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:

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

Table of Contents

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
             |  ...

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

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 L3SM [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:

- vpn-description:

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

- vpn-profile-cfg:

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

- status-timestamp:

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

- 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.

- svc-transport-encapsulation:

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

- rt-rd:

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

- vpn-node-group:

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

Identities

- bw-direction:Identity for the bandwidth direction.
- qos-profile-direction:Base identity for QoS profile direction.
- customer-application:Base identity for customer application.
- ie-type:Defines Import-Export routing profiles.
- site-network-access-type:Base identity for site-network-access type.
- operational-status:Base identity for the operational status.
- administrative-status:Base identity for administrative status.
- encapsulation-type:Base identity for encapsulation type.
- tag-type:Base identity from which all tag types are derived.
- protocol-type:Base identity for Protocol Type.
- vpn-topology:Base identity for VPN topology.
- role:Base identity for site or node type.
- vpn-signaling-type:Identity of VPN signaling types
- service-type:Identity of service type.
- vxlan-peer-mode:Base identity for the VXLAN peer mode.
- multicast-gp-address-mapping:Identity for multicast group mapping type.
- multicast-tree-type:Base identity for multicast tree type.
- 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/&gt;
        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 urllc {
     base customer-application;
     description
       "Identity for an Ultra-Reliable and Low Latency
        Communications (URLLC) application.  Note that a
        URLLC application demands network performance
        with a wide variety of characteristics, such as latency,
        reliability, and many other parameters.";
   }

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

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

   identity 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 aouto-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 assigment.";
    }
    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: July 6, 2021

B. Claise
Cisco Systems, Inc.
J. Quilbeuf
Independent
D. Lopez
Telefonica I+D
D. Voyer
Bell Canada
T. Arumugam
Cisco Systems, Inc.
January 2, 2021

Service Assurance for Intent-based Networking Architecture draft-claise-opsawg-service-assurance-architecture-04

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 July 6, 2021.

Table of Contents

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:

- Subservice Expressions: expression graph representing all the computations to execute for a subservice.
- Service Expressions: expression graph representing all the computations to execute for a service instance, i.e. including the computations for all dependent subservices.
- 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+ [I-D.ietf-opsawg-tacacs], 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.

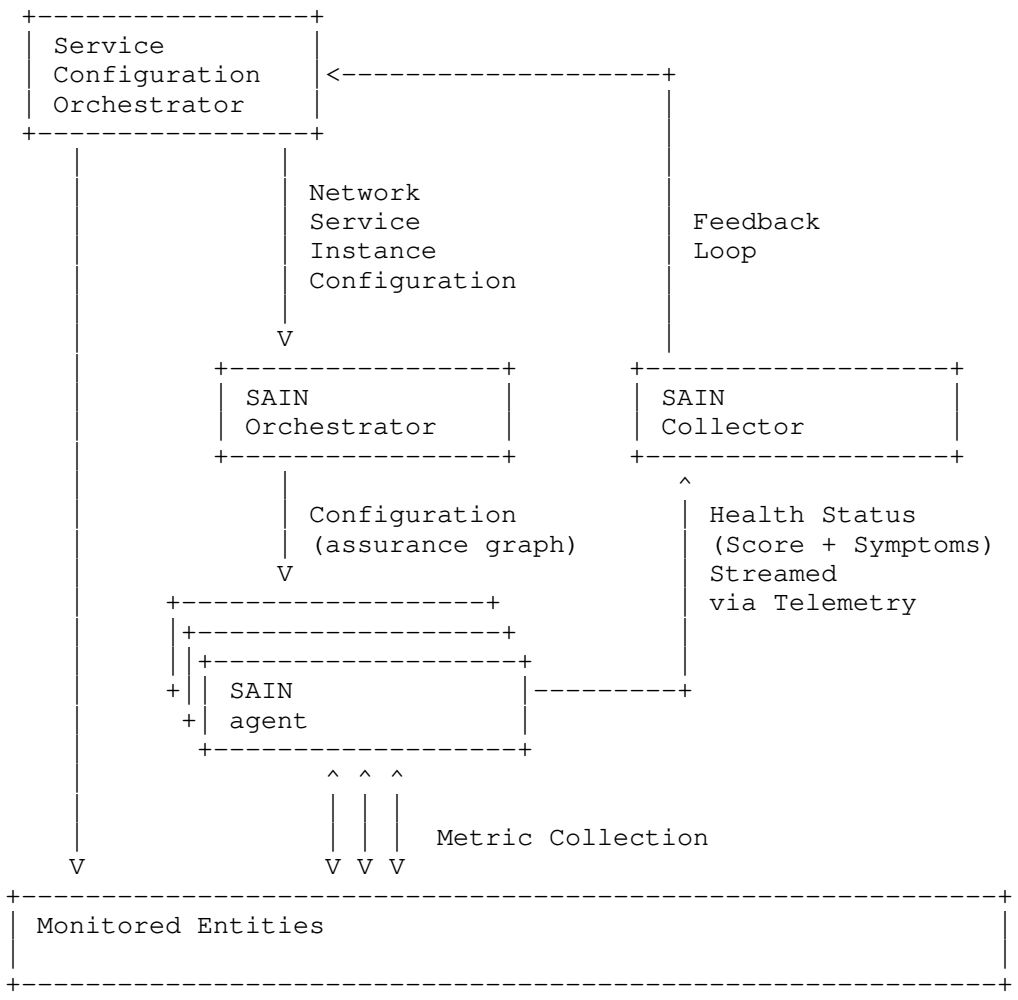


Figure 1: SAIN Architecture

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

- 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.

- Stream (via telemetry [RFC8641]) operational and config metric values when possible, else continuously poll.

