "Indicates the address family for which the
             Router-ID applies.";
    }
    leaf router-id {
        type inet:ip-address;
        description
            "The router-id information can be an IPv4 or IPv6
             address. This can be used, for example, to
             configure an IPv6 address as a router-id
             when such capability is supported by underlay
             routers. In such case, the configured value
             overrides the generic one defined at the VPN
             node level.";
    }
    }
    uses vpn-instance-profile;
}
container msdp {
    if-feature "msdp";
    description
        "Includes MSDP-related parameters.";
    leaf peer {
        type inet:ipv4-address;
        description
            "Indicates the IPv4 address of the MSDP peer.";
    }
    leaf local-address {
        type inet:ipv4-address;
        description
            "Indicates the IPv4 address of the local end.
             This local address must be configured on
             the node.";
    }
    uses vpn-common:service-status;
}
uses vpn-common:vpn-components-group;
uses vpn-common:service-status;
container vpn-network-accesses {
    description
        "List of network accesses.";
    list vpn-network-access {
```

```
key "id";
description
  "List of network accesses.";
leaf id {
  type vpn-common:vpn-id;
  description
    "Identifier for the network access.";
}
leaf interface-id {
  type string;
  description
    "Identifier for the physical or logical
    interface.
    The identification of the sub-interface
    is provided at the connection and/or IP
    connection levels.";
}
leaf description {
  type string;
  description
    "Textual description of the network access.";
}
leaf vpn-network-access-type {
  type identityref {
    base vpn-common:site-network-access-type;
  }
  default "vpn-common:point-to-point";
  description
    "Describes the type of connection, e.g.,
    point-to-point.";
}
leaf vpn-instance-profile {
  type leafref {
    path "/l3vpn-ntw/vpn-services/vpn-service/vpn-nodes"
      + "/vpn-node/active-vpn-instance-profiles"
      + "/vpn-instance-profile/profile-id";
  }
  description
    "An identifier of an active VPN instance profile.";
}
uses vpn-common:service-status;
container connection {
  description
    "Defines layer 2 protocols and parameters that are
    required to enable connectivity between the PE
    and the CE.";
  container encapsulation {
    description
```

```
    "Container for layer 2 encapsulation.";
leaf type {
  type identityref {
    base vpn-common:encapsulation-type;
  }
  default "vpn-common:priority-tagged";
  description
    "Encapsulation type. By default, the type of
    the tagged interface is 'priority-tagged'.";
}
container dot1q {
  when "derived-from-or-self(..type, "
    + "'vpn-common:dot1q')" {
    description
      "Only applies when the type of the
      tagged interface is 'dot1q'.";
  }
  description
    "Tagged interface.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:c-vlan";
    description
      "Tag type. By default, the tag type is
      'c-vlan'.";
  }
  leaf cvlan-id {
    type uint16 {
      range "1..4094";
    }
    description
      "VLAN identifier.";
  }
}
container priority-tagged {
  when "derived-from-or-self(..type, "
    + "'vpn-common:priority-tagged')" {
    description
      "Only applies when the type of the
      tagged interface is 'priority-tagged'.";
  }
  description
    "Priority tagged.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
  }
}
```

```
    }
    default "vpn-common:c-vlan";
    description
      "Tag type. By default, the tag type is
      'c-vlan'.";
  }
}
container qinq {
  when "derived-from-or-self(..type, "
    + "'vpn-common:qinq')" {
    description
      "Only applies when the type of the tagged
      interface is QinQ.";
  }
  description
    "Includes QinQ parameters.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:s-c-vlan";
    description
      "Tag type. By default, the tag type is
      'c-s-vlan'.";
  }
  leaf svlan-id {
    type uint16;
    mandatory true;
    description
      "S-VLAN identifier.";
  }
  leaf cvlan-id {
    type uint16;
    mandatory true;
    description
      "C-VLAN identifier.";
  }
}
}
choice l2-service {
  description
    "The layer 2 connectivity service can be
    provided by indicating a pointer to an L2VPN or
    by specifying a layer 2 tunnel service.";
  container l2-tunnel-service {
    description
      "Defines a layer 2 tunnel termination.
      It is only applicable when a tunnel is
```

```
required. The supported values are:
pseudowire, VPLS, and VXLAN. Other
values may be defined, if needed.";
leaf type {
  type identityref {
    base l2-tunnel-type;
  }
  description
    "Selects the tunnel termination option for
    each vpn-network-access.";
}
container pseudowire {
  when "derived-from-or-self(..../type, "
    + "'pseudowire')" {
    description
      "Only applies when the type of the layer 2
      service type is pseudowire .";
  }
  description
    "Includes pseudowire termination parameters.";
  leaf vcid {
    type uint32;
    description
      "Indicates a PW or VC identifier.";
  }
  leaf far-end {
    type union {
      type uint32;
      type inet:ip-address;
    }
    description
      "Neighbor reference.";
    reference
      "RFC 8077: Pseudowire Setup and Maintenance
      Using the Label Distribution
      Protocol (LDP), Section 6.1";
  }
}
container vpls {
  when "derived-from-or-self(..../type, "
    + "'vpls')" {
    description
      "Only applies when the type of the layer 2
      service type is VPLS.";
  }
  description
    "VPLS termination parameters.";
  leaf vcid {
```

```
        type uint32;
        description
            "VC Identifier.";
    }
    leaf-list far-end {
        type union {
            type uint32;
            type inet:ip-address;
        }
        description
            "Neighbor reference.";
    }
}
container vxlan {
    when "derived-from-or-self(..../type, "
        + "'vxlan')" {
        description
            "Only applies when the type of the layer 2
            service type is VXLAN.";
    }
    description
        "VXLAN termination parameters.";
    leaf vni-id {
        type uint32;
        mandatory true;
        description
            "VXLAN Network Identifier (VNI).";
    }
    leaf peer-mode {
        type identityref {
            base vpn-common:vxlan-peer-mode;
        }
        default "vpn-common:static-mode";
        description
            "Specifies the VXLAN access mode. By
            default, the peer mode is set to
            'static-mode'.";
    }
    leaf-list peer-ip-address {
        type inet:ip-address;
        description
            "List of peer's IP addresses.";
    }
}
}
case l2vpn {
    leaf l2vpn-id {
        type vpn-common:vpn-id;
    }
}
```

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



```
"Defines how addresses are allocated to the
peer site.

If there is no value for the address
allocation type, then IPv4 addressing is not
enabled.";
}
choice allocation-type {
  description
  "Choice of the IPv4 address allocation.";
  case provider-dhcp {
    description
    "DHCP allocated addresses related
    parameters. IP addresses are allocated
    by DHCP that is operated by the provider";
    leaf dhcp-service-type {
      type enumeration {
        enum server {
          description
            "Local DHCP server.";
        }
        enum relay {
          description
            "Local DHCP relay. DHCP requests are
            relayed to a provider's server.";
        }
      }
      description
      "Indicates the type of DHCP service to
      be enabled on this access.";
    }
  }
  choice service-type {
    description
    "Choice based on the DHCP service type.";
    case relay {
      description
      "Container for list of provider's DHCP
      servers (i.e., dhcp-service-type is set
      to relay).";
      leaf-list server-ip-address {
        type inet:ipv4-address;
        description
        "IPv4 addresses of the provider's DHCP
        server to use by the local DHCP
        relay.";
      }
    }
    case server {
```

```
description
  "A choice about how addresses are assigned
  when a local DHCP server is enabled.";
choice address-assign {
  default "number";
  description
    "Choice for how IPv4 addresses are
    assigned.";
  case number {
    leaf number-of-dynamic-address {
      type uint16;
      default "1";
      description
        "Specifies the number of IP
        addresses to be assigned to the
        customer on this access.";
    }
  }
  case explicit {
    container customer-addresses {
      description
        "Container for customer
        addresses to be allocated
        using DHCP.";
      list address-pool {
        key "pool-id";
        description
          "Describes IP addresses to be
          allocated by DHCP.

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

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



```
+ "cation-type, 'provider-dhcp-slaac')" {
  description
    "Only applies when addresses are
    allocated by DHCPv6 provided by the
    operator.";
}
description
  "DHCPv6 allocated addresses related
  parameters.";
leaf dhcp-service-type {
  type enumeration {
    enum server {
      description
        "Local DHCPv6 server.";
    }
    enum relay {
      description
        "DHCPv6 relay.";
    }
  }
  description
    "Indicates the type of the DHCPv6 service to
    be enabled on this access.";
}
choice service-type {
  description
    "Choice based on the DHCPv6 service type.";
  case relay {
    leaf-list server-ip-address {
      type inet:ipv6-address;
      description
        "IPv6 addresses of the provider's
        DHCPv6 server.";
    }
  }
  case server {
    choice address-assign {
      default "number";
      description
        "Choice about how IPv6 prefixes are
        assigned by the DHCPv6 server.";
      case number {
        leaf number-of-dynamic-address {
          type uint16;
          default "1";
          description
            "Describes the number of IPv6
            prefixes that are allocated to
```



```
key "id";
description
  "List of routing protocols used on
  the CE/PE link. This list can be augmented.";
leaf id {
  type string;
  description
    "Unique identifier for routing protocol.";
}
leaf type {
  type identityref {
    base vpn-common:routing-protocol-type;
  }
  description
    "Type of routing protocol.";
}
list routing-profiles {
  key "id";
  description
    "Routing profiles.";
  leaf id {
    type leafref {
      path "/l3vpn-ntw/vpn-profiles"
        + "/valid-provider-identifiers"
        + "/routing-profile-identifier/id";
    }
    description
      "Routing profile to be used.";
  }
  leaf type {
    type identityref {
      base vpn-common:ie-type;
    }
    description
      "Import, export, or both.";
  }
}
container static {
  when "derived-from-or-self(..type, "
    + "'vpn-common:static-routing') " {
    description
      "Only applies when protocol is static.";
  }
  description
    "Configuration specific to static routing.";
  container cascaded-lan-prefixes {
    description
      "LAN prefixes from the customer.";
  }
}
```

```
list ipv4-lan-prefixes {
  if-feature "vpn-common:ipv4";
  key "lan next-hop";
  description
    "List of LAN prefixes for the site.";
  leaf lan {
    type inet:ipv4-prefix;
    description
      "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
      "Internal tag to be used in VPN
      policies.";
  }
  leaf next-hop {
    type union {
      type inet:ip-address;
      type predefined-next-hop;
    }
    description
      "The next-hop that is to be used
      for the static route. This may be
      specified as an IP address or a
      pre-defined next-hop type (e.g.,
      discard or local-link).";
  }
  leaf bfd-enable {
    if-feature "vpn-common:bfd";
    type boolean;
    description
      "Enables BFD.";
  }
  leaf metric {
    type uint32;
    description
      "Indicates the metric associated with
      the static route.";
  }
  leaf preference {
    type uint32;
    description
      "Indicates the preference of the static
      routes.";
  }
  uses vpn-common:service-status;
}
```

```
list ipv6-lan-prefixes {
  if-feature "vpn-common:ipv6";
  key "lan next-hop";
  description
    "List of LAN prefixes for the site.";
  leaf lan {
    type inet:ipv6-prefix;
    description
      "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
      "Internal tag to be used in VPN
      policies.";
  }
  leaf next-hop {
    type union {
      type inet:ip-address;
      type predefined-next-hop;
    }
    description
      "The next-hop that is to be used for the
      static route. This may be specified as
      an IP address or a pre-defined next-hop
      type (e.g., discard or local-link).";
  }
  leaf bfd-enable {
    if-feature "vpn-common:bfd";
    type boolean;
    description
      "Enables BFD.";
  }
  leaf metric {
    type uint32;
    description
      "Indicates the metric associated with
      the static route.";
  }
  leaf preference {
    type uint32;
    description
      "Indicates the preference associated
      with the static route.";
  }
  uses vpn-common:service-status;
}
}
```

```
}
container bgp {
  when "derived-from-or-self(..../type, "
    + "'vpn-common:bgp-routing')" {
    description
      "Only applies when protocol is BGP.";
  }
  description
    "BGP-specific configuration.";
  leaf description {
    type string;
    description
      "Includes a description of the BGP session.

      This description is meant to be used for
      diagnosis purposes. The semantic of the
      description is local to an
      implementation.";
  }
  leaf local-as {
    type inet:as-number;
    description
      "Indicates a local AS Number (ASN) if a
      distinct ASN than the one configured at
      the VPN node level is needed.";
  }
  leaf peer-as {
    type inet:as-number;
    mandatory true;
    description
      "Indicates the customer's ASN when
      the customer requests BGP routing.";
  }
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "This node contains the address families to be
      activated. Dual-stack means that both IPv4
      and IPv6 will be activated.";
  }
  leaf local-address {
    type union {
      type inet:ip-address;
      type if:interface-ref;
    }
    description

```

```
        "Set the local IP address to use for the BGP
        transport session. This may be expressed as
        either an IP address or a reference to an
        interface.";
    }
    leaf-list neighbor {
        type inet:ip-address;
        description
            "IP address(es) of the BGP neighbor. IPv4
            and IPv6 neighbors may be indicated if
            two sessions will be used for IPv4 and
            IPv6.";
    }
    leaf multihop {
        type uint8;
        description
            "Describes the number of IP hops allowed
            between a given BGP neighbor and the PE.";
    }
    leaf as-override {
        type boolean;
        default "false";
        description
            "Defines whether ASN override is enabled,
            i.e., replace the ASN of the customer
            specified in the AS_Path attribute with
            the local ASN.";
    }
    leaf allow-own-as {
        type uint8;
        default "0";
        description
            "Specifies the number of occurrences
            of the provider's ASN that can occur
            within the AS_PATH before it
            is rejected.";
    }
    leaf prepend-global-as {
        type boolean;
        default "false";
        description
            "In some situations, the ASN that is
            provided at the VPN node level may be
            distinct from the one configured at the
            VPN network access level. When such
            ASNs are provided, they are both
            prepended to the BGP route updates
            for this access. To disable that
```

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

```
list redistribute-connected {
  key "address-family";
  description
    "Indicates the per-AF policy to follow
    for connected routes.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates the address family.";
  }
  leaf enable {
    type boolean;
    description
      "Enables to redistribute connected
      routes.";
  }
}
container bgp-max-prefix {
  description
    "Controls the behavior when a prefix
    maximum is reached.";
  leaf max-prefix {
    type uint32;
    default "5000";
    description
      "Indicates the maximum number of BGP
      prefixes allowed in the BGP session.

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

      If the limit is exceeded, the action
      indicated in violate-action will be
      followed.";
    reference
      "RFC 4271: A Border Gateway Protocol 4
      (BGP-4), Section 8.2.2";
  }
  leaf warning-threshold {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    units "percent";
    default "75";
    description
```

```
        "When this value is reached, a warning
          notification will be triggered.";
    }
    leaf violate-action {
      type enumeration {
        enum warning {
          description
            "Only a warning message is sent to
             the peer when the limit is
             exceeded.";
        }
        enum discard-extra-paths {
          description
            "Discards extra paths when the
             limit is exceeded.";
        }
        enum restart {
          description
            "The BGP session restarts after
             a time interval.";
        }
      }
    }
    description
      "BGP neighbor max-prefix violate
       action.";
  }
  leaf restart-timer {
    type uint32;
    units "seconds";
    description
      "Time interval after which the BGP
       session will be reestablished.";
  }
}
container bgp-timers {
  description
    "Includes two BGP timers that can be
     customized when building a VPN service
     with BGP used as CE-PE routing
     protocol.";
  leaf keepalive {
    type uint16 {
      range "0..21845";
    }
    units "seconds";
    default "30";
    description
      "This timer indicates the KEEPALIVE
```

messages' frequency between a PE and a BGP peer.

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

It is suggested that the maximum time between KEEPALIVE messages would be one third of the Hold Time interval.";

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

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

```
choice option {
  description
    "Choice of authentication options.";
  case ao {
    description
      "Uses TCP-Authentication Option
      (TCP-AO).";
    reference
      "RFC 5925: The TCP Authentication
      Option.";
    leaf enable-ao {
      type boolean;
      description
        "Enables TCP-AO.";
    }
    leaf ao-keychain {
      type key-chain:key-chain-ref;
      description
        "Reference to the TCP-AO key chain.";
      reference
        "RFC 8177: YANG Key Chain.";
    }
  }
  case md5 {
    description
      "Uses MD5 to secure the session.";
    reference
      "RFC 4364: BGP/MPLS IP Virtual Private
      Networks (VPNs),
      Section 13.2";
    leaf md5-keychain {
      type key-chain:key-chain-ref;
      description
        "Reference to the MD5 key chain.";
      reference
        "RFC 8177: YANG Key Chain";
    }
  }
  case explicit {
    leaf key-id {
      type uint32;
      description
        "Key Identifier.";
    }
    leaf key {
      type string;
      description
        "BGP authentication key.
```

```
        This model only supports the subset
        of keys that are representable as
        ASCII strings.";
    }
    leaf crypto-algorithm {
        type identityref {
            base key-chain:crypto-algorithm;
        }
        description
            "Indicates the cryptographic algorithm
            associated with the key.";
    }
}
case ipsec {
    description
        "Specifies a reference to an IKE
        Security Association (SA).";
    leaf sa {
        type string;
        description
            "Indicates the administrator-assigned
            name of the SA.";
    }
}
}
}
}
uses vpn-common:service-status;
}
container ospf {
    when "derived-from-or-self(..type, "
        + "'vpn-common:ospf-routing')" {
        description
            "Only applies when protocol is OSPF.";
    }
    description
        "OSPF-specific configuration.";
    leaf address-family {
        type identityref {
            base vpn-common:address-family;
        }
        description
            "Indicates whether IPv4, IPv6, or
            both are to be activated.";
    }
    leaf area-id {
        type yang:dotted-quad;
        mandatory true;
    }
}
```

```
description
  "Area ID.";
reference
  "RFC 4577: OSPF as the Provider/Customer
  Edge Protocol for BGP/MPLS IP
  Virtual Private Networks
  (VPNs), Section 4.2.3
  RFC 6565: OSPFv3 as a Provider Edge to
  Customer Edge (PE-CE) Routing
  Protocol, Section 4.2";
}
leaf metric {
  type uint16;
  default "1";
  description
    "Metric of the PE-CE link. It is used
    in the routing state calculation and
    path selection.";
}
container sham-links {
  if-feature "vpn-common:rtg-ospf-sham-link";
  description
    "List of sham links.";
  reference
    "RFC 4577: OSPF as the Provider/Customer
    Edge Protocol for BGP/MPLS IP
    Virtual Private Networks
    (VPNs), Section 4.2.7
    RFC 6565: OSPFv3 as a Provider Edge to
    Customer Edge (PE-CE) Routing
    Protocol, Section 5";
  list sham-link {
    key "target-site";
    description
      "Creates a sham link with another site.";
    leaf target-site {
      type string;
      description
        "Target site for the sham link connection.
        The site is referred to by its
        identifier.";
    }
    leaf metric {
      type uint16;
      default "1";
      description
        "Metric of the sham link. It is used in
        the routing state calculation and path
```

```

        selection. The default value is set
        to 1.";
reference
  "RFC 4577: OSPF as the Provider/Customer
  Edge Protocol for BGP/MPLS IP
  Virtual Private Networks
  (VPNs), Section 4.2.7.3
  RFC 6565: OSPFv3 as a Provider Edge to
  Customer Edge (PE-CE) Routing
  Protocol, Section 5.2";
    }
  }
}
leaf max-lsa {
  type uint32 {
    range "1..4294967294";
  }
  description
    "Maximum number of allowed LSAs OSPF.";
}
container authentication {
  description
    "Authentication configuration.";
  leaf enable {
    type boolean;
    default "false";
    description
      "Enables or disables authentication.";
  }
}
container keying-material {
  when "../enable = 'true'";
  description
    "Container for describing how an OSPF
    session is to be secured between a CE
    and a PE.";
  choice option {
    description
      "Options for OSPF authentication.";
    case auth-key-chain {
      leaf key-chain {
        type key-chain:key-chain-ref;
        description
          "key-chain name.";
      }
    }
    case auth-key-explicit {
      leaf key-id {
        type uint32;
      }
    }
  }
}

```

```
        description
            "Key identifier.";
    }
    leaf key {
        type string;
        description
            "OSPF authentication key.
            This model only supports the subset
            of keys that are representable as
            ASCII strings.";
    }
    leaf crypto-algorithm {
        type identityref {
            base key-chain:crypto-algorithm;
        }
        description
            "Indicates the cryptographic algorithm
            associated with the key.";
    }
}
case ipsec {
    leaf sa {
        type string;
        description
            "Indicates the administrator-assigned
            name of the SA.";
        reference
            "RFC 4552: Authentication
            /Confidentiality for
            OSPFv3";
    }
}
}
}
}
}
uses vpn-common:service-status;
}
container isis {
    when "derived-from-or-self(..../type, "
        + "'vpn-common:isis-routing')" {
        description
            "Only applies when protocol is IS-IS.";
    }
    description
        "IS-IS specific configuration.";
    leaf address-family {
        type identityref {
            base vpn-common:address-family;
        }
    }
}
```

```
    }
    description
      "Indicates whether IPv4, IPv6, or both
      are to be activated.";
  }
  leaf area-address {
    type area-address;
    mandatory true;
    description
      "Area address.";
  }
  leaf level {
    type identityref {
      base vpn-common:isis-level;
    }
    description
      "Can be level-1, level-2, or level-1-2.";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the PE-CE link. It is used
      in the routing state calculation and
      path selection.";
  }
  leaf mode {
    type enumeration {
      enum active {
        description
          "Interface sends or receives IS-IS
          protocol control packets.";
      }
      enum passive {
        description
          "Suppresses the sending of IS-IS
          updates through the specified
          interface.";
      }
    }
    default "active";
    description
      "IS-IS interface mode type.";
  }
  container authentication {
    description
      "Authentication configuration.";
    leaf enable {
```

```
type boolean;
default "false";
description
  "Enables or disables authentication.";
}
container keying-material {
  when "../enable = 'true'";
  description
    "Container for describing how an IS-IS
    session is to be secured between a CE
    and a PE.";
  choice option {
    description
      "Options for IS-IS authentication.";
    case auth-key-chain {
      leaf key-chain {
        type key-chain:key-chain-ref;
        description
          "key-chain name.";
      }
    }
    case auth-key-explicit {
      leaf key-id {
        type uint32;
        description
          "Key Identifier.";
      }
      leaf key {
        type string;
        description
          "IS-IS authentication key.
          This model only supports the subset
          of keys that are representable as
          ASCII strings.";
      }
      leaf crypto-algorithm {
        type identityref {
          base key-chain:crypto-algorithm;
        }
        description
          "Indicates the cryptographic algorithm
          associated with the key.";
      }
    }
  }
}
}
}
uses vpn-common:service-status;
```

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

```
leaf holddown-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "180";
  description
    "Specifies the interval before better routes
     are released.";
}
leaf flush-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "240";
  description
    "Indicates the RIP flush timer. That is,
     the amount of time that must elapse before
     a route is removed from the routing
     table.";
}
}
leaf default-metric {
  type uint8 {
    range "0..16";
  }
  default "1";
  description
    "Sets the default metric.";
}
container authentication {
  description
    "Authentication configuration.";
  leaf enable {
    type boolean;
    default "false";
    description
      "Enables or disables authentication.";
  }
}
container keying-material {
  when "../enable = 'true'";
  description
    "Container for describing how a RIP
     session is to be secured between a CE
     and a PE.";
  choice option {
    description
```



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

```
BFD is set as a fault detection mechanism,
but other mechanisms can be defined in the
future.";
container bfd {
  if-feature "vpn-common:bfd";
  description
    "Container for BFD.";
  leaf session-type {
    type identityref {
      base vpn-common:bfd-session-type;
    }
    default "vpn-common:classic-bfd";
    description
      "Specifies the BFD session type.";
  }
  leaf desired-min-tx-interval {
    type uint32;
    units "microseconds";
    default "1000000";
    description
      "The minimum interval between transmission of
      BFD control packets that the operator
      desires.";
    reference
      "RFC 5880: Bidirectional Forwarding Detection
      (BFD), Section 6.8.7";
  }
  leaf required-min-rx-interval {
    type uint32;
    units "microseconds";
    default "1000000";
    description
      "The minimum interval between received BFD
      control packets that the PE should support.";
    reference
      "RFC 5880: Bidirectional Forwarding Detection
      (BFD), Section 6.8.7";
  }
  leaf local-multiplier {
    type uint8 {
      range "1..255";
    }
    default "3";
    description
      "Specifies the detection multiplier that is
      transmitted to a BFD peer.

      The detection interval for the receiving
```

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

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

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

    The provider can propose some profiles
    to the customer, depending on the
    service level the customer wants to
    achieve.";
}
container authentication {
  presence "Enables BFD authentication";
  description
    "Parameters for BFD authentication.";
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
      "Name of the key-chain.";
  }
}
```

```
        leaf meticulous {
            type boolean;
            description
                "Enables meticulous mode.";
            reference
                "RFC 5880: Bidirectional Forwarding
                Detection (BFD), Section 6.7";
        }
    }
    uses vpn-common:service-status;
}
container security {
    description
        "Site-specific security parameters.";
    container encryption {
        if-feature "vpn-common:encryption";
        description
            "Container for CE-PE security encryption.";
        leaf enabled {
            type boolean;
            default "false";
            description
                "If true, traffic encryption on the
                connection is required. Otherwise, it
                is disabled.";
        }
        leaf layer {
            when "../enabled = 'true'" {
                description
                    "It is included only when encryption
                    is enabled.";
            }
            type enumeration {
                enum layer2 {
                    description
                        "Encryption occurs at Layer 2.";
                }
                enum layer3 {
                    description
                        "Encryption occurs at Layer 3.
                        For example, IPsec may be used when
                        a customer requests Layer 3
                        encryption.";
                }
            }
        }
        description
            "Indicates the layer on which encryption
```

```
        is applied.";
    }
}
container encryption-profile {
    when "../encryption/enabled = 'true'" {
        description
            "Indicates the layer on which encryption
            is enabled.";
    }
    description
        "Container for encryption profile.";
    choice profile {
        description
            "Choice for the encryption profile.";
        case provider-profile {
            leaf profile-name {
                type leafref {
                    path "/l3vpn-ntw/vpn-profiles"
                        + "/valid-provider-identifiers"
                        + "/encryption-profile-identifier/id";
                }
                description
                    "Name of the service provider's profile
                    to be applied.";
            }
        }
        case customer-profile {
            leaf customer-key-chain {
                type key-chain:key-chain-ref;
                description
                    "Customer-supplied key chain.";
            }
        }
    }
}
container service {
    description
        "Service parameters of the attachment.";
    leaf inbound-bandwidth {
        if-feature "vpn-common:inbound-bw";
        type uint64;
        units "bps";
        description
            "From the customer site's perspective, the
            service inbound bandwidth of the connection
            or download bandwidth from the SP to
            the site. Note that the L3SM uses 'input-
```

```
        -bandwidth' to refer to the same concept.";
    }
    leaf outbound-bandwidth {
        if-feature "vpn-common:outbound-bw";
        type uint64;
        units "bps";
        description
            "From the customer site's perspective,
            the service outbound bandwidth of the
            connection or upload bandwidth from
            the site to the SP. Note that the L3SM uses
            'output-bandwidth' to refer to the same
            concept.";
    }
    leaf mtu {
        type uint32;
        units "bytes";
        description
            "MTU at service level. If the service is IP,
            it refers to the IP MTU. If Carriers'
            Carriers (CsC) is enabled, the requested MTU
            will refer to the MPLS maximum labeled packet
            size and not to the IP MTU.";
    }
    container qos {
        if-feature "vpn-common:qos";
        description
            "QoS configuration.";
        container qos-classification-policy {
            description
                "Configuration of the traffic classification
                policy.";
            uses vpn-common:qos-classification-policy;
        }
        container qos-action {
            description
                "List of QoS action policies.";
            list rule {
                key "id";
                description
                    "List of QoS actions.";
                leaf id {
                    type string;
                    description
                        "An identifier of the QoS action rule.";
                }
                leaf target-class-id {
                    type string;
                }
            }
        }
    }
}
```

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

```
    }
    leaf direction {
      type identityref {
        base vpn-common:qos-profile-direction;
      }
      default "vpn-common:both";
      description
        "The direction to which the QoS profile
         is applied.";
    }
  }
}
container carriers-carrier {
  if-feature "vpn-common:carriers-carrier";
  description
    "This container is used when the customer
     provides MPLS-based services. This is
     only used in the case of CsC (i.e., a
     customer builds an MPLS service using an
     IP VPN to carry its traffic).";
  leaf signaling-type {
    type enumeration {
      enum ldp {
        description
          "Use LDP as the signaling protocol
           between the PE and the CE. In this
           case, an IGP routing protocol must
           also be configured.";
      }
      enum bgp {
        description
          "Use BGP as the signaling protocol
           between the PE and the CE.
           In this case, BGP must also be configured
           as the routing protocol.";
        reference
          "RFC 8277: Using BGP to Bind MPLS Labels
           to Address Prefixes";
      }
    }
    default "bgp";
    description
      "MPLS signaling type.";
  }
}
container ntp {
  description
```

```
    "Time synchronization may be needed in some
    VPNs such as infrastructure and Management
    VPNs. This container includes parameters to
    enable NTP service.";
reference
  "RFC 5905: Network Time Protocol Version 4:
  Protocol and Algorithms
  Specification";
leaf broadcast {
  type enumeration {
    enum client {
      description
        "The VPN node will listen to NTP broadcast
        messages on this VPN network access.";
    }
    enum server {
      description
        "The VPN node will behave as a broadcast
        server.";
    }
  }
  description
    "Indicates NTP broadcast mode to use for the
    VPN network access.";
}
container auth-profile {
  description
    "Pointer to a local profile.";
  leaf profile-id {
    type string;
    description
      "A pointer to a local authentication
      profile on the VPN node is provided.";
  }
}
uses vpn-common:service-status;
}
container multicast {
  if-feature "vpn-common:multicast";
  description
    "Multicast parameters for the network
    access.";
  leaf access-type {
    type enumeration {
      enum receiver-only {
        description
          "The peer site only has receivers.";
      }
    }
  }
}
```

```
enum source-only {
  description
    "The peer site only has sources.";
}
enum source-receiver {
  description
    "The peer site has both sources and
    receivers.";
}
}
default "source-receiver";
description
  "Type of multicast site.";
}
leaf address-family {
  type identityref {
    base vpn-common:address-family;
  }
  description
    "Indicates the address family.";
}
leaf protocol-type {
  type enumeration {
    enum host {
      description
        "Hosts are directly connected to the
        provider network.