- 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.

- TThe 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.

- 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.

- 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.

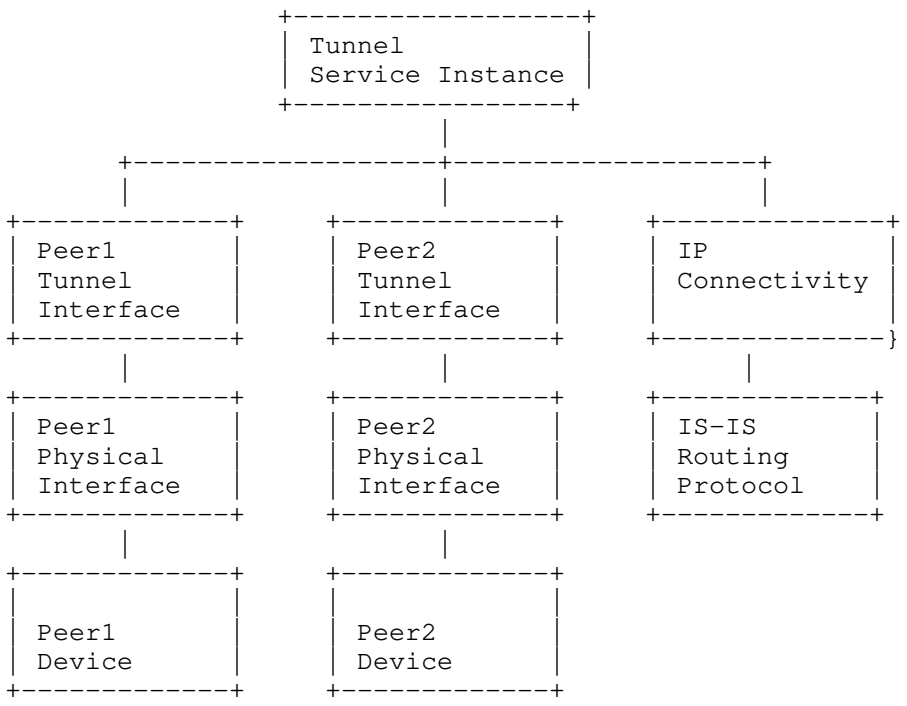


Figure 2: Assurance Graph Example

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:

- Try to capture the intent of the service instance, i.e. what is the service instance trying to achieve,

- 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:

- 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.

- 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:

- 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.
- 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 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:

- 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.
- 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.

- 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 Orchesttrator, 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:

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

- Youssef El Fathi
- 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.

[I-D.ietf-opsawg-tacacs]
Dahm, T., Ota, A., dcmgash@cisco.com, d., Carrel, D., and L. Grant, "The TACACS+ Protocol", draft-ietf-opsawg-tacacs-18 (work in progress), March 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>.

Appendix A. Changes between revisions

v02 - v03

- Timing Concepts
- New Assurance Graph Generation

v01 - v02

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

v00 - v01

- Terminology clarifications
- Figure 1 improved

Acknowledgements

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

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.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

Email: tarumuga@cisco.com

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: July 6, 2021

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

January 2, 2021

YANG Modules for Service Assurance
draft-claise-opsawg-service-assurance-yang-06

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 July 6, 2021.

Copyright Notice

Table of Contents

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:

- machine readable
- vendor independent
- 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:

- 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 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:

- 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".

- 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:

- A type: identity inheriting of the base identity for subservice,

- An id: string uniquely identifying the subservice among those with the same identity,

- 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 ce 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
```

```
            for easy consumption.";
          uses symptom;
        }
      }
      container dependencies {
        description
          "configure the dependencies between the subservices, along with their t
ypes.";
        list dependency {
          key "type id";
          description
            "List of soft dependencies of the subservice.";
          uses subservice-dependency;
        }
      }
    }
  }
}
```

<CODE ENDS>

5. Subservice Extension: ietf-service-assurance-device YANG module

5.1. Tree View

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

```
module: ietf-service-assurance-device
  augment /service-assurance:subservices/service-assurance:subservice/service-ass
urance:parameter:
    +--rw device-idty
       +--rw device?   string
```

5.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance and ietf-service-assurance-device 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
        +--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:jquilbeu@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-ass
urance: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:subservice/service-ass
urance: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
        |  +--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
```

6.3. Concepts

For our InterafaceHealthy 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-ass
urance: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-ass
urance: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
ntityref
        +--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:jquilbeu@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-ass
urance: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-idt
y')";
           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:

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

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

v03 - v04

- Add the interface subservice, with two parameters

v02 - v03

- Added the maintenace window concepts

v01 - v02

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

v00 - v01

- Terminology clarifications
- 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
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

S. Barguil, Ed.
O. Gonzalez de Dios, Ed.
Telefonica
M. Boucadair
Orange
L. Munoz
Vodafone
L. Jalil
Verizon
J. Ma
China Unicom

November 02, 2020

A Layer 2 VPN Network YANG Model
draft-ietf-opsawg-l2nm-01

Abstract

This document defines a YANG Data model (called, L2NM) that can be used to manage the provisioning of Layer 2 VPN services within a Service Provider Network. This YANG module provides representation of the Layer 2 VPN Service from a network standpoint. The module is meant to be used by a Network Controller to derive the configuration information that will be sent to relevant network devices.

The L2NM YANG Data model complements the Layer 2 Service Model (RFC8466) by providing a network-centric view of the service that is internal to a Service Provider.

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 May 6, 2021.