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

        Typically, IGMP and PIM will be
        implemented.";
    }
  }
}
default "both";
```

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

```
leaf max-group-sources {
  type uint32;
  description
    "The maximum number of group sources.";
}
leaf version {
  type identityref {
    base vpn-common:igmp-version;
  }
  default "vpn-common:igmpv2";
  description
    "Version of the IGMP.";
}
uses vpn-common:service-status;
}
container mld {
  when "../protocol-type = 'host' and "
    + "../address-family = 'vpn-common:ipv6' or "
    + "'vpn-common:dual-stack'";
  if-feature "vpn-common:mld";
  description
    "Includes MLD-related parameters.";
  list static-group {
    key "group-addr";
    description
      "Multicast static source/group associated to
      the MLD session";
    leaf group-addr {
      type rt-types:ipv6-multicast-group-address;
      description
        "Multicast group IPv6 address.";
    }
    leaf source-addr {
      type rt-types:ipv6-multicast-source-address;
      description
        "Multicast source IPv6 address.";
    }
  }
}
leaf max-groups {
  type uint32;
  description
    "Indicates the maximum number of groups.";
}
leaf max-entries {
  type uint32;
  description
    "Indicates the maximum number of MLD
    entries.";
```

```
    }
    leaf max-group-sources {
      type uint32;
      description
        "The maximum number of group sources.";
    }
    leaf version {
      type identityref {
        base vpn-common:mld-version;
      }
      default "vpn-common:mldv2";
      description
        "Version of the MLD protocol.";
    }
    uses vpn-common:service-status;
  }
  container pim {
    when "../protocol-type = 'router'";
    if-feature "vpn-common:pim";
    description
      "Only applies when protocol type is PIM.";
    leaf hello-interval {
      type rt-types:timer-value-seconds16;
      default "30";
      description
        "PIM hello-messages interval. If set to
        'infinity' or 'not-set', no periodic
        Hello messages are sent.";
      reference
        "RFC 7761: Protocol Independent Multicast -
        Sparse Mode (PIM-SM): Protocol
        Specification (Revised),
        Section 4.11";
    }
    leaf dr-priority {
      type uint32;
      default "1";
      description
        "Indicates the preference in the DR election
        process. A larger value has a higher
        priority over a smaller value.";
      reference
        "RFC 7761: Protocol Independent Multicast -
        Sparse Mode (PIM-SM): Protocol
        Specification (Revised),
        Section 4.3.2";
    }
  }
  uses vpn-common:service-status;
```


QoS, bandwidth, routing protocols, keying material), leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create an L3VPN service or add a new network access. In addition to using NACM to prevent authorized access, such activity can be detected by adequately monitoring and tracking network configuration changes.

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

- * 'customer-name' and 'ip-connection': An attacker can retrieve privacy-related information which can be used to track a customer. Disclosing such information may be considered as a violation of the customer-provider trust relationship.
- * 'keying-material': An attacker can retrieve the cryptographic keys protecting the underlying VPN service (CE-PE routing, in particular). These keys could be used to inject spoofed routing advertisements.

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

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

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

10. IANA Considerations

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

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

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

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

11. References

11.1. Normative References

- [I-D.ietf-opsawg-vpn-common]
Barguil, S., Dios, O. G. D., Boucadair, M., and Q. Wu, "A Layer 2/3 VPN Common YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-vpn-common-11, 23 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-vpn-common-11.txt>>.
- [ISO10589] ISO, "Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)", 2002, <International Standard 10589:2002, Second Edition>.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <<https://www.rfc-editor.org/info/rfc1195>>.
- [RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <<https://www.rfc-editor.org/info/rfc2080>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, DOI 10.17487/RFC2236, November 1997, <<https://www.rfc-editor.org/info/rfc2236>>.
- [RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <<https://www.rfc-editor.org/info/rfc2453>>.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<https://www.rfc-editor.org/info/rfc2710>>.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <<https://www.rfc-editor.org/info/rfc4552>>.
- [RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, DOI 10.17487/RFC4577, June 2006, <<https://www.rfc-editor.org/info/rfc4577>>.
- [RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <<https://www.rfc-editor.org/info/rfc5308>>.

- [RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <<https://www.rfc-editor.org/info/rfc5701>>.
- [RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, DOI 10.17487/RFC5709, October 2009, <<https://www.rfc-editor.org/info/rfc5709>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<https://www.rfc-editor.org/info/rfc5798>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <<https://www.rfc-editor.org/info/rfc5925>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <<https://www.rfc-editor.org/info/rfc6513>>.

- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <<https://www.rfc-editor.org/info/rfc6514>>.
- [RFC6565] Pillay-Esnault, P., Moyer, P., Doyle, J., Ertekin, E., and M. Lundberg, "OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol", RFC 6565, DOI 10.17487/RFC6565, June 2012, <<https://www.rfc-editor.org/info/rfc6565>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7166] Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, DOI 10.17487/RFC7166, March 2014, <<https://www.rfc-editor.org/info/rfc7166>>.
- [RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <<https://www.rfc-editor.org/info/rfc7474>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <<https://www.rfc-editor.org/info/rfc8177>>.

- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.
- [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <<https://www.rfc-editor.org/info/rfc8519>>.

11.2. Informative References

- [I-D.evenwu-opsawg-yang-composed-vpn]
Even, R., Wu, B., Wu, Q., and YingCheng, "YANG Data Model for Composed VPN Service Delivery", Work in Progress, Internet-Draft, draft-evenwu-opsawg-yang-composed-vpn-03, 8 March 2019, <<https://www.ietf.org/archive/id/draft-evenwu-opsawg-yang-composed-vpn-03.txt>>.
- [I-D.ietf-bess-evpn-prefix-advertisement]
Rabadan, J., Henderickx, W., Drake, J. E., Lin, W., and A. Sajassi, "IP Prefix Advertisement in EVPN", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-prefix-advertisement-11, 18 May 2018, <<https://www.ietf.org/archive/id/draft-ietf-bess-evpn-prefix-advertisement-11.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-11, 11 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgp-model-11.txt>>.

[I-D.ietf-pim-yang]

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

[I-D.ietf-rtgwg-qos-model]

Choudhary, A., Jethanandani, M., Strahle, N., Aries, E., and I. Chen, "A YANG Data Model for Quality of Service (QoS)", Work in Progress, Internet-Draft, draft-ietf-rtgwg-qos-model-04, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-rtgwg-qos-model-04.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-08, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-08.txt>>.

[I-D.ietf-teas-ietf-network-slices]

Farrel, A., Gray, E., 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-04, 23 August 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slices-04.txt>>.

[I-D.ogondio-opsawg-uni-topology]

Dios, O. G. D., Barguil, S., Wu, Q., and M. Boucadair, "A YANG Model for User-Network Interface (UNI) Topologies", Work in Progress, Internet-Draft, draft-ogondio-opsawg-uni-topology-01, 2 April 2020, <<https://www.ietf.org/archive/id/draft-ogondio-opsawg-uni-topology-01.txt>>.

[IEEE802.1AX]

"Link Aggregation", IEEE Std 802.1AX-2020, 2020.

- [PYANG] "pyang", November 2020,
<<https://github.com/mbj4668/pyang>>.
- [RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <<https://www.rfc-editor.org/info/rfc3618>>.
- [RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <<https://www.rfc-editor.org/info/rfc3644>>.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <<https://www.rfc-editor.org/info/rfc4026>>.
- [RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <<https://www.rfc-editor.org/info/rfc4110>>.
- [RFC4176] El Mghazli, Y., Ed., Nadeau, T., Boucadair, M., Chan, K., and A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management", RFC 4176, DOI 10.17487/RFC4176, October 2005, <<https://www.rfc-editor.org/info/rfc4176>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC6037] Rosen, E., Ed., Cai, Y., Ed., and IJ. Wijnands, "Cisco Systems' Solution for Multicast in BGP/MPLS IP VPNs", RFC 6037, DOI 10.17487/RFC6037, October 2010, <<https://www.rfc-editor.org/info/rfc6037>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.

- [RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <<https://www.rfc-editor.org/info/rfc6952>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <<https://www.rfc-editor.org/info/rfc7297>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <<https://www.rfc-editor.org/info/rfc7880>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <<https://www.rfc-editor.org/info/rfc8077>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <<https://www.rfc-editor.org/info/rfc8453>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.
- [RFC8633] Reilly, D., Stenn, H., and D. Sibold, "Network Time Protocol Best Current Practices", BCP 223, RFC 8633, DOI 10.17487/RFC8633, July 2019, <<https://www.rfc-editor.org/info/rfc8633>>.
- [RFC8695] Liu, X., Sarda, P., and V. Choudhary, "A YANG Data Model for the Routing Information Protocol (RIP)", RFC 8695, DOI 10.17487/RFC8695, February 2020, <<https://www.rfc-editor.org/info/rfc8695>>.
- [RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.

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

Appendix A. L3VPN Examples

A.1. 4G VPN Provisioning Example

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

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

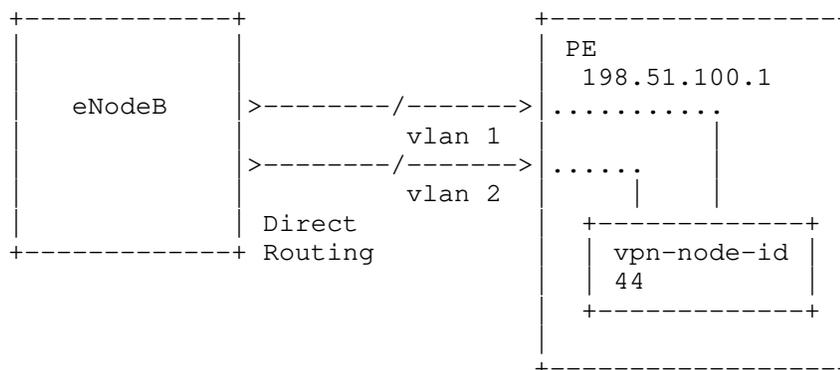


Figure 31: Mobile Backhaul Example

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

First: Create the 4G VPN service (Figure 32).

```

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

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

```

Figure 32: Create VPN Service

Second: Create a VPN node as depicted in Figure 33. In this type of service, the VPN node is equivalent to the VRF configured in the physical device ('ne-id'=198.51.100.1).

```

===== NOTE: '\ ' line wrapping per RFC 8792 =====
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G
Host: example.com
Content-Type: application/yang-data+json

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

```

Figure 33: Create VPN Node

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

```

===== NOTE: '\ ' line wrapping per RFC 8792 =====
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G/vpn-nodes/vpn-node=44
content-type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1",
        "description": "Interface SYNC to eNODE-B",
        "vpn-network-access-type": "ietf-vpn-common:point-to-point",
        "vpn-instance-profile": "simple-profile",
        "status": {

```

```
    "admin-status": {
      "status": "ietf-vpn-common:admin-up"
    }
  },
  "connection": {
    "encapsulation": {
      "type": "ietf-vpn-common:dot1q",
      "dot1q": {
        "cvlan-id": 1
      }
    }
  },
  "ip-connection": {
    "ipv4": {
      "local-address": "192.0.2.1",
      "prefix-length": 30,
      "address-allocation-type": "static-address",
      "static-addresses": {
        "primary-address": "1",
        "address": [
          {
            "address-id": "1",
            "customer-address": "192.0.2.2"
          }
        ]
      }
    },
    "ipv6": {
      "local-address": "2001:db8::1",
      "prefix-length": 64,
      "address-allocation-type": "static-address",
      "primary-address": "1",
      "address": [
        {
          "address-id": "1",
          "customer-address": "2001:db8::2"
        }
      ]
    }
  },
  "routing-protocols": {
    "routing-protocol": [
      {
        "id": "1",
        "type": "ietf-vpn-common:direct"
      }
    ]
  }
}
```

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

```

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

```

Figure 34: Create VPN Network Access

A.2. Loopback Interface

An example of loopback interface is depicted in Figure 35.

```

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

```

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

A.3. Overriding VPN Instance Profile Parameters

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

```
{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "override-example",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "HUB",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 64510,
              "rd-suffix": 1001,
              "address-family": [
                {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "maximum-routes": [
                    {
                      "protocol": "ietf-vpn-common:any",
                      "maximum-routes": 100
                    }
                  ]
                }
              ]
            }
          ]
        },
        {
          "profile-id": "SPOKE",
          "role": "ietf-vpn-common:spoke-role",
          "local-as": 64510,
          "address-family": [
            {
              "address-family": "ietf-vpn-common:dual-stack",
              "maximum-routes": [
                {
                  "protocol": "ietf-vpn-common:any",
                  "maximum-routes": 1000
                }
              ]
            }
          ]
        }
      ]
    }
  }
}
```



```
    ]
  }
]
}
},
{
  "vpn-node-id": "PE3",
  "ne-id": "pe3",
  "router-id": "198.51.100.3",
  "active-vpn-instance-profiles": {
    "vpn-instance-profile": [
      {
        "profile-id": "SPOKE"
      }
    ]
  }
},
{
  "vpn-node-id": "PE4",
  "ne-id": "pe4",
  "router-id": "198.51.100.4",
  "active-vpn-instance-profiles": {
    "vpn-instance-profile": [
      {
        "profile-id": "SPOKE"
      }
    ]
  }
}
]
}
}
]
}
}
```

Figure 36: VPN Instance Profile Example (Message Body)

A.4. Multicast VPN Provisioning Example

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

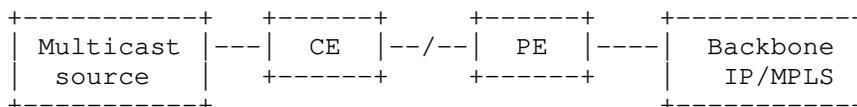


Figure 37: Multicast L3VPN Service Example

An example is provided below to illustrate how to configure a multicast L3VPN service using the L3NM.

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

```

{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "Multicast-IPTV",
        "vpn-description": "Multicast IPTV VPN service",
        "customer-name": "a-name",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "multicast",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 65536,
              "multicast": {
                "rp": {
                  "rp-group-mappings": {
                    "rp-group-mapping": [
                      {
                        "id": 1,
                        "rp-address": "203.0.113.17",
                        "groups": {
                          "group": [
                            {
                              "id": 1,
                              "group-address": "239.130.0.0/15"
                            }
                          ]
                        }
                      ]
                    }
                  }
                }
              }
            ]
          },
          "rp-discovery": {
            "rp-discovery-type": "ietf-vpn-common:static-rp"
          }
        }
      ]
    }
  ]
}

```

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

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

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

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

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

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

```

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

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

Appendix B. Implementation Status

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

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

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

B.1. Nokia Implementation

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

B.2. Huawei Implementation

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

B.3. Infinera Implementation

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

B.4. Ribbon-ECI Implementation

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

B.5. Juniper Implementation

<https://github.com/IETF-OPSAWG-WG/lxnm/blob/master/Implementattion/Juniper>

Acknowledgements

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

Daniel King, Daniel Voyer, Luay Jalil, and Stephane Litkowski contributed to early version of the individual submission. Many thanks to Robert Wilton for the AD review. Thanks to Andrew Malis for the routing directorate review, Rifaat Shekh-Yusef for the security directorate review, Qin Wu for the opsdire review, and Pete Resnick for the genart directorate review. Thanks to Michael Scharf for the discussion on TCP-AO. Thanks to Martin Duke, Lars Eagert, Zaheduzzaman Sarker, Roman Danyliw, Erik Kline, Benjamin Kaduk, Francesca Palombini, and Eric Vyncke for the IESG review.

This work was supported in part by the European Commission funded H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727) and Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (G.A. 101015857).

Contributors

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

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

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

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

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

Paul Sherratt
Gamma Telecom
Email: paul.sherratt@gamma.co.uk

Authors' Addresses

Samier Barguil
Telefonica
Madrid
Spain

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
Spain

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone
Spain

Email: luis-angel.munoz@vodafone.com

Alejandro Aguado
Nokia
Madrid
Spain

Email: alejandro.aguado_martin@nokia.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 19, 2020

E. Lear
Cisco Systems
S. Rose
NIST
May 18, 2020

SBOM Extension for MUD
draft-lear-opsawg-mud-sbom-00

Abstract

Software bills of materials (SBOMs) are formal descriptions of what pieces of software are included in a product. This memo specifies a means for manufacturers to state how SBOMs may be retrieved through an extension to manufacturer usage descriptions (MUD).

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 November 19, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	2
1.1.	How This Information Is Used	3
1.2.	SBOM formats	3
1.3.	Discussion points	3
2.	The mud-sbom extension model extension	4
3.	The mud-sbom augmentation to the MUD YANG model	4
4.	Examples	7
4.1.	Without ACLS	7
4.2.	Located on the Device	8
4.3.	SBOM Obtained from Contact Information	9
4.4.	With ACLS	9
5.	Security Considerations	12
6.	IANA Considerations	12
6.1.	MUD Extension	12
6.2.	Well-Known Prefix	12
7.	References	13
7.1.	Normative References	13
7.2.	Informative References	13
Appendix A.	Changes from Earlier Versions	13
Authors' Addresses	14

1. Introduction

Manufacturer Usage Descriptions (MUD) [RFC8520] provides a means for devices to identify what they are and what sort of network access they need. This memo specifies a YANG model [RFC6991] for reporting and a means for transmitting the report, and appropriate extensions to the MUD file to indicate how to report and how often.

Software bills of material (SBOMs) are descriptions of what software, including versioning and dependencies, a device contains. There are different SBOM formats such as Software Package Data Exchange [SPDX] and Software Identity Tags [SWID].

This memo extends the MUD YANG schema to provide location information of an SBOM.

These SBOMs are typically found in one of three ways:

- o on devices themselves
- o on a web site (e.g., via URI)
- o through direct contact with the manufacturer.

Some devices will have interfaces that permit direct SBOM retrieval. Examples of these interfaces might be 'ssh' or an HTTP endpoint for retrieval. There may also be private interfaces as well.

When a web site is used, versioning information about the SBOM is implicit based on the MUD file.

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

1.1. How This Information Is Used

SBOMs are used for numerous purposes, including vulnerability assessment, license management, and inventory management. This memo provides means for either automated or semi-automated collection of that information. For devices that can output a MUD URL, the mechanism may be highly automated. For devices that have a MUD URL in either their documentation or within a QR code on a box, the mechanism is semi-automated (someone has to scan the QR code or enter the URL).

Note that SBOMs may change more frequently than access control requirements. A change to software does not necessarily mean a change to control channels that are used. Therefore, it is important to retrieve the MUD file as suggested by the manufacturer in the cache-validity period. In many cases, only the SBOM list will have been updated.

1.2. SBOM formats

There are multiple ways to express an SBOM. When these are retrieved either directly from the device or directly from a web server, tools will need to observe the content-type header to determine precisely which format is being transmitted. Because IoT devices in particular have limited capabilities, use of a specific Accept: header in HTTP or the Accept Option in CoAP is NOT RECOMMENDED. Instead, backend tooling MUST silently discard SBOM information sent with a media type that is not understood.

1.3. Discussion points

The following is discussion to be removed at time of RFC publication.

- o Is the model structured correctly?

- o Are there other retrieval mechanisms that need to be specified?
- o Do we need to be more specific in how to authenticate and retrieve SBOMs?
- o What are the implications if the MUD URL is an extension in a certificate (e.g. an IDevID cert)?

2. The mud-sbom extension model extension

We now formally define this extension. This is done in two parts. First, the extension name "sbom" is listed in the "extensions" array of the MUD file.

Second, the "mud" container is augmented with a list of SBOM sources.

This is done as follows:

```

module: ietf-mud-sbom
  augment /mud:mud:
    +--rw sboms* [version-info]
      +--rw version-info          string
      +--rw (sbom-type)?
        +--:(url)
          | +--rw sbom-url?      inet:uri
        +--:(local-uri)
          | +--rw sbom-local*    enumeration
        +--:(contact-info)
          +--rw contact-uri?    inet:uri

```

3. The mud-sbom augmentation to the MUD YANG model

```

<CODE BEGINS>file "ietf-mud-sbom@2020-03-06.yang"
module ietf-mud-sbom {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-sbom";
  prefix mud-sbom;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-mud {
    prefix mud;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact

```

```
"WG
  Web: http://tools.ietf.org/wg/opsawg/
  WG List: opsawg@ietf.org
  Author: Eliot Lear lear@cisco.com ";
description
  "This YANG module augments the ietf-mud model to provide for
  reporting of SBOMs.

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

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

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

  The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
  NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
  'MAY', and 'OPTIONAL' in this document are to be interpreted as
  described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
  they appear in all capitals, as shown here. ";

revision 2020-03-06 {
  description
    "Initial proposed standard.";
  reference
    "RFC XXXX: Extension for MUD Reporting";
}

grouping mud-sbom-extension {
  description
    "SBOM extension grouping";
  list sboms {
    key "version-info";
    leaf version-info {
      type string;
      description
        "A version string that is applicable for this SBOM list entry.
        The format of this string is left to the device manufacturer.
        How the network administrator determines the version of
        software running on the device is beyond the scope of this
        memo.";
```

```
}
choice sbom-type {
  case url {
    leaf sbom-url {
      type inet:uri;
      description
        "A statically located URI.";
    }
  }
  case local-uri {
    leaf-list sbom-local {
      type enumeration {
        enum coap {
          description
            "Use COAP schema to retrieve SBOM";
        }
        enum coaps {
          description
            "Use COAPS schema to retrieve SBOM";
        }
        enum http {
          description
            "Use HTTP schema to retrieve SBOM";
        }
        enum https {
          description
            "Use HTTPS schema to retrieve SBOM";
        }
      }
    }
    description
      "The choice of sbom-local means that the SBOM resides at
      a location indicated by an indicted scheme for the
      device in question, at well known location
      './.well-known/sbom'. For example, if the MUD file
      indicates that coaps is to be used and the host is
      located at address 10.1.2.3, the SBOM could be retrieved
      at 'coaps://10.1.2.3/.well-known/sbom'. N.B., coap and
      http schemes are NOT RECOMMENDED.";
  }
}
case contact-info {
  leaf contact-uri {
    type inet:uri;
    description
      "This MUST be either a tel, http, https, or
      mailto uri schema that customers can use to
      contact someone for SBOM information.";
  }
}
```

```
    }
    description
      "choices for SBOM retrieval.";
  }
  description
    "list of methods to get an SBOM.";
}

augment "/mud:mud" {
  description
    "Add extension for SBOMs.";
  uses mud-sbom-extension;
}
}
```

<CODE ENDS>

4. Examples

In this example MUD file that uses a cloud service, the Frobinator presents a location of the SBOM in a URL. Note, the ACLs in a MUD file are NOT required, although they are a very good idea for IP-based devices. The first MUD file demonstrates how to get the SBOM without ACLs, and the second has ACLs.

4.1. Without ACLS

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://iot-device.example.com/dnsname",
    "last-update": "2019-01-15T10:22:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "device that wants to talk to a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://frobinator.example.com/doc/frob2000",
    "model-name": "Frobinator 2000",
    "extensions" : [
      "sbom"
    ],
    "sboms" : [
      {
        "version-info" : "FrobOS Release 1.1",
        "sbom-url" : "https://frobinator.example.com/sboms/f20001.1",
      }
    ]
  }
}
```

4.2. Located on the Device

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://iot-device.example.com/dnsname",
    "last-update": "2019-01-15T10:22:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "device that wants to talk to a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://frobinator.example.com/doc/frob2000",
    "model-name": "Frobinator 2000",
    "extensions" : [
      "sbom"
    ],
    "sboms" : [
      {
        "version-info" : "FrobOS Release 1.1",
        "sbom-local" : "coaps:///.well-known/sbom",
      }
    ]
  }
}
```

4.3. SBOM Obtained from Contact Information

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://iot-device.example.com/dnsname",
    "last-update": "2019-01-15T10:22:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "device that wants to talk to a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://frobinator.example.com/doc/frob2000",
    "model-name": "Frobinator 2000",
    "extensions" : [
      "sbom"
    ],
    "sboms" : [
      {
        "version-info" : "FrobOS Release 1.1",
        "contact-uri" : "mailto:sbom-requst@example.com",
      }
    ]
  }
}
```

4.4. With ACLS

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://iot-device.example.com/dnsname",
    "last-update": "2019-01-15T10:22:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "device that wants to talk to a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://frobinator.example.com/doc/frob2000",
    "model-name": "Frobinator 2000",
    "extensions" : [
      "sbom"
    ],
    "sboms" : [
      {
        "version-info" : "FrobOS Release 1.1",
        "sbom-url" : "https://frobinator.example.com/sboms/f20001.1",
      }
    ],
    "from-device-policy": {
```

```
"access-lists": {
  "access-list": [
    {
      "name": "mud-96898-v4fr"
    },
    {
      "name": "mud-96898-v6fr"
    }
  ]
},
"to-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-96898-v4to"
      },
      {
        "name": "mud-96898-v6to"
      }
    ]
  }
},
"ietf-access-control-list:acls": {
  "acl": [
    {
      "name": "mud-96898-v4to",
      "type": "ipv4-acl-type",
      "aces": {
        "ace": [
          {
            "name": "cl0-todev",
            "matches": {
              "ipv4": {
                "ietf-acl:src-dnsname": "cloud-service.example.com"
              }
            },
            "actions": {
              "forwarding": "accept"
            }
          }
        ]
      }
    }
  ]
},
{
  "name": "mud-96898-v4fr",
  "type": "ipv4-acl-type",
```

```
"aces": {
  "ace": [
    {
      "name": "cl0-frdev",
      "matches": {
        "ipv4": {
          "ietf-acldns:dst-dnsname": "cloud-service.example.com"
        }
      },
      "actions": {
        "forwarding": "accept"
      }
    }
  ]
},
{
  "name": "mud-96898-v6to",
  "type": "ipv6-acl-type",
  "aces": {
    "ace": [
      {
        "name": "cl0-todev",
        "matches": {
          "ipv6": {
            "ietf-acldns:src-dnsname": "cloud-service.example.com"
          }
        },
        "actions": {
          "forwarding": "accept"
        }
      }
    ]
  }
},
{
  "name": "mud-96898-v6fr",
  "type": "ipv6-acl-type",
  "aces": {
    "ace": [
      {
        "name": "cl0-frdev",
        "matches": {
          "ipv6": {
            "ietf-acldns:dst-dnsname": "cloud-service.example.com"
          }
        },
        "actions": {
```

```
        "forwarding": "accept"
      }
    }
  ]
}
}
```

At this point, the management system can attempt to retrieve the SBOM, and determine which format is in use through the content-type header on the response to a GET request.

5. Security Considerations

SBOMs provide an inventory of software. If firmware is available to an attacker, the attacker may well already be able to derive this very same software inventory. Manufacturers MAY restrict access to SBOM information using appropriate authorization semantics within HTTP. In particular, if a system attempts to retrieve an SBOM via HTTP, if the client is not authorized, the server MUST produce an appropriate error, with instructions on how to register a particular client. One example may be to issue a certificate to the client for this purpose after a registration process has taken place. Another example would involve the use of OAUTH in combination with a federations of SBOM servers.

To further mitigate attacks against a device, manufacturers SHOULD recommend access controls through the normal MUD mechanism.

6. IANA Considerations

6.1. MUD Extension

The IANA is requested to add "controller-candidate" to the MUD extensions registry as follows:

```
Extension Name: sbom
Standard reference: This document
```

6.2. Well-Known Prefix

The following well known URI is requested in accordance with [RFC8615]:

URI suffix: "sbom"
Change controller: "IETF"
Specification document: This memo
Related information: See ISO/IEC 19970-2 and SPDX.org

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

7.2. Informative References

- [SPDX] The Linux Foundation, "SPDX Specification 2.1", 2016.
- [SWID] ISO/IEC, "Information technology -- IT asset management -- Part 2: Software identification tag", ISO 19770-2:2015, 2015.

Appendix A. Changes from Earlier Versions

Draft -00:

- o Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Scott Rose
NIST
100 Bureau Dr
Gaithersburg MD 20899
USA

Phone: +1 301-975-8439
Email: scott.rose@nist.gov

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2021

D. Li
J. Wu
Tsinghua
Y. Gu
Huawei
L. Qin
Tsinghua
T. Lin
H3C
July 12, 2020

Source Address Validation Architecture (SAVA): Intra-domain Use Cases
draft-li-sava-intra-domain-use-cases-00

Abstract

This document identifies scenarios where existing approaches for detection and mitigation of source address spoofing don't perform perfectly. Either Ingress ACL filtering [RFC3704], unicast Reverse Path Forwarding (uRPF) [RFC3704], feasible path uRPF [RFC 3704], or Enhanced Feasible-Path uRPF [RFC8704] has limitations regarding either automated implementation objective or detection accuracy objective (0% false positive and 0% false negative). This document identifies two such scenarios and provides solution discussions.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 January 13, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Source Address Validation	2
1.2. Existing SAV Techniques Overview	3
1.3. SAV Requirements and Challenges	4
2. Terminology	6
3. Problem Statement	7
3.1. SAVA Intra-domain Use Case 1: Intra-AS Multi-homing	7
3.2. SAVA Intra-domain Use Case 2: Inter-AS Multihoming	8
4. Solution Consideration	10
5. Security Considerations	11
6. Contributors	11
7. Acknowledgments	11
8. Normative References	11
Authors' Addresses	14

1. Introduction

1.1. Source Address Validation

The Internet is open to traffic, which means that a sender can generate traffic and send to any receiver in the Internet without permission of the receiver. Although this openness design improves the scalability of the Internet, it also leaves security risks, namely, a sender can forge his/her source address when sending the packets, which is also well known as source address spoofing.

Due to the lack of source address spoofing detection mechanism, Denial of Service (DoS) attacks seriously compromise network security. By employing source address spoofing, attackers can well hide themselves and pin the blame on the destination networks. Administrators often spend a lot of effort identifying attack packets

without being able to locate the attacker's true source address. In addition to DOS attacks, source address spoofing is also used in a multitude of ways. The threats of source address spoofing have been well documented in [RFC6959]. To briefly summarize, the possible attacks by source address spoofing includes single-packet attack, flood-based DoS, poisoning attack, spoof-based worm/malware propagation, reflective attack, accounting subversion, man-in-the-middle attack, third-party recon, etc.

1.2. Existing SAV Techniques Overview

Source address validation (SAV) verifies the authenticity of the packet's source address to detect and mitigate source address spoofing [RFC2827]. Source Address Validation Improvement (SAVI) method [RFC7039] implements SAV at a fine granularity of host-level IP address validation. The unicast Reverse Path Forwarding (uRPF) techniques (such as Strict uRPF, Feasible uRPF and Loose uRPF) [RFC3704] are particularly designed to perform SAV in the granularity of IP network. The Enhanced Feasible-Path Unicast Reverse Path Forwarding (EFP-uRPF) methods [RFC8704] further improve Feasible uRPF to reduce false positives in the case of inter-AS routing asymmetry and multihoming.

SAVI, typically performed at the access network, is enforced in switches, where the mapping relationship between an IP address and other "trust anchor" is maintained. A "trust anchor" can be link-layer information (such as MAC address), physical port of a switch to connect a host, etc. It enforces hosts to use legitimate IP source addresses. However, given numerous access networks managed by different operators, it is far away from practice for all the access networks to simultaneously deploy SAVI. Therefore, in order to mitigate the security risks raised by source address spoofing, SAV performed in network border routers is also necessary. Although it does not provide the same filtering granularity as SAVI does, it still helps the tracing of spoofing to a minimized network range.

Ingress ACLs [RFC2827], typically performed at the network border routers, is performed by manually maintaining a traffic filtering access list which contains acceptable source address for each interface. Only packets with a source address encompassed in the access list can be accepted. It strictly specifies the source address space of incoming packets. However, manual configuration brings scalability and reliability issues.

Strict uRPF, typically performed at the network (IGP areas or ASes) border routers, requires that a data packet can be only accepted when the FIB contains a prefix that encompasses the source address and the corresponding out-interface matches the data incoming

interface. It has the advantages of simple operation, easy deployment, and automatic update. However, in the case of multihoming, when the data incoming interface is different from the out-interface of the packet source IP address, using the longest prefix match, also referred to as asymmetric routing of data packets, Strict uRPF exhibits false positive.