Table of Contents

1. Introduction

[RFC8466] defines a L2VPN Service Model (L2SM) YANG data model that can be used for L2VPN service ordering matters between customers and Service Providers (SPs). This document complements the L2SM model by creating a network-centric view of the service which can be exposed by a Network to a Service Controller within the Service Provider Network. In particular, the model can be used in the communication between the entity that interacts directly with the customer, the service orchestrator, (either fully automated or a human operator) and the entity in charge of network orchestration and control (a.k.a., network controller/orchestrator).

The data model defined in this document is called the L2VPN Network Model (L2NM), playing the role of Service Delivery Model (Figure 3 of [RFC8466]. The module supports additional capabilities, such as exposing operational parameters, transport protocols selection and precedence. It also serves as a multi-domain orchestration interface, because this model can transport resources (i.e., VCID) between domains. The data model keeps minimum customer-related information.

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

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

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8466], [RFC8309], and [RFC8453] and uses terminology from those documents. The meaning of the symbols in YANG tree diagrams is [RFC8340].

This document makes use of the following terms:

- L2 VPN Customer Service Model (L2SM): Describes the service characterization (requirements) of a L2 VPN that interconnects a set of sites from the perspective of the customer. The customer service model does not provide details on the Service Provider Network. The L2 VPN Customer Service model is defined in [RFC8466].

- L2 VPN Service Network Model (L2NM): Refers to the YANG module that describes a L2 VPN Service with a network-centric view. It contains information of the Service Provider network and might include allocated resources. It can be used by network

controllers to manage the Layer 2 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 or to expose the list of active L2VPN services.

- Service Orchestrator: Refers to a functional entity that interacts with the customer of a L2 VPN relying upon, e.g. L2SM. The Service Orchestrator is responsible of the CE-PE attachment circuits, the PE selection, and requesting the activation of the L2 VPN service to a network controller.

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

- VPN node (vpn-node): Is an abstraction that represents a set of policies applied on a PE and that belong to a single VPN service (vpn-service). A VPN service involves one or more VPN nodes. The VPN node will identify the Service Provider node on which the VPN is deployed.

- VPN network access (vpn-network-access): Is 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.

- VPN Service Provider (SP): Is a Service Provider that offers VPN-related services.

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

1.2. Requirements Language

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

2. Reference architecture

Figure 1 illustrates how 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".

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".

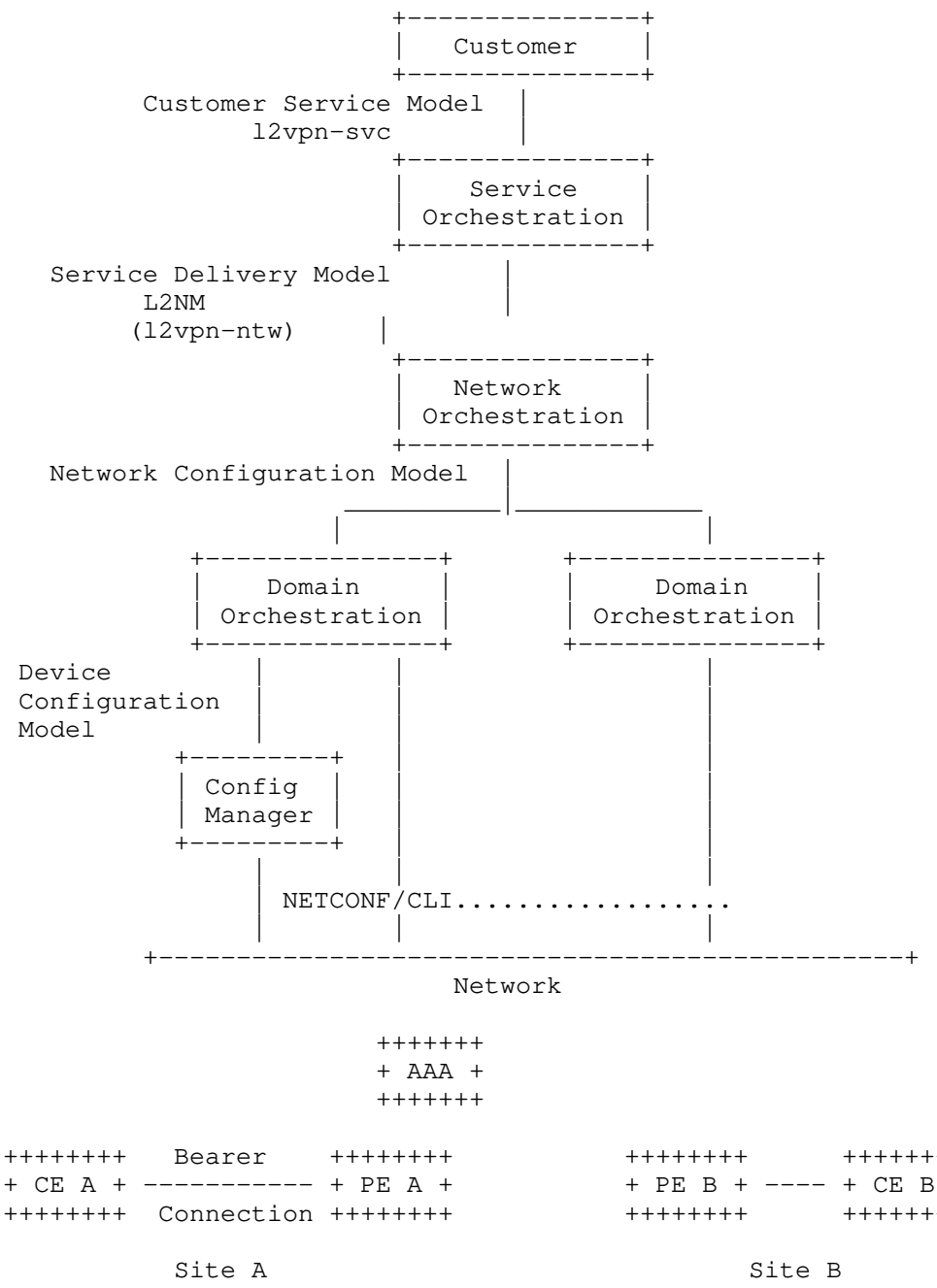


Figure 1: L2SM and L2NM Interaction

Figure 2 shows how L2SM and L2NM may be used in the context of the ACTN architecture [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC). It shows the interfaces between these functional units: the CNC-MDSC Interface (CMI), the MDSC-PNC Interface (MPI), and the Southbound Interface (SBI).