Loose uRPF, sacrificing the directionality of Strict uRPF, only requires that the packet's source IP exists as a FIB entry. Intuitively, Loose uRPF cannot prevent the attacker from forging a source address that already exists in the FIB.

Feasible uRPF, typically performed at the network border routers, helps mitigate false positive of Strict uRPF in the multihoming scenarios. Instead of installing only the best route into FIB as Strict uRPF does, Feasible uRPF installs all alternative paths into the FIB. It helps reduce false positive filtering compared with the Strict uRPF, in the case when multiple paths are learnt from different interfaces. However, it should be noted that Feasible uRPF only works when multiple paths is learnt. There are cases when a device only learns one path but still has packets coming from other valid interfaces.

EFP-uRPF, specifically performed at the AS border routers, further improves Feasible uRPF in the inter-AS scenario. An AS performing EFP-uRPF maintains an individual RPF list on every customer/peer interface. It introduces two algorithms (i.e., Algorithm A and Algorithm B) regarding different application scenarios. In the case that a customer interface fails to learn any route from the directly connected customer AS, enabling Algorithm A at this customer interface may exhibit false positive filtering. In this case, enabling Algorithm B may mitigate the false positive. However, in case of directly connected customer ASes spoofing each other, Algorithm B exhibits false negative.

1.3. SAV Requirements and Challenges

As the above overview indicates, to evaluate the quality of a specific SAV technique, one should balance between two general requirements: precise filtering and automatic implementation.

- o Precise filtering: Two important indicators for precise filtering.
 - 1) 0% false positive. If legitimate packets may be dropped, it can seriously affect the user's internet experience.
 - 2) 0% false negative. If some packets with a forged source address may pass through the SAV smoothly, it will pose potential security risks.

- o Automatic implementation: In reality, the address space of an administrative domain (AD) may grow or update, and the routing policy within the address domain may be dynamically adjusted. One solution that relies entirely on manual configuration is neither scalable nor easy to deploy.

Then to consider the whole network SAV solution, one should never rely on a single point but a systematic SAV technique combination deployed at different network levels. As shown in Figure 1, packet filtering at different levels from the access network to the AS boarder are all needed. In Figure 1, the administrative domain (AD) concept is used, which refers to a network domain managed by the same operator (OTT, ISP and so on). One AD is allowed to be divided into several sub-ADs and managed by different inner groups. There may exist different levels for sub-ADs. For example, sub-AD1 is the upper level compared to sub-AD2, meaning that sub-AD2 needs to connect through sub-AD1 for external reachability (i.e., networks outside AD1). So filtering at sub-AD boarders (between different levels and within the same level) is also necessary. Further, different sub-ADs can belong to one single AS or multiple ASes, which makes the filtering at the sub-AD boarders either intra-AS filtering or inter-AS filtering. In the rest of this document, we use the term SAVA (SAV architecture) to refer to the discussion of the systematic SAV solution.

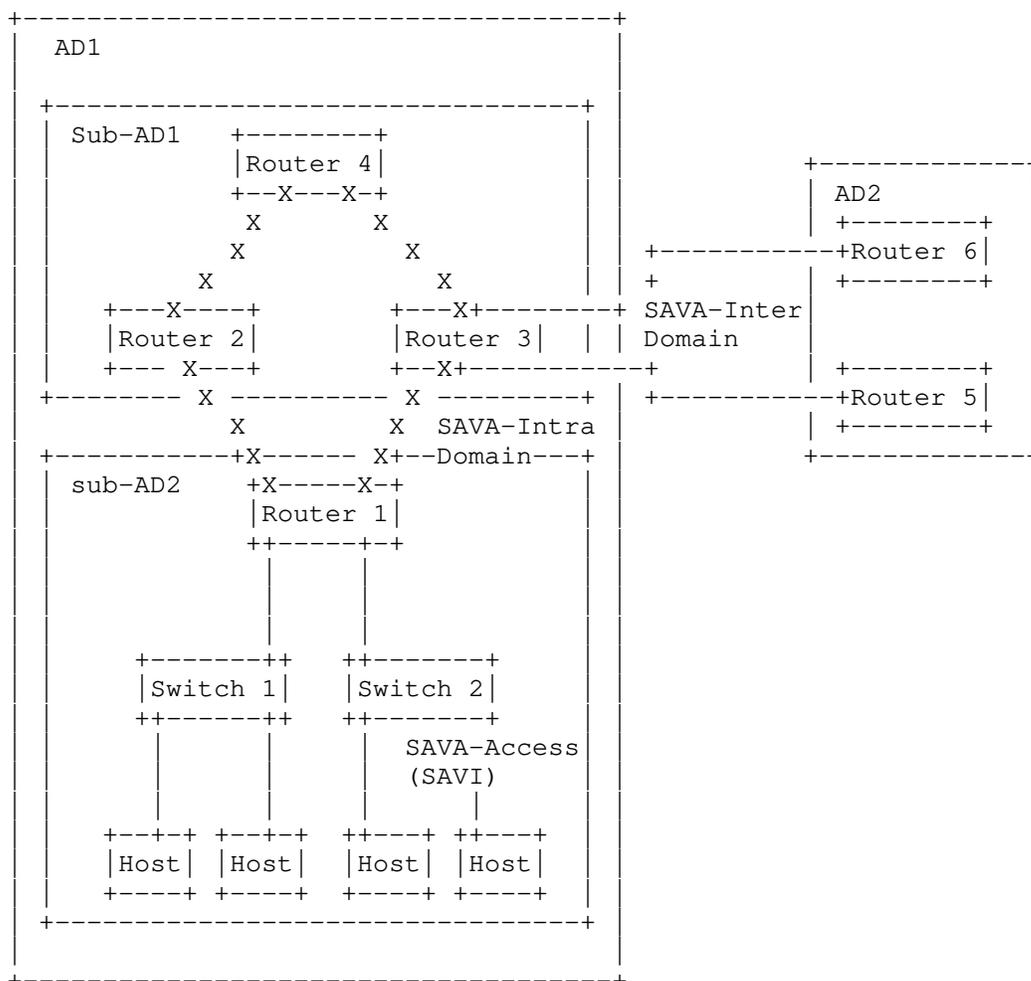


Figure 1: SAVA

Looking back at specific SAV approaches, most limitations are caused by multihoming. Further, it is due to the routing information asymmetry at the mutil-homed devices. This document identifies two specific scenarios where existing SAV techniques fail to meet the above mentioned requirements.

2. Terminology

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System

BGP: Border Gateway Protocol

FIB: Forwarding Information Base

SAV: Source Address Validation

SAVA: Source Address Validation Architecture

AD: Administrative Domain

3. Problem Statement

As stated in Section 1.3, existing methods, e.g., Loose/Strict mode uRPF, FP-uRPF, EFP-uRPF are not able to achieve 100% accurate filtering (i.e., 0% FN and 0% FP) in certain scenarios. This document specifically indicate two typical intra-domain cases that conventional approaches fail to cover: 1) all sub-ADs are under the same AS; 2) sub-ADs are under different ASes.

3.1. SAVA Intra-domain Use Case 1: Intra-AS Multi-homing

Figure 2 illustrates an intra-AS multihoming case, where sub-AD1, sub-AD2 and sub-AD3 are under the same AS.

Router 1 is multi-homed to Router 2 and Router 3. Router 1 doesn't announce any of its routes to Router 2 nor Router 3. Static routes are configured on Router 1, Router 2 and Router 3. Supposedly, both Router 2 and Router 3 should have static routes P1/P2 with Router 1 as next hop configured. However, due to configuration error, or traffic control purpose, on Router 3, no P1/P2 static routes are configured. Router 2 and Router 3 are connected with ISIS or OSPF. P1/P2 are flooded from Router 2 to Router 3.

Router 5 is single-homed to Router 3. Router 5 announces P3 to Router 3 using ISIS or OSPF. Router 3 floods P3 to Router 2 .

Now suppose two data flow coming from Router 1 to Router 3: Flow 1 with source IP as P1, and Flow 2 with source IP as P3 (IP spoofing). Using existing SAV methods at Router 3, Flow 1 is supposed to be passed, while Flow 2 is supposed to be dropped.

- o Loose uRPF: works for Flow 1, but fails for Flow 2.
- o Strict uRPF: works for Flow 2, but fails for Flow 1 (the incoming interface does not match P1/P2's out-interface).
- o FP-uRPF: works for Flow 2, but fails for Flow 1 (no feasible path for P1/P2 other than the best route exists).

- o EFP-uRPF: does not apply at the intra-AS case.

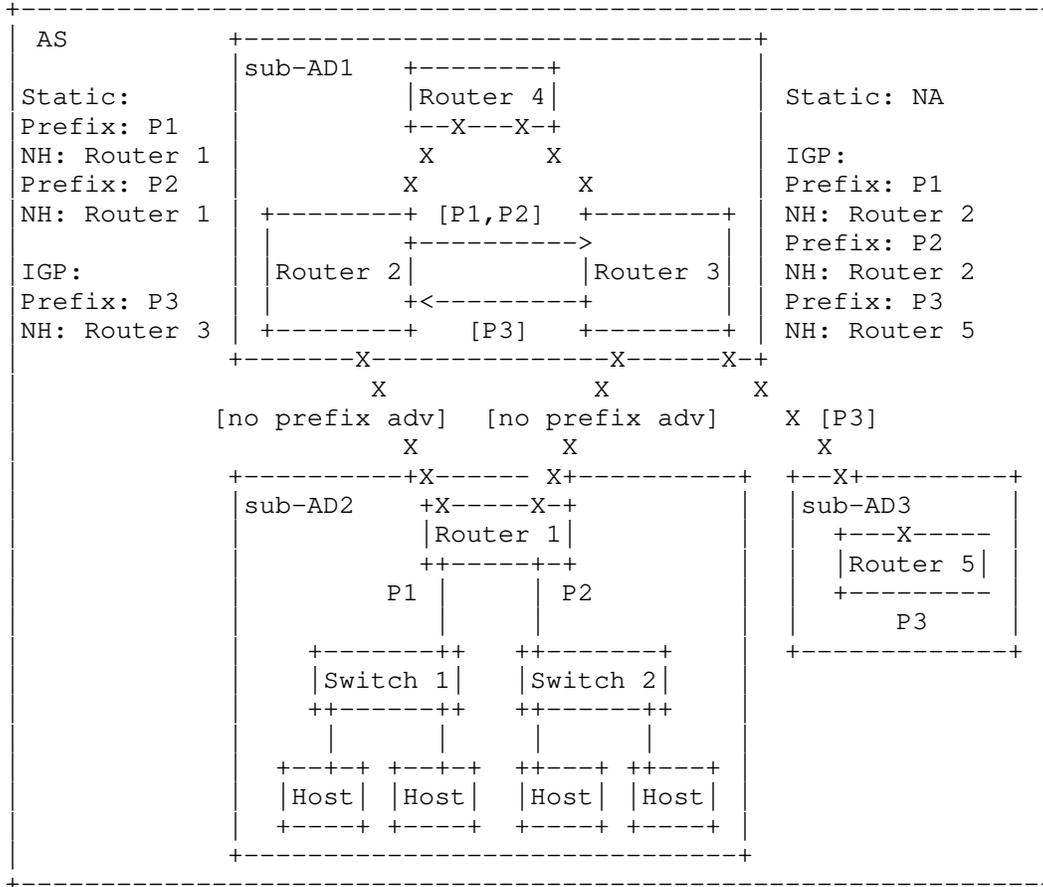


Figure 2: Asymmetric data flow in the Intra-AS scenario

3.2. SAVA Intra-domain Use Case 2: Inter-AS Multihoming

Figure 3 illustrates an inter-AS multihoming case, where sub-AD1, sub-AD2 and sub-AD3 are under three different ASes.

Router 1 (AS2) is multi-homed to Router 2 (AS1) and Router 3 (AS1). Router 1 announces P1/P2 to Router 2 through BGP. Router 1 doesn't announce any of its routes to Router 3 due to policy control. P1/P2 are propagated from Router 2 to Router 3 through BGP.

Router 5 (AS3) is single-homed to Router 3 (AS1). Router 5 announces P3 to Router 3 through BGP. Router 3 propagates P3 to Router 2 through BGP.

Now suppose two data flow coming from Router 1 to Router 3: Flow 1 with source IP as P1, and Flow 2 with source IP as P3 (IP spoofing). Using existing SAV methods at Router 3, Flow 1 is supposed to be passed, while Flow 2 is supposed to be dropped.

- o Loose uRPF: works for Flow 1, but fails for Flow 2.
- o Strict uRPF: works for Flow 2, but fails for Flow 1 (the incoming interface does not match P1/P2's out-interface).
- o FP-uRPF: works for Flow 2, but fails for Flow 1 (no feasible path for P1/P2 other than the best route exists).
- o EFP-uRPF: works for Flow 1, but fails for Flow 2 using Algorithm B. Works for Flow 2, but fails for Flow 1 when using Algorithm A.

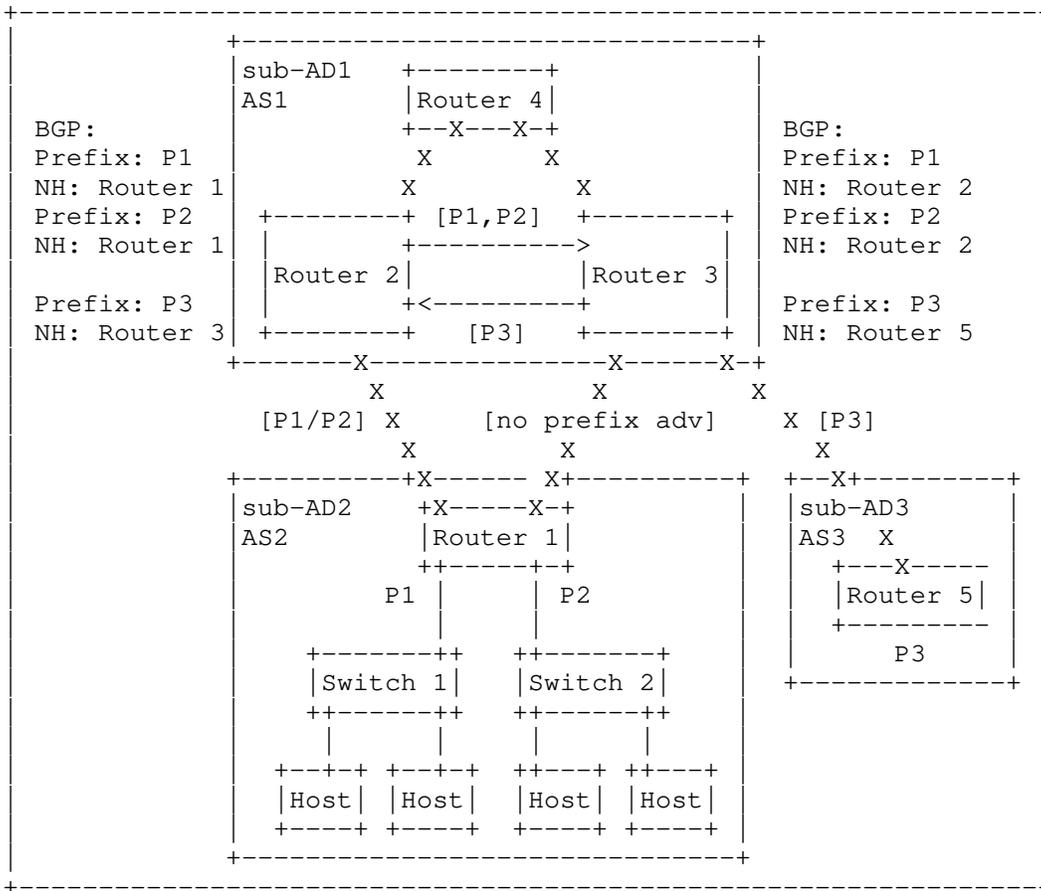


Figure 3: Asymmetric data flow in the Inter-AS scenario

4. Solution Consideration

Both EFP-uRPF and FP-uRPF try to achieve a balance between flexibility (Loose uRPF) and directionality (Strict uRPF).