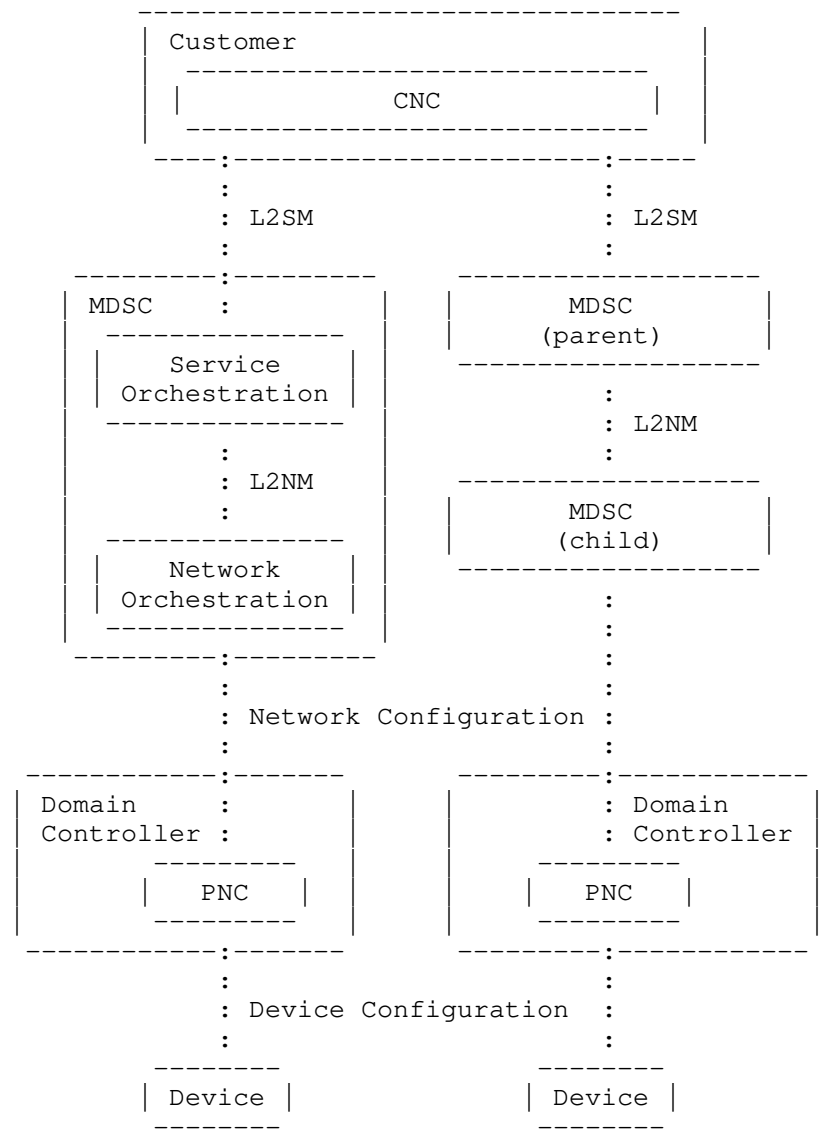


Figure 2: L2SM and L2NM in the Context of ACTN

3. Description of the L2NM YANG Module

The L2NM module ('ietf-l2vpn-ntw') is meant to manage L2 VPNs within a service provider network. In particular, the 'ietf-l2vpn-ntw' module can be used to create, modify, and retrieve L2VPN Services in a Network Controller. The module is not aimed at maintaining customer-related information.

Editor's note: Next version of the document will include the full description of the parameters. When the parameters match with L2SM, the exact reference will be done

3.1. Structure of the Module

The 'ietf-l2vpn-ntw' module uses two main containers: 'vpn-services' and 'vpn-profiles'. The 'vpn-services' container maintains a set of L2 VPN Services managed in the service provider network. The module allows to create a new l2 VPN service by adding a new instance of 'vpn-service'. The 'vpn-service' is the data structure that abstracts the VPN Service.

```
module: ietf-l2vpn-ntw
+--rw l2vpn-ntw
 +--rw vpn-profiles
 |     .......
 +--rw vpn-services
     +--rw vpn-service* [vpn-id]
       ........
```

Figure 3: Simplified L2NM Tree Structure

3.2. VPN Profiles

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

This document does not make any assumption about the exact definition of these profiles. How such profiles are defined is deployment specific. The model only includes an identifier to these profiles to ease identifying local policies when building a VPN service. As shown in Figure 4, the following identifiers can be included:

- 'cloud-identifier': This identifier refers to a cloud service.

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

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

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

- 'forwarding-profile-identifier': A forwarding profile refers to the policies that apply to the forwarding of packets conveyed within a VPN. Such policies may consist at applying Access Control Lists (ACLs).

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

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

Figure 4: VPN Profiles Subtree Structure

3.3. L2 VPN Service

The 'vpn-service' is the data structure that abstracts a L2 VPN Service within the SP Network. Every 'vpn-service' has a unique identifier: vpn-id. Such vpn-id is only meaningful locally within the Network controller. In order to facilitate the recognition of the service, a 'customer-name' and a 'description' may be included.