In the inter-AS multi-homing scenario, EFP-uRPF further improves FR-uRPF's directionality, thanks to the availability of the route origin information. More specifically, the construction of RPF lists using EFP-uRPF Algorithm A or B is augmented with data from Route Origin Authorization (ROA) [RFC6482], as well as Internet Routing Registry (IRR) data, making EFP-uRPF performing better than FR-uRPF regarding directionality. In fact, the global availability of ROA and IRR databases provides a way for the multiple transit providers of the

same multihomed network to share such information without extra way of data synchronization.

In addition, although ERP-uRPF is striving for more accurate RPF list construction, there's still currently no way of constructing an 100%-accurate RPF list in the case shown in Figure 3. In order to to conquer such problem, it could help if devices in the upper level sub-AD(s) (i.e., Router 2 and Router 3) can share more information with each other through certain way.

What's worse, in case of the intra-AS multi-homing, as indicated in Figure2, there's no such prefix to sub-AD mapping (e.g., P3 originates from sub-AD3) database publicly available as ROA or IRR database, or automatically retrievable as RPKI ROA through RTR protocol [RFC8210]. Thus, enhancing such information sharing between devices of the upper level sub-AD(s) (i.e., Router 2 and Router 3) for the same multi-homed network, by extending certain routing protocols, could be a possible way.

5. Security Considerations

TBD

6. Contributors

TBD

7. Acknowledgments

TBD

8. Normative References

[I-D.brockners-inband-oam-requirements]

Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-grow-bmp-adj-rib-out]

Evens, T., Bayraktar, S., Lucente, P., Mi, K., and S. Zhuang, "Support for Adj-RIB-Out in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-adj-rib-out-07 (work in progress), August 2019.

- [I-D.ietf-grow-bmp-local-rib]
Evens, T., Bayraktar, S., Bhardwaj, M., and P. Lucente,
"Support for Local RIB in BGP Monitoring Protocol (BMP)",
draft-ietf-grow-bmp-local-rib-07 (work in progress), May
2020.
- [I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores",
draft-ietf-netconf-yang-push-25 (work in progress), May
2019.
- [I-D.openconfig-rtgw-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack,
C., and C. Morrow, "gRPC Network Management Interface
(gNMI)", draft-openconfig-rtgw-gnmi-spec-01 (work in
progress), March 2018.
- [I-D.song-ntf]
Song, H., Zhou, T., Li, Z., Fioccola, G., Li, Z.,
Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Toward a
Network Telemetry Framework", draft-song-ntf-02 (work in
progress), July 2018.
- [RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin,
"Simple Network Management Protocol (SNMP)", RFC 1157,
DOI 10.17487/RFC1157, May 1990,
<<https://www.rfc-editor.org/info/rfc1157>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
DOI 10.17487/RFC1191, November 1990,
<<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and
dual environments", RFC 1195, DOI 10.17487/RFC1195,
December 1990, <<https://www.rfc-editor.org/info/rfc1195>>.
- [RFC1213] McCloghrie, K. and M. Rose, "Management Information Base
for Network Management of TCP/IP-based internets: MIB-II",
STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991,
<<https://www.rfc-editor.org/info/rfc1213>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <<https://www.rfc-editor.org/info/rfc3704>>.
- [RFC3719] Parker, J., Ed., "Recommendations for Interoperable Networks using Intermediate System to Intermediate System (IS-IS)", RFC 3719, DOI 10.17487/RFC3719, February 2004, <<https://www.rfc-editor.org/info/rfc3719>>.
- [RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <<https://www.rfc-editor.org/info/rfc3988>>.
- [RFC6232] Wei, F., Qin, Y., Li, Z., Li, T., and J. Dong, "Purge Originator Identification TLV for IS-IS", RFC 6232, DOI 10.17487/RFC6232, May 2011, <<https://www.rfc-editor.org/info/rfc6232>>.
- [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>>.
- [RFC6959] McPherson, D., Baker, F., and J. Halpern, "Source Address Validation Improvement (SAVI) Threat Scope", RFC 6959, DOI 10.17487/RFC6959, May 2013, <<https://www.rfc-editor.org/info/rfc6959>>.
- [RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <<https://www.rfc-editor.org/info/rfc7039>>.

- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <<https://www.rfc-editor.org/info/rfc7854>>.
- [RFC8210] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", RFC 8210, DOI 10.17487/RFC8210, September 2017, <<https://www.rfc-editor.org/info/rfc8210>>.
- [RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <<https://www.rfc-editor.org/info/rfc8231>>.
- [RFC8704] Sriram, K., Montgomery, D., and J. Haas, "Enhanced Feasible-Path Unicast Reverse Path Forwarding", BCP 84, RFC 8704, DOI 10.17487/RFC8704, February 2020, <<https://www.rfc-editor.org/info/rfc8704>>.

Authors' Addresses

Dan Li
Tsinghua
Beijing
China

Email: tolidan@tsinghua.edu.cn

Jianping Wu
Tsinghua
Beijing
China

Email: jianping@cernet.edu.cn

Yunan Gu
Huawei
Beijing
China

Email: guyunan@huawei.com

Lancheng Qin
Tsinghua
Beijing
China

Email: qlc19@mails.tsinghua.edu.cn

Tao Lin
H3C
Beijing
China

Email: lintao@h3c.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2021

T. Graf
Swisscom
March 24, 2021

Export of MPLS Segment Routing Label Type Information in
IP Flow Information Export (IPFIX)
draft-tgraf-ipfix-mpls-sr-label-type-07

Abstract

This document introduces additional code points in the mplsTopLabelType Information Element for IS-IS, OSPFv2, OSPFv3 and BGP MPLS Segment Routing (SR) extensions to enable Segment Routing label protocol type information in IP Flow Information Export (IPFIX).

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 September 25, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. MPLS Segment Routing Top Label Type	2
3. IANA Considerations	3
4. Security Considerations	4
5. Acknowledgements	4
6. References	4
Author's Address	6

1. Introduction

Besides BGP-4 [RFC8277], LDP [RFC5036] and BGP VPN [RFC4364], four new routing-protocols, OSPFv2 Extensions [RFC8665], OSPFv3 Extensions [RFC8666], IS-IS Extensions [RFC8667] and BGP Prefix-SID [RFC8669] have been added to the list of routing-protocols able to propagate Segment Routing labels for the MPLS data plane [RFC8660].

Traffic Accounting in Segment Routing Networks

[I-D.ali-spring-sr-traffic-accounting] describes how IPFIX can be leveraged to account traffic to MPLS Segment Routing label dimensions within a Segment Routing domain.

In the Information Model for IP Flow Information Export IPFIX [RFC7012], the information element `mplsTopLabelType(46)` describes which MPLS control plane protocol allocated the top-of-stack label in the MPLS label stack. RFC 7012 section 7.2 [RFC7012] describes the "IPFIX MPLS label type (Value 46)" sub-registry [IANA-IPFIX-IE46] where new code points should be added.

2. MPLS Segment Routing Top Label Type

By introducing four new code points to information element `mplsTopLabelType(46)` for IS-IS, OSPFv2, OSPFv3 and BGP Prefix-SID, when Segment Routing with one of these four routing protocols is deployed, we get insight into which traffic is being forwarded based on which MPLS control plane protocol.

A typical use case scenario is to monitor MPLS control plane migrations from LDP to IS-IS or OSPF Segment Routing. Such a migration can be done node by node as described in RFC8661 [RFC8661]

Another use case is the monitoring of a migration to a Seamless MPLS SR [I-D.hegde-spring-mpls-seamless-sr] architecture where prefixes are propagated with dynamic BGP labels according to RFC8277

[RFC8277], BGP Prefix-SID according to RFC8669 [RFC8669] and used for the forwarding between IGP domains. Adding an additional layer into the MPLS data plane to above described use case.

Both use cases can be verified by using `mplsTopLabelType(46)`, `mplsTopLabelIPv4Address(47)`, `mplsTopLabelStackSection(70)` and `forwardingStatus(89)` dimensions to get insights into

- o how many packets are forwarded or dropped
- o if dropped, for which reasons
- o the MPLS provider edge loopback address and label protocol

By looking at the MPLS label value itself, it is not always clear as to which label protocol it belongs, since they could potentially share the same label allocation range. This is the case for IGP-Adjacency SID's, LDP and dynamic BGP labels as an example.

3. IANA Considerations

This document specifies four additional code points for IS-IS, OSPFv2, OSPFv3 and BGP Prefix-SID Segment Routing extension in the existing sub-registry "IPFIX MPLS label type (Value 46)" of the "IPFIX Information Elements" and one new "IPFIX Information Element" with a new sub-registry in the "IP Flow Information Export (IPFIX) Entities" name space.

Value	Description	Reference	Requester
TBD1	OSPFv2 Segment Routing	RFC8665	[RFC-to-be]
TBD2	OSPFv3 Segment Routing	RFC8666	[RFC-to-be]
TBD3	IS-IS Segment Routing	RFC8667	[RFC-to-be]
TBD4	BGP Segment Routing Prefix-SID	RFC8669	[RFC-to-be]

Figure 1: Updates to "IPFIX MPLS label type (Value 46)" SubRegistry

Note to IANA:

- o Please assign TBD1 to 4 to the next available numbers according to the "IPFIX MPLS label type (Value 46)" sub-registry [IANA-IPFIX-IE46] procedure.

- o Please replace the [RFC-to-be] with the RFC number assigned to this document.

Note to RFC-editor:

- o Please remove above two IANA notes.

4. Security Considerations

There exists no extra security considerations regarding the allocation of these new IPFIX information elements compared to RFC7012 [RFC7012].

5. Acknowledgements

I would like to thank to the IE doctors, Paul Aitken and Andrew Feren, as well Benoit Claise, Loa Andersson, Tianran Zhou, Pierre Francois, Bruno Decreane, Paolo Lucente, Hannes Gredler, Ketan Talaulikar, Sabrina Tanamal, Erik Auerswald and Sergey Fomin for their review and valuable comments.

6. References

6.1. Normative References

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <<https://www.rfc-editor.org/info/rfc7012>>.

6.2. Informative References

[I-D.ali-spring-sr-traffic-accounting]
Filsfils, C., Talaulikar, K., Sivabalan, S., Horneffer, M., Raszuk, R., Litkowski, S., Voyer, D., and R. Morton, "Traffic Accounting in Segment Routing Networks", draft-ali-spring-sr-traffic-accounting-04 (work in progress), February 2020.

[I-D.hegde-spring-mpls-seamless-sr]
Hegde, S., Bowers, C., Xu, X., Gulko, A., Bogdanov, A., Uttaro, J., Jalil, L., Khaddam, M., and A. Alston, "Seamless Segment Routing", draft-hegde-spring-mpls-seamless-sr-04 (work in progress), January 2021.

- [IANA-IPFIX-IE46] "IANA IP Flow Information Export (IPFIX) Information Element #46 SubRegistry", <<https://www.iana.org/assignments/ipfix/ipfix.xhtml#ipfix-mpls-label-type>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <<https://www.rfc-editor.org/info/rfc8660>>.
- [RFC8661] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., and S. Litkowski, "Segment Routing MPLS Interworking with LDP", RFC 8661, DOI 10.17487/RFC8661, December 2019, <<https://www.rfc-editor.org/info/rfc8661>>.
- [RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <<https://www.rfc-editor.org/info/rfc8665>>.
- [RFC8666] Psenak, P., Ed. and S. Previdi, Ed., "OSPFv3 Extensions for Segment Routing", RFC 8666, DOI 10.17487/RFC8666, December 2019, <<https://www.rfc-editor.org/info/rfc8666>>.
- [RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <<https://www.rfc-editor.org/info/rfc8667>>.

[RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <<https://www.rfc-editor.org/info/rfc8669>>.

Author's Address

Thomas Graf
Swisscom
Binzring 17
Zurich 8045
Switzerland

Email: thomas.graf@swisscom.com

OPSAWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 25, 2021

B. Wu
Q. Wu
Huawei
M. Boucadair
Orange
O. Gonzalez de Dios
Telefonica
B. Wen
Comcast
C. Liu
China Unicom
H. Xu
China Telecom
January 21, 2021

A YANG Model for Network and VPN Service Performance Monitoring
draft-www-opsawg-yang-vpn-service-pm-03

Abstract

The data model defined in RFC8345 introduces vertical layering relationships between networks that can be augmented to cover network/service topologies. This document defines a YANG model for both Network Performance Monitoring and VPN Service Performance Monitoring that can be used to monitor and manage network performance on the topology at higher layer or the service topology between VPN sites.

This document does not define metrics for network performance or mechanisms for measuring network performance. The YANG model defined in this document is designed as an augmentation to the network topology YANG model defined in RFC 8345 and draws on relevant YANG types defined in RFC 6991, RFC 8299, RFC 8345, and RFC 8532.

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 July 25, 2021.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. Network and VPN Service Performance Monitoring Model Usage	3
3.1. Retrieval via Pub/Sub Mechanism	4
3.2. On demand Retrieval via RPC Model	5
4. Description of the Data Model	5
4.1. Layering Relationship Between Multiple Layers of Topology	5
4.2. Network Level	6
4.3. Node Level	7
4.4. Link and Termination Point Level	8
5. Example of I2RS Pub/Sub Retrieval	11
6. Example of RPC-based Retrieval	12
7. Network and VPN Service Assurance YANG Module	14
8. Security Considerations	26
9. IANA Considerations	26
10. Acknowledgements	27
11. Contributors	27
12. References	27
12.1. Normative References	27
12.2. Informative References	29
Authors' Addresses	30

1. Introduction

[RFC4176] provides a framework for L3VPN operations and management and specifies that performance management is required after service configuration. This document defines a YANG Model for both network performance monitoring and VPN service performance monitoring that can be used to monitor and manage network performance on the topology level or the service topology between VPN sites.

This document does not introduce new metrics for network performance or mechanisms for measuring network performance, but uses the existing mechanisms and statistics to show the performance monitoring statistics at the network and service layers. The YANG model defined in this document is designed as an augmentation to the network topology YANG model defined in [RFC8345] and draws on relevant YANG types defined in [RFC6991], [RFC8299], [RFC8345], and [RFC8532].

2. Terminology

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

Tree diagrams used in this document follow the notation defined in [RFC8340].

3. Network and VPN Service Performance Monitoring Model Usage

Models are key for automatic management operations. According to [I-D.ietf-opsawg-model-automation-framework] , together with service and network models, performance measurement telemetry model can monitor network performance to meet specific service SLA requirements. The model defined in this document is to derive VPN or network level performance data based on lower-level data collected via monitoring counters in the devices.