The topology of the VPN service is expressed in the 'vpn-service-topology' leaf.

A VPN Service is built by adding instances of 'vpn-node' to the 'vpn-nodes' container. The 'vpn-node' 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'. Note that, as this is a network data model, the information about customers site is not needed. Such information, is only relevant in the L2SM model.

```
+--rw vpn-services
   +--rw vpn-service* [vpn-id]
      +--rw status
      |  +--rw admin-status
      |  |  +--rw status?         identityref
      |  |  +--rw last-updated?   yang:date-and-time
      |  +--ro oper-status
      |     +--ro status?         identityref
      |     +--ro last-updated?   yang:date-and-time
      +--rw vpn-id                      vpn-id
      +--rw vpn-name?                   string
      +--rw vpn-description?            string
      +--rw customer-name?              string
      +--rw l2sm-vpn-id?                vpn-common:vpn-id
      +--rw vpn-svc-type?               identityref
      +--rw svc-topo?                   identityref
      +--rw multicast-like {vpn-common:multicast}?
      |  +--rw enabled?                 boolean
      |  +--rw customer-tree-flavors
      |     +--rw tree-flavor*   identityref
      +--rw extranet-vpns {vpn-common:extranet-vpn}?
      |  +--rw extranet-vpn* [vpn-id]
      |     +--rw vpn-id              vpn-common:vpn-id
      |     +--rw local-sites-role?   identityref
      +--rw svc-mtu?                    uint32
      +--rw ce-vlan-preservation?       boolean
      +--rw ce-vlan-cos-perservation?   boolean
      +--rw underlay-transport
      |  +--rw type*   identityref
      +--rw vpn-nodes
             .....
```

Figure 5

3.3.1. L2 VPN Service Types

The L2 VPN Service types directly matches with the L2VPN Service types defined in Section 5.1.3 of [RFC8466]:

- Point-to-point VPWSs.
- Point-to-point or point-to-multipoint VPWSs [RFC8214].
- Multipoint VPLSs.
- Multipoint VPLSs connecting one or more root sites and a set of leaf sites but preventing inter-leaf-site communication.
- EVPN services [RFC7432].
- EVPN VPWSs between two customer sites or a set of customer sites as specified in [RFC8214].

3.3.2. Underlying Transport Selection

The model enables network operators to select the type of transport protocol underlay. Also, in scenarios with multiple domains and NNI types, the selection of the transport protocol underlay is required. The Service Provider Network might have several underlay possibilities available. If no underlay transport protocol is specified, the Network Controller will take care of the transport decision. The following options are supported in the "underlay-transport" container:

LDP: MPLS with LDP (martini encapsulation).

GRE: A mesh of GRE tunnels is established between vpn-nodes.

BGP: BGP tunnels (kompella encapsulation) are preferred to route traffic between VPN nodes.

TE: TE tunnels (either RSVP-TE or SR) are prefered. The mapping details will be specified in draft-ietf-te-service-mapping.

SR: Non-TE SR is preferred to route traffic.

3.3.3. VPN Node

The 'vpn-node' 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 vpn-nodes
     +--rw vpn-node* [vpn-node-id ne-id]
        +--rw vpn-node-id             vpn-common:vpn-id
        +--rw description?            string
        +--rw node-role?              identityref
        +--rw ne-id                   string
        +--rw port-id?                string
        +--rw status
        |  +--rw admin-status
        |  |  +--rw status?         identityref
        |  |  +--rw last-updated?   yang:date-and-time
        |  +--ro oper-status
        |     +--ro status?         identityref
        |     +--ro last-updated?   yang:date-and-time
        +--rw signaling-options* [type]
        |  +--rw type         identityref
        |  +--rw l2vpn-bgp
        |  |  +--rw pwe-encapsulation-type?   identityref
        |  |  +--rw vpn-target* [id]
        |  |  |  +--rw id                    int8
        |  |  |  +--rw route-targets* [route-target]
        |  |  |  |  +--rw route-target
        |  |  |  |           rt-types:route-target
        |  |  |  +--rw route-target-type
        |  |  |           rt-types:route-target-type
        |  |  +--rw vpn-policies
        |  |  |  +--rw import-policy?   string
        |  |  |  +--rw export-policy?   string
        |  |  +--rw pwe-mtu
        |  |  |  +--rw allow-mtu-mismatch?   boolean
        |  |  +--rw address-family?
        |  |           vpn-common:address-family
        |  +--rw evpn-bgp
        |  |  +--rw vpn-id?              leafref
        |  |  +--rw type?                identityref
        |  |  +--rw address-family?
        |  |  |       vpn-common:address-family
        |  |  +--rw mac-learning-mode?   identityref
        |  |  +--rw arp-suppress?        boolean
        |  +--rw t-ldp-pwe
        |  |  +--rw type?                       identityref
        |  |  +--rw pwe-encapsulation-type?   identityref
        |  |  +--rw pwe-mtu?                  uint16
        |  |  +--rw ac-pw-list* [peer-addr vc-id]
        |  |  |  +--rw peer-addr      inet:ip-address
```