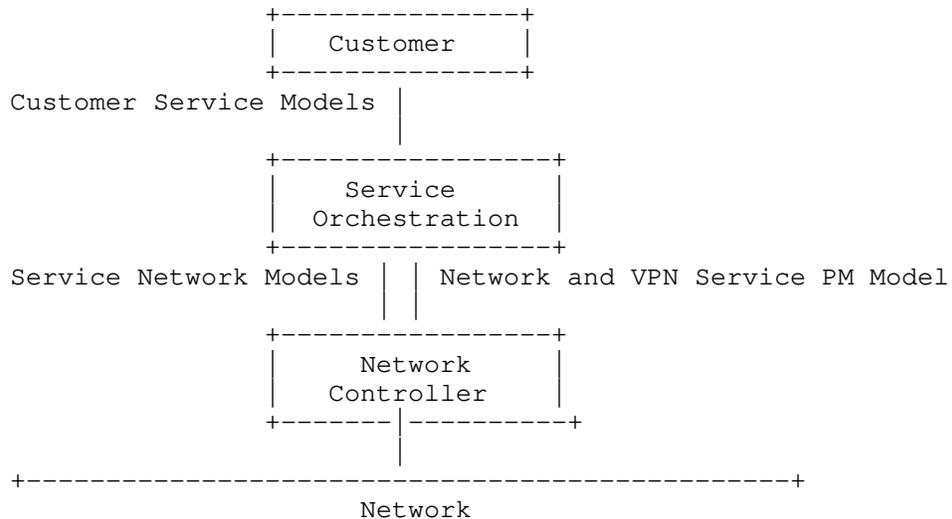


Figure 1: Reference Architecture

As shown in Figure 1 , the network and VPN service performance monitoring model can be used to expose some performance information to the above layer. The information can be used by the orchestrator to subscribe to performance data. The controller will then notify the orchestrator of corresponding parameter changes.

Before using the Network and VPN Service PM Model, the mapping between the VPN Service topology and the underlying physical network has been setup, and the performance monitoring data per link in the underlying network can be collected using network performance measurement method such as MPLS Loss and Delay Measurement [RFC6374].

The performance monitoring information reflecting the quality of the Network or VPN service such as end to end network performance data between source node and destination node in the network or between VPN sites can be aggregated or calculated using, for example, PCEP solution [RFC8233] [RFC7471] [RFC8570] [RFC8571] or LMAP [RFC8194].

The measurement interval and report interval associated with these performance data usually depends on configuration parameters.

3.1. Retrieval via Pub/Sub Mechanism

Some applications such as service-assurance applications, which must maintain a continuous view of operational data and state, can use subscription model [RFC8641] to subscribe to the specific Network

performance data or VPN service performance data they are interested in, at the data source.

The data source can then use the Network and VPN service assurance model defined in this document and the YANG Push model [RFC8641] to distribute specific telemetry data to target recipients.

3.2. On demand Retrieval via RPC Model

To obtain a snapshot of a large amount of performance data from a network element (including network controllers), service-assurance applications may use polling-based methods such as RPC model to fetch performance data on demand.

4. Description of the Data Model

This document defines the YANG module "ietf-network-vpn-pm", which is an augmentation to the "ietf-network" and "ietf-network-topology".

The performance monitoring data is augmented to service topology as shown in Figure 2.

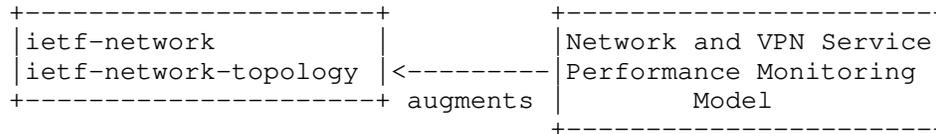


Figure 2: Module Augmentation

4.1. Layering Relationship Between Multiple Layers of Topology

[RFC8345] defines a YANG [RFC7950] data model for network/service topologies and inventories. The service topology described in [RFC8345] includes the virtual topology for a service layer above Layer 1 (L1), Layer 2 (L2), and Layer 3 (L3). This service topology has the generic topology elements of node, link, and terminating point. One typical example of a service topology is described in Figure 3 of [RFC8345]: two VPN service topologies instantiated over a common L3 topology. Each VPN service topology is mapped onto a subset of nodes from the common L3 topology.

Figure 3 illustrates an example of a topology mapping between the VPN service topology and an underlying network:

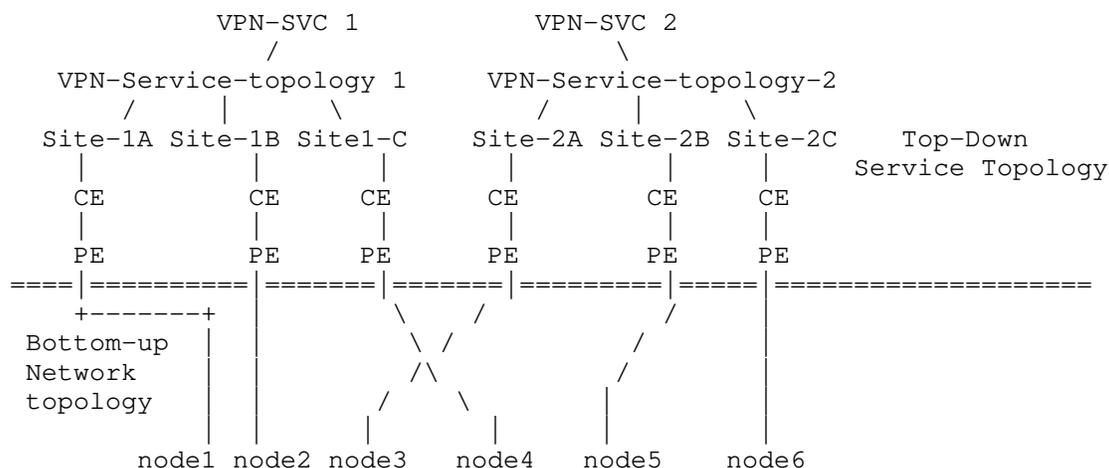


Figure 3: Example of topology mapping between VPN Service Topo and Underlying network

As shown in Figure 3, two VPN services topologies are both built on top of one common underlying physical network:

- o VPN-SVC 1: supporting "hub-spoke" communications for Customer 1 connecting the customer's access at 3 sites. Site-1A, Site-1B, and Site-1C are connected to PEs that are mapped to nodes 1, 2, and 3 in the underlying physical network. Site-1 A plays the role of hub while Site-2 B and C plays the role of spoke.
- o VPN-SVC 2: supporting "hub-spoke disjoint" communications for Customer 2 connecting the customer's access at 3 sites. Site-2A, Site-2B, and Site-2C are connected to PEs that are mapped to nodes 4, 5, and 6 in the underlying physical network. Site-2 A and B play the role of hub while Site-2 C plays the role of spoke.

4.2. Network Level

For network performance monitoring, the attributes of "Network Level" that defined in [RFC8345] do not need to be extended.

For VPN service performance monitoring, this document defines some new network service type: "L3VPN, L2VPN". When a network topology data instance contains the L3VPN or L2VPN network type, it represents an VPN instance that can perform performance monitoring.

This model defines only the following minimal set of Network level network topology attributes:

- o "vpn-id": Refers to an identifier of VPN service (e.g., L3NM[I-D.ietf-opsawg-l3sm-l3nm]). This identifier allows to correlate the performance status with the network service configuration.
- o "vpn-topo": The type of VPN service topology, this model supports "any-to-any", "Hub and Spoke" (where Hubs can exchange traffic), and "Hub and Spoke disjoint" (where Hubs cannot exchange traffic). [RFC8299] defines a YANG model for L3VPN Service Delivery. Three types of VPN service topologies are supported in : "any to any", "hub and spoke", and "hub and spoke disjoint". These VPN topology types can be used to describe how VPN sites communicate with each other.

```

module: ietf-network-vpn-pm
  augment /nw:networks/nw:network/nw:network-types:
    +--rw network-service-type!
      +--rw network-service-type?  identityref
  augment /nw:networks/nw:network:
    +--rw vpn-topo-attributes
      +--rw vpn-id?      vpn-common:vpn-id
      +--rw vpn-topology?  identityref

```

Figure 4: Network Level View of the hierarchies

4.3. Node Level

For network performance monitoring, the attributes of "Node Level" that defined in [RFC8345] do not need to be extended.

For VPN service performance monitoring, this model defines only the following minimal set of Node level network topology attributes:

- o "node-type" (Attribute): Indicates the type of the node, such as PE or ASBR. This "node-type" can be used to report performance metric between any two nodes each with specific node-type.
- o "site-id" (Constraint): Uniquely identifies the site within the overall network infrastructure.
- o "site-role" (Constraint): Defines the role of the site in a particular VPN topology.

- o "vpn-summary-statistics": IPv4 statistics, and IPv6 statistics have been specified separately. And MAC statistics could be extended for L2VPN.

```
augment /nw:networks/nw:network/nw:node:
  +--rw node-attributes
  |   +--rw node-type?    identityref
  |   +--rw site-id?     string
  |   +--rw site-role?   identityref
  +--rw vpn-summary-statistics
  |   +--rw ipv4
  |   |   +--rw total-routes?          uint32
  |   |   +--rw total-active-routes?   uint32
  |   +--rw ipv6
  |   |   +--rw total-routes?          uint32
  |   |   +--rw total-active-routes?   uint32
```

Figure 5: Node Level View of the hierarchies

4.4. Link and Termination Point Level

The link nodes are classified into two types: one is topology link defined in [RFC8345], and the other is abstract link of a VPN between PEs.

The performance data of the link is a collection of counters that report the performance status. The data for the topology link can be based on BGP-LS [RFC8571]. The statistics of the VPN abstract links can be collected based on VPN OAM mechanisms, e.g. TWAMP etc. Alternatively, the data can base on the underlay technology OAM mechanism, for example, GRE tunnel OAM.

```

augment /nw:networks/nw:network/nt:link:
  +--rw link-type?    identityref
augment /nw:networks/nw:network/nt:link:
  +--rw low-percentile?           percentile
  +--rw middle-percentile?        percentile
  +--rw high-percentile?          percentile
  +--rw reference-time?           yang:date-and-time
  +--rw measurement-interval?     uint32
  +--ro link-telemetry-attributes
    +--ro loss-statistics
      +--ro packet-loss-count?    yang:counter32
      +--ro packet-reorder-count? yang:counter32
      +--ro packets-out-of-seq-count? yang:counter32
      +--ro packets-dup-count?    yang:counter32
      +--ro loss-ratio?            percentage
    +--ro delay-statistics
      +--ro direction?            identityref
      +--ro unit-value?           identityref
      +--ro min-delay-value?      yang:gauge64
      +--ro max-delay-value?      yang:gauge64
      +--ro low-delay-percentile? yang:gauge64
      +--ro middle-delay-percentile? yang:gauge64
      +--ro high-delay-percentile? yang:gauge64
    +--ro jitter-statistics
      +--ro unit-value?           identityref
      +--ro min-jitter-value?     yang:gauge32
      +--ro max-jitter-value?     yang:gauge32
      +--ro low-jitter-percentile? yang:gauge32
      +--ro middle-jitter-percentile? yang:gauge32
      +--ro high-jitter-percentile? yang:gauge32
augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--ro tp-telemetry-attributes
    +--ro inbound-octets?         yang:counter64
    +--ro inbound-unicast?        yang:counter64
    +--ro inbound-nunicast?       yang:counter64
    +--ro inbound-discards?       yang:counter32
    +--ro inbound-errors?         yang:counter32
    +--ro inbound-unknown-protocol? yang:counter32
    +--ro outbound-octets?        yang:counter64
    +--ro outbound-unicast?       yang:counter64
    +--ro outbound-nunicast?      yang:counter64
    +--ro outbound-discards?      yang:counter32
    +--ro outbound-errors?        yang:counter32
    +--ro outbound-qlen?          uint32

```

Figure 6: Link and Termination point Level View of the hierarchies

For the nodes of the link in the figure, this module defines the following minimal set of link level performance attributes:

- o "link-type": Indicates the abstract link of a VPN, such as GRE or IP-in-IP. The leaf refers to an identifier of VPN Common "underlay-transport" [I-D.ietf-opsawg-vpn-common], which describes the transport technology to carry the traffic of the VPN service.
- o Percentile parameters: The module supports reporting delay and jitter metric by percentile values. By default, low percentile (10th percentile), mid percentile (50th percentile), high percentile (90th percentile) are used. Setting a percentile into 0.00 indicates the client is not interested in receiving particular percentile. If all percentile nodes are set to 0.00, this represents that no percentile related nodes will be reported for a given performance metric (e.g. one-way delay, one-way delay variation) and only peak/min values will be reported. For example, a client can inform the server that it is interested in receiving only high percentiles. Then for a given link, at a given "reference-time" "measurement-interval", the high-delay-percentile and high-jitter-percentile will be reported.
- o Loss Statistics: A set of loss statistics attributes that are used to measure end to end loss between VPN sites or between any two network nodes. The exact loss value or the loss percentage can be reported.
- o Delay Statistics: A set of delay statistics attributes that are used to measure end to end latency between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.
- o Jitter Statistics: A set of IP Packet Delay Variation [RFC3393] statistics attributes that are used to measure end to end jitter between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.

For the nodes of "termination points" in the figure, the module defines the following minimal set of statistics:

- o Inbound statistics: A set of inbound statistics attributes that are used to measure the inbound statistics of the termination point, such as received packets, received packets with errors, etc.
- o Outbound statistics: A set of outbound statistics attributes that are used to measure the outbound statistics of the termination

point, such as sent packets, packets that could not be sent due to errors, etc.

5. Example of I2RS Pub/Sub Retrieval

This example shows the way for a client to subscribe for the Performance monitoring information between node A and node B in the L3 network topology built on top of the underlying network . The performance monitoring parameter that the client is interested in is end to end loss attribute.

```
<rpc netconf:message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <stream-subtree-filter>
      <networks
        xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topo">
        <network>
          <network-id>l3-network</network-id>
          <network-service-type
            xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
            L3VPN
          </network-service-type>
          <node>
            <node-id>A</node-id>
            <node-attributes
              xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
              <node-type>pe</node-type>
            </node-attributes>
            <termination-point
              xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topology">
              <tp-id>1-0-1</tp-id>
              <tp-telemetry-attributes
                xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
                <inbound-octets>150</inbound-octets>
                <outbound-octets>100</outbound-octets>
              </tp-telemetry-attributes>
            </termination-point>
          </node>
          <node>
            <node-id>B</node-id>
            <node-attributes
              xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
              <node-type>pe</node-type>
            </node-attributes>
            <termination-point
              xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topology">
```

```

        <tp-id>2-0-1</tp-id>
        <tp-telemetry-attributes
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
        <inbound-octets>150</inbound-octets>
        <outbound-octets>100</outbound-octets>
        </tp-telemetry-attributes>
    </termination-point>
</node>
<link
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topology">
    <link-id>A-B</link-id>
    <source>
        <source-node>A</source-node>
    </source>
    <destination>
        <dest-node>B</dest-node>
    </destination>
    <link-type>mpls-te</link-type>
    <link-telemetry-attributes
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
        <loss-statistics>
            <packet-loss-count>100</packet-loss-count>
        </loss-statistics>
    </link-telemetry-attributes>
    </link>
</network>
</networks>
</stream-subtree-filter>
<period
xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    500
</period>
</establish-subscription>
</rpc>

```

6. Example of RPC-based Retrieval

This example shows the way for the client to use RPC model to fetch performance data on demand, e.g., the client requests "packet-loss-count" between PE1 in site 1 and PE2 in site 2 belonging to the same VPN1.

```

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="1">
    <report
xmlns="urn:ietf:params:xml:ns:yang:example-service-pm-report">
        <networks xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topo">
            <network>

```

```
<network-id>vpn1</network-id>
<node>
  <node-id>A</node-id>
  <node-attributes
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
    <node-type>pe</node-type>
  </node-attributes>
  <termination-point
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topology">
    <tp-id>1-0-1</tp-id>
    <tp-telemetry-attributes
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
      <inbound-octets>100</inbound-octets>
      <outbound-octets>150</outbound-octets>
    </tp-telemetry-attributes>
    </termination-point>
  </node>
<node>
  <node-id>B</node-id>
  <node-attributes
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
    <node-type>pe</node-type>
  </node-attributes>
  <termination-point
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topology">
    <tp-id>2-0-1</tp-id>
    <tp-telemetry-attributes
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
      <inbound-octets>150</inbound-octets>
      <outbound-octets>100</outbound-octets>
    </tp-telemetry-attributes>
    </termination-point>
  </node>
<link>
  <link-id>A-B</link-id>
  <source>
    <source-node>A</source-node>
  </source>
  <destination>
    <dest-node>B</dest-node>
  </destination>
  <link-type>mpls-te</link-type>
  <telemetry-attributes
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-pm">
    <loss-statistics>
      <packet-loss-count>120</packet-loss-count>
    </loss-statistics>
  </telemetry-attributes>
```

```
        </link>
      </network>
    </report>
  </rpc>
```