```
|  |  |  +--rw vc-id          vpn-common:vpn-id
|  |  |  +--rw pw-type?       identityref
|  |  |  +--rw pw-priority?   uint32
|  |  +--rw qinq
|  |     +--rw s-tag?   uint32
|  |     +--rw c-tag?   uint32
|  +--rw l2tp-pwe
|     +--rw type?                  identityref
|     +--rw encapsulation-type?    identityref
|     +--rw ac-pw-list* [peer-addr vc-id]
|        +--rw peer-addr      inet:ip-address
|        +--rw vc-id          string
|        +--rw pw-priority?   uint32
+--rw vpn-network-accesses
```

Figure 6

3.3.3.1. Signaling options

This sub-tree defines the L2VPN service type, according to the several signalling options to exchange membership information between the PE that is used. There are some common parameters inside each of them (e.g encapsulation type, MTU) but some others are:

l2vpn-bgp. The service is a Multipoint VPLSs that use a BGP control plane as described in [RFC4761] and [RFC6624]. The VPLS members exchange Route Targets with related import/export policies.

evpn-bgp. The service is a Multipoint VPLSs that use also a BGP control plane but also includes the additional features and related parameters described in [RFC7432] and [RFC7209].

t-ldp-pwe. A Multipoint VPLSs that use a mesh of LDP-signaled Pseudowires [RFC6074], including as parameters the list of Pseudowires that constitute the mesh, with their details (VC-IDs and endpoints).

L2tp-pwe. Multipoint VPLSs that use L2TP-signaled Pseudowires [RFC6074].

```
+--rw signaling-options* [type]
  +--rw type         identityref
  +--rw l2vpn-bgp
  |  +--rw pwe-encapsulation-type?   identityref
  |  +--rw vpn-target* [id]
  |  |  +--rw id                    int8
  |  |  +--rw route-targets* [route-target]
  |  |  |  +--rw route-target
  |  |  |          rt-types:route-target
  |  |  +--rw route-target-type
  |  |           rt-types:route-target-type
  |  +--rw vpn-policies
  |  |  +--rw import-policy?   string
  |  |  +--rw export-policy?   string
  |  +--rw pwe-mtu
  |  |  +--rw allow-mtu-mismatch?   boolean
  |  +--rw address-family?
  |           vpn-common:address-family
  +--rw evpn-bgp
  |  +--rw vpn-id?              leafref
  |  +--rw type?                identityref
  |  +--rw address-family?
  |  |       vpn-common:address-family
  |  +--rw mac-learning-mode?   identityref
  |  +--rw arp-suppress?        boolean
  +--rw t-ldp-pwe
  |  +--rw type?                       identityref
  |  +--rw pwe-encapsulation-type?   identityref
  |  +--rw pwe-mtu?                    uint16
  |  +--rw ac-pw-list* [peer-addr vc-id]
  |  |  +--rw peer-addr      inet:ip-address
  |  |  +--rw vc-id          vpn-common:vpn-id
  |  |  +--rw pw-type?       identityref
  |  |  +--rw pw-priority?   uint32
  |  +--rw qinq
  |     +--rw s-tag?   uint32
  |     +--rw c-tag?   uint32
  +--rw l2tp-pwe
     +--rw type?                 identityref
     +--rw encapsulation-type?   identityref
     +--rw ac-pw-list* [peer-addr vc-id]
        +--rw peer-addr      inet:ip-address
        +--rw vc-id          string
        +--rw pw-priority?   uint32
```

Figure 7

3.3.3.2. VPN Network Access

A 'vpn-network-access' 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. As such, every 'vpn-network-access' MUST belong to one and only one 'vpn-node'.

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

The Site Network Access is comprised of:

id: Identifier of the vpn network access.

description: Text describing the vpn network access.

interface-mtu: maximum transmission unit or maximum frame size of the interface belonging to the vpn network access. When a frame is larger than the MTU, it is broken down, or fragmented, into smaller pieces by the network protocol to accommodate the MTU of the network"

status: Administrative and operational status of the service.

ethernet-service-oam: Carries information about the service OAM

```
          +--rw vpn-network-accesses
       +--rw vpn-network-access* [id]
          +--rw id
          |       vpn-common:vpn-id
          +--rw description?
          |       string
          +--rw Interface-mtu?
          |       uint32
          +--rw status
          |  +--rw admin-status
          |  |  +--rw status?         identityref
          |  |  +--rw last-updated?   yang:date-and-time
          |  +--ro oper-status
          |     +--ro status?         identityref
          |     +--ro last-updated?   yang:date-and-time
          +--rw access-diversity
          |       {vpn-common:placement-diversity}?
          |  +--rw groups
          |  |  +--rw fate-sharing-group-size?   uint16
          |  |  +--rw group-color?               string
```

```
|  |  +--rw group* [group-id]
|  |     +--rw group-id    string
|  +--rw constraints
|     +--rw constraint* [constraint-type]
|        +--rw constraint-type    identityref
|        +--rw target
|           +--rw (target-flavor)?
|              +--:(id)
|              |  +--rw group* [group-id]
|              |     +--rw group-id    string
|              +--:(all-accesses)
|              |  +--rw all-other-accesses?
|              |          empty
|              +--:(all-groups)
|                 +--rw all-other-groups?
|                         empty
+--rw connection
|       ....
+--rw availability
|  +--rw access-priority?       uint32
|  +--rw (redundancy-mode)?
|     +--:(single-active)
|     |  +--rw single-active?   boolean
|     +--:(all-active)
|        +--rw all-active?      boolean
+--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?
|           uint32
+--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?
|   |   |           vpn-common: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?
|   |               vpn-common:ccm-priority-type
|   +--rw y-1731* [maid]
|      +--rw maid
|      |       string
|      +--rw mep-id?
|      |       uint32
|      +--rw 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
+--rw mac-loop-prevention
|  +--rw frequency?         uint32
|  +--rw protection-type?   identityref
|  +--rw number-retries?    uint32
+--rw access-control-list
|  +--rw mac* [mac-address]
```

```
                        |     +--rw mac-address    yang:mac-address
                        +--rw mac-addr-limit
                           +--rw mac-num-limit?   uint16
                           +--rw time-interval?   uint32
                           +--rw action?          identityref
```

Figure 8

3.3.3.2.1. Connection

The connection container is used to configure the relevant properties of the interface that is attached to the VPN, for example the encapsulation type, the physical interface or creating a lag.

```
                  +--rw connection
                     +--rw encapsulation-type?    identityref
                     +--rw eth-inf-type*          identityref
                     +--rw dot1q-interface
                     |  +--rw l2-access-type?   identityref
                     |  +--rw dot1q {vpn-common:dot1q}?
                     |  |  +--rw physical-inf?   string
                     |  |  +--rw c-vlan-id?      uint32
                     |  +--rw qinq {vpn-common:qinq}?
                     |  |  +--rw s-vlan-id?   uint32
                     |  |  +--rw c-vlan-id?   uint32
                     |  +--rw qinany {vpn-common:qinany}?
                     |  |  +--rw s-vlan-id?   uint32
                     |  +--rw vxlan {vxlan}?
                     |     +--rw vni-id?      uint32
                     |     +--rw peer-mode?   identityref
                     |     +--rw peer-list* [peer-ip]
                     |        +--rw peer-ip    inet:ip-address
                     +--rw phy-interface
                     |  +--rw port-number?           uint32
                     |  +--rw port-speed?            uint32
                     |  +--rw mode?
                     |  |       vpn-common:neg-mode
                     |  +--rw phy-mtu?               uint32
                     |  +--rw flow-control?          string
                     |  +--rw oam-802.3ah-link {oam-3ah}?
                     |  |  +--rw enable?   boolean
                     |  +--rw uni-loop-prevention?   boolean
                     +--rw lag-interface
                     |       {vpn-common:lag-interface}?
                     |  +--rw lag-interface*
                     |          [lag-interface-number]
                     |     +--rw lag-interface-number    uint32
                     |     +--rw lacp
```

```
                        |        +--rw lacp-state?         boolean
                        |        +--rw lacp-mode?          boolean
                        |        +--rw lacp-speed?         boolean
                        |        +--rw mini-link?          uint32
                        |        +--rw system-priority?    uint16
                        |        +--rw member-link-list
                        |        |  +--rw member-link* [name]
                        |        |     +--rw name
                        |        |     |       string
                        |        |     +--rw port-speed?
                        |        |     |       uint32
                        |        |     +--rw mode?
                        |        |     |       vpn-common:neg-mode
                        |        |     +--rw link-mtu?
                        |        |     |       uint32
                        |        |     +--rw oam-802.3ah-link
                        |        |             {oam-3ah}?
                        |        |        +--rw enable?   boolean
                        |        +--rw flow-control?       string
                        |        +--rw lldp?               boolean
                        +--rw cvlan-id-to-svc-map* [svc-id]
                        |  +--rw svc-id      leafref
                        |  +--rw cvlan-id* [vid]
                        |     +--rw vid    uint32
                        +--rw split-horizon
                           +--rw group-name?   string
```

Figure 9

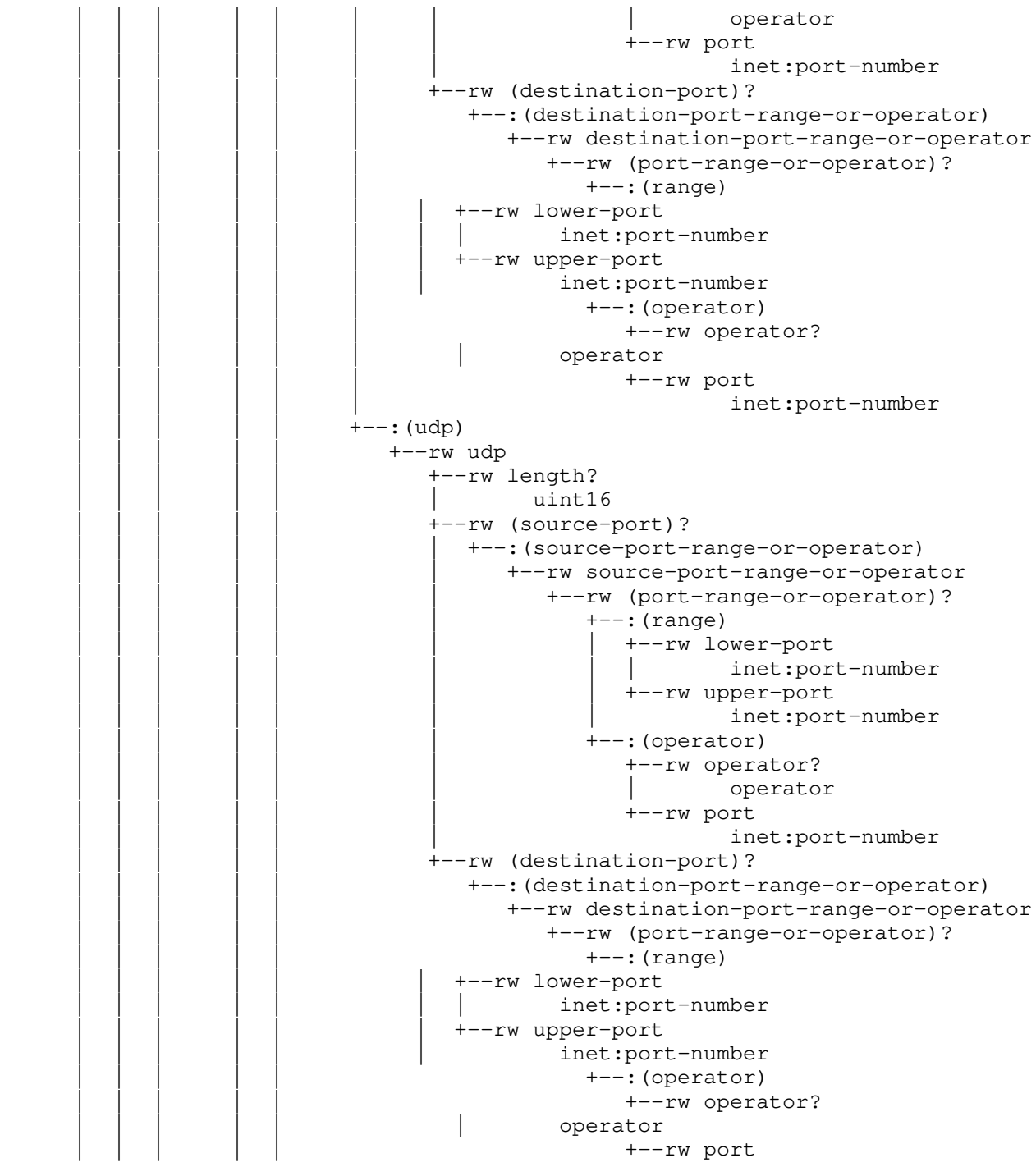
3.3.3.2.2. Layer 2 service requirements

This container is usedd to indicate the details of the ethernet service such as bandwidth or qos.