7. Network and VPN Service Assurance YANG Module

This module uses types defined in [RFC8345], [RFC8299] and [RFC8532].

```
<CODE BEGINS> file "ietf-network-vpn-pm@2021-01-15.yang"
module ietf-network-vpn-pm {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm";
  prefix nvp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Types.";
  }
  import ietf-vpn-common {
    prefix vpn-common;
  }
  import ietf-network {
    prefix nw;
    reference
      "Section 6.1 of RFC 8345: A YANG Data Model for Network
      Topologies";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "Section 6.2 of RFC 8345: A YANG Data Model for Network
      Topologies";
  }
  import ietf-lime-time-types {
    prefix lime;
    reference
      "RFC 8532: Generic YANG Data Model for the Management of
      Operations, Administration, and Maintenance (OAM) Protocols
      That Use Connectionless Communications";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "Editor: Qin Wu
     <bill.wu@huawei.com>
```

```
Editor: Bo Wu
      <lane.wubo@huawei.com>
Editor: Mohamed Boucadair
      <mohamed.boucadair@orange.com>";
description
  "This module defines a model for Network and VPN Service Performance
  monitoring.

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

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://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 2021-01-15 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Model for Network and VPN Service Performance
    Monitoring";
}

identity pe {
  base vpn-common:role;
  description
    "Identity for PE type";
}

identity ce {
  base vpn-common:role;
  description
    "Identity for CE type";
}

identity asbr {
  base vpn-common:role;
  description
    "Identity for ASBR type";
}

identity p {
```

```
    base vpn-common:role;
    description
      "Identity for P type";
  }

  identity link-type {
    base vpn-common:protocol-type;
    description
      "Base identity for link type, e.g., GRE, MPLS TE, VXLAN.";
  }

  identity VXLAN {
    base link-type;
    description
      "Base identity for VXLAN Tunnel.";
  }

  identity ip-in-ip {
    base link-type;
    description
      "Base identity for IP in IP Tunnel.";
  }

  identity direction {
    description
      "Base Identity for measurement direction including
      one way measurement and two way measurement.";
  }

  identity one-way {
    base direction;
    description
      "Identity for one way measurement.";
  }

  identity two-way {
    base direction;
    description
      "Identity for two way measurement.";
  }

  typedef percentage {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    description
      "Percentage.";
  }
```

```
}

typedef percentile {
  type decimal64 {
    fraction-digits 5;
  }
  description
    "The percentile is a statistical value that indicates that a
    certain percentage of a set of data falls below it.";
}

grouping vpn-summary-statistics {
  description
    "VPN Statistics grouping used for network topology
    augmentation.";
  container vpn-summary-statistics {
    description
      "Container for VPN summary statistics.";
    container ipv4 {
      leaf total-routes {
        type uint32;
        description
          "Total routes for the VPN.";
      }
      leaf total-active-routes {
        type uint32;
        description
          "Total active routes for the VPN.";
      }
      description
        "IPv4-specific parameters.";
    }
    container ipv6 {
      leaf total-routes {
        type uint32;
        description
          "Total routes for the VPN.";
      }
      leaf total-active-routes {
        type uint32;
        description
          "Total active routes for the VPN.";
      }
      description
        "IPv6-specific parameters.";
    }
  }
}
```

```
grouping link-error-statistics {
  description
    "Grouping for per link error statistics.";
  container loss-statistics {
    description
      "Per link loss statistics.";
    leaf packet-loss-count {
      type yang:counter32;
      description
        "Total received packet drops count.";
    }
    leaf packet-reorder-count {
      type yang:counter32;
      description
        "Total received packet reordered count.";
    }
    leaf packets-out-of-seq-count {
      type yang:counter32;
      description
        "Total received out of sequence count.";
    }
    leaf packets-dup-count {
      type yang:counter32;
      description
        "Total received packet duplicates count.";
    }
    leaf loss-ratio {
      type percentage;
      description
        "Loss ratio of the packets. Express as percentage
        of packets lost with respect to packets sent.";
    }
  }
}

grouping link-delay-statistics {
  description
    "Grouping for per link delay statistics";
  container delay-statistics {
    description
      "Link delay summarised information. By default,
      one way measurement protocol (e.g., OWAMP) is used
      to measure delay.";
    leaf direction {
      type identityref {
        base direction;
      }
      default "one-way";
    }
  }
}
```

```
    description
      "Define measurement direction including one way
       measurement and two way measurement.";
  }
  leaf unit-value {
    type identityref {
      base lime:time-unit-type;
    }
    default "lime:milliseconds";
    description
      "Time units, where the options are s, ms, ns, etc.";
  }
  leaf min-delay-value {
    type yang:gauge64;
    description
      "Minimum delay value observed.";
  }
  leaf max-delay-value {
    type yang:gauge64;
    description
      "Maximum delay value observed.";
  }
  leaf low-delay-percentile {
    type yang:gauge64;
    description
      "Low percentile of the delay observed with
       specific measurement method.";
  }
  leaf middle-delay-percentile {
    type yang:gauge64;
    description
      "Middle percentile of the delay observed with
       specific measurement method.";
  }
  leaf high-delay-percentile {
    type yang:gauge64;
    description
      "High percentile of the delay observed with
       specific measurement method.";
  }
}
}

grouping link-jitter-statistics {
  description
    "Grouping for per link jitter statistics";
  container jitter-statistics {
    description
```

```
    "Link jitter summarised information. By default,
      jitter is measured using IP Packet Delay Variation
      (IPDV).";
  leaf unit-value {
    type identityref {
      base lime:time-unit-type;
    }
    default "lime:milliseconds";
    description
      "Time units, where the options are s, ms, ns, etc.";
  }
  leaf min-jitter-value {
    type yang:gauge32;
    description
      "Minimum jitter value observed.";
  }
  leaf max-jitter-value {
    type yang:gauge32;
    description
      "Maximum jitter value observed.";
  }
  leaf low-jitter-percentile {
    type yang:gauge32;
    description
      "Low percentile of the jitter observed.";
  }
  leaf middle-jitter-percentile {
    type yang:gauge32;
    description
      "Middle percentile of the jitter observed.";
  }
  leaf high-jitter-percentile {
    type yang:gauge32;
    description
      "High percentile of the jitter observed.";
  }
}

grouping tp-svc-telemetry {
  leaf inbound-octets {
    type yang:counter64;
    description
      "The total number of octets received on the
        interface, including framing characters.";
  }
  leaf inbound-unicast {
    type yang:counter64;
  }
}
```

```
    description
      "Inbound unicast packets were received, and delivered
      to a higher layer during the last period.";
  }
  leaf inbound-nunicast {
    type yang:counter64;
    description
      "The number of non-unicast (i.e., subnetwork-
      broadcast or subnetwork-multicast) packets
      delivered to a higher-layer protocol.";
  }
  leaf inbound-discards {
    type yang:counter32;
    description
      "The number of inbound packets which were chosen
      to be discarded even though no errors had been
      detected to prevent their being deliverable to a
      higher-layer protocol.";
  }
  leaf inbound-errors {
    type yang:counter32;
    description
      "The number of inbound packets that contained
      errors preventing them from being deliverable to a
      higher-layer protocol.";
  }
  leaf inbound-unknown-protocol {
    type yang:counter32;
    description
      "The number of packets received via the interface
      which were discarded because of an unknown or
      unsupported protocol.";
  }
  leaf outbound-octets {
    type yang:counter64;
    description
      "The total number of octets transmitted out of the
      interface, including framing characters.";
  }
  leaf outbound-unicast {
    type yang:counter64;
    description
      "The total number of packets that higher-level
      protocols requested be transmitted to a
      subnetwork-unicast address, including those that
      were discarded or not sent.";
  }
  leaf outbound-nunicast {
```

```
    type yang:counter64;
    description
      "The total number of packets that higher-level
       protocols requested be transmitted to a non-
       unicast (i.e., a subnetwork-broadcast or
       subnetwork-multicast) address, including those
       that were discarded or not sent.";
  }
  leaf outbound-discards {
    type yang:counter32;
    description
      "The number of outbound packets which were chosen
       to be discarded even though no errors had been
       detected to prevent their being transmitted. One
       possible reason for discarding such a packet could
       be to free up buffer space.";
  }
  leaf outbound-errors {
    type yang:counter32;
    description
      "The number of outbound packets that contained
       errors preventing them from being deliverable to a
       higher-layer protocol.";
  }
  leaf outbound-qlen {
    type uint32;
    description
      " Length of the queue of the interface from where
       the packet is forwarded out. The queue depth could
       be the current number of memory buffers used by the
       queue and a packet can consume one or more memory buffers
       thus constituting device-level information.";
  }
  description
    "Grouping for interface service telemetry.";
}

augment "/nw:networks/nw:network/nw:network-types" {
  description
    "Defines the service topologyies types";
  container network-service-type {
    presence "Indicates Network service topology";
    leaf network-service-type {
      type identityref {
        base vpn-common:service-type;
      }
      description
        "The presence identifies the network service type,
```

```
        e.g., L3VPN, L2VPN, etc.";
    }
    description
        "Container for vpn service type.";
}
}

augment "/nw:networks/nw:network" {
    when 'nw:network-types/nvp:network-service-type' {
        description
            "Augment only for VPN Network topology.";
    }
    description
        "Augment the network with service topology attributes";
    container vpn-topo-attributes {
        leaf vpn-id {
            type vpn-common:vpn-id;
            description
                "Pointer to the parent VPN service(e.g., L3NM),
                if any.";
        }
        leaf vpn-topology {
            type identityref {
                base vpn-common:vpn-topology;
            }
            description
                "VPN service topology, e.g., hub-spoke, any-to-any,
                hub-spoke-disjoint";
        }
        description
            "Container for vpn topology attributes.";
    }
}

augment "/nw:networks/nw:network/nw:node" {
    when '../nw:network-types/nvp:network-service-type' {
        description
            "Augment only for VPN Network topology.";
    }
    description
        "Augment the network node with service topology attributes";
    container node-attributes {
        leaf node-type {
            type identityref {
                base vpn-common:role;
            }
            description
                "Node type, e.g., PE, P, ASBR.";
        }
    }
}
```

```
    }
    leaf site-id {
      type string;
      description
        "Associated vpn site";
    }
    leaf site-role {
      type identityref {
        base vpn-common:role;
      }
      default "vpn-common:any-to-any-role";
      description
        "Role of the site in the VPN.";
    }
  }
  description
    "Container for service topology attributes.";
}
uses vpn-summary-statistics;
}

augment "/nw:networks/nw:network/nt:link" {
  when '../nw:network-types/nvp:network-service-type' {
    description
      "Augment only for VPN Network topology.";
  }
  description
    "Augment the network topology link with service topology
    attributes";
  leaf link-type {
    type identityref {
      base vpn-common:protocol-type;
    }
    description
      "Underlay-transport type, e.g., GRE, LDP, etc.";
  }
}

augment "/nw:networks/nw:network/nt:link" {
  description
    "Augment the network topology link with service topology
    attributes";
  leaf low-percentile {
    type percentile;
    default "10.00";
    description
      "Low percentile to report. Setting low-percentile
      into 0.00 indicates the client is not interested in receiving
      low percentile.";
  }
}
```

```
    }
    leaf middle-percentile {
      type percentile;
      default "50.00";
      description
        "Middle percentile to report. Setting middle-percentile
         into 0.00 indicates the client is not interested in receiving
         middle percentile.";
    }
    leaf high-percentile {
      type percentile;
      default "90.00";
      description
        "High percentile to report. Setting high-percentile
         into 0.00 indicates the client is not interested in receiving
         high percentile";
    }
    leaf reference-time {
      type yang:date-and-time;
      description
        "The time that the current Measurement Interval started.";
    }
    leaf measurement-interval {
      type uint32;
      units "seconds";
      default "60";
      description
        "Interval to calculate performance metric.";
    }
    container link-telemetry-attributes {
      config false;
      uses link-error-statistics;
      uses link-delay-statistics;
      uses link-jitter-statistics;
      description
        "Container for service telemetry attributes.";
    }
  }

augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
  description
    "Augment the network topology termination point with vpn
     service attributes";
  container tp-telemetry-attributes {
    config false;
    uses tp-svc-telemetry;
    description
      "Container for termination point service telemetry attributes.";
  }
}
```

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

8. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol [RFC6241]. The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. 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 NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

- o /nw:networks/nw:network/svc-topo:svc-telemetry-attributes
- o /nw:networks/nw:network/nw:node/svc-topo:node-attributes

9. IANA Considerations

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

```
URI: urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.
```

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

Name: ietf-network-vpn-pm
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm
Maintained by IANA: N
Prefix: nvp
Reference: RFC XXXX

10. Acknowledgements

Thanks to Joe Clarke, Adrian Farrel, Greg Mirsky, Roque Gagliano, Erez Segev for reviewing this draft and providing important input to this document.

11. Contributors

Michale Wang
Huawei
Email: wangzitao@huawei.com

Roni Even
Huawei
Email: ron.even.tlv@gmail.com

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <<https://www.rfc-editor.org/info/rfc3393>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<https://www.rfc-editor.org/info/rfc6374>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [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>>.

- [RFC8532] Kumar, D., Wang, Z., Wu, Q., Ed., Rahman, R., and S. Raghavan, "Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8532, DOI 10.17487/RFC8532, April 2019, <<https://www.rfc-editor.org/info/rfc8532>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.

12.2. Informative References

- [I-D.ietf-opsawg-l3sm-l3nm]
barguil, s., Dios, O., Boucadair, M., Munoz, L., and A. Aguado, "A Layer 3 VPN Network YANG Model", draft-ietf-opsawg-l3sm-l3nm-05 (work in progress), October 2020.
- [I-D.ietf-opsawg-model-automation-framework]
WU, Q., Boucadair, M., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", draft-ietf-opsawg-model-automation-framework-10 (work in progress), October 2020.
- [I-D.ietf-opsawg-vpn-common]
barguil, s., Dios, O., Boucadair, M., and Q. WU, "A Layer 2/3 VPN Common YANG Model", draft-ietf-opsawg-vpn-common-03 (work in progress), January 2021.
- [RFC4176] El Mghazli, Y., Ed., Nadeau, T., Boucadair, M., Chan, K., and A. Gonguet, "Framework for Layer 3 Virtual Private Networks (L3VPN) Operations and Management", RFC 4176, DOI 10.17487/RFC4176, October 2005, <<https://www.rfc-editor.org/info/rfc4176>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [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>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.

- [RFC8233] Dhody, D., Wu, Q., Manral, V., Ali, Z., and K. Kumaki, "Extensions to the Path Computation Element Communication Protocol (PCEP) to Compute Service-Aware Label Switched Paths (LSPs)", RFC 8233, DOI 10.17487/RFC8233, September 2017, <<https://www.rfc-editor.org/info/rfc8233>>.
- [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>>.
- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/info/rfc8570>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.

Authors' Addresses

Bo Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: lane.wubo@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Oscar Gonzalez de Dios
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Bin Wen
Comcast

Email: bin_wen@comcast.com

Change Liu
China Unicom

Email: liuc131@chinaunicom.cn

Honglei Xu
China Telecom

Email: xuhl.bri@chinatelecom.cn