```
+--rw service
   |  +--rw svc-input-bandwidth
   |  |       {vpn-common:input-bw}?
   |  |  +--rw input-bandwidth* [type]
   |  |     +--rw type      identityref
   |  |     +--rw cos-id?   uint8
   |  |     +--rw cir?      uint64
   |  |     +--rw cbs?      uint64
   |  |     +--rw eir?      uint64
   |  |     +--rw ebs?      uint64
   |  |     +--rw pir?      uint64
   |  |     +--rw pbs?      uint64
   |  +--rw svc-output-bandwidth {output-bw}?
   |  |  +--rw output-bandwidth* [type]
```

```
   |  |      +--rw type      identityref
   |  |      +--rw cos-id?   uint8
   |  |      +--rw cir?      uint64
   |  |      +--rw cbs?      uint64
   |  |      +--rw eir?      uint64
   |  |      +--rw ebs?      uint64
   |  |      +--rw pir?      uint64
   |  |      +--rw pbs?      uint64
   |  +--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
+--rw (l4)?
+--:(tcp)
+--rw tcp
+--rw sequence-number?
uint32
+--rw acknowledgement-number?
uint32
+--rw data-offset?
uint8
+--rw reserved?
uint8
+--rw flags?
bits
+--rw window-size?
uint16
+--rw urgent-pointer?
uint16
+--rw options?
binary
+--rw (source-port)?
+--:(source-port-range-or-operator)
+--rw source-port-range-or-operator
+--rw (port-range-or-operator)?
+--:(range)
+--rw lower-port
inet:port-number
+--rw upper-port
inet:port-number
+--:(operator)
+--rw operator?


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

```
|  |  |  |     |  |                                    inet:port-number
|  |  |  |     |  +--:(match-application)
|  |  |  |     |     +--rw match-application?
|  |  |  |     |              identityref
|  |  |  |     +--rw target-class-id?
|  |  |  |               string
|  |  |  +--rw qos-profile
|  |  |     +--rw qos-profile* [profile]
|  |  |        +--rw profile      leafref
|  |  |        +--rw direction?   identityref
|  |  +--rw precedence
|  |     +--rw precedence?   identityref
```

Figure 10

4. Relation with other YANG Models

The L2NM model, aimed at managing the L2VPN Services in a Service Provider Network controller/orchestrator has relations with other YANG modules.

4.1. Relation with L2SM

[RFC8466] defines a L2VPN Service YANG data Model (L2SM) that can be used for communication between customers and VPN service providers. Hence, the model provides inputs to the Network Operator to deliver such service to the customer. Hence, most parts of the model can be directly mapped into L2NM.

- Service requirements: The service requirements can be directly taken from L2SM to L2NM.

- Sites: The sites from L2SM are used to select the Service Prodider node. The site information is NOT maintained in L2NM

4.2. Relation with Network Topology

The L2NM model manages VPN Services running over Service Provider Backbone network. The set of nodes over which it is possible to deploy a L2 VPN Service MAY be part of the topology contained in an ietf-network module.

4.3. Relation with Device Models

Creating services in the l2vpn-ntw module will will lead at some point to the configuration of devices. Hence, it is foreseen that the data for the device yang modules will be derived partially from

the L2NM vpn-service container. Note that L2NM is NOT a device model.

5. YANG Module

```
<CODE BEGINS>file "ietf-l2vpn-ntw@2020-11-02.yang"
module ietf-l2vpn-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw";
  prefix l2vpn-ntw;

  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC CCCC: A Layer 2/3 VPN Common YANG Model";
  }

  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>
     Author:    Mohamed Boucadair
                <mailto:mohamed.boucadair@orange.com>
     Author:    Luis Angel Munoz
                <mailto:luis-angel.munoz@vodafone.com>
     Author:    Luay Jalil
                <mailto:luay.jalil@verizon.com>
     Author:    Jichun Ma
                <mailto:majc16@chinaunicom.cn>
    ";
  description
    "The YANG module defines a generic network configuration
```

```
     model for Layer 2 VPN services common across all of the
     vendor implementations.

     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 2020-11-02 {
    description
      "Initial version.";
    reference
      "RFC XXXX: A Layer 2 VPN Network YANG Model.";
  }

  /* Features */

  feature multicast-like {
    description
      "Indicates the support of multicast-like capabilities
       in a L2VPN.";
  }

  feature target-sites {
    description
      "Indicates the support of 'target-sites' match flow
       parameter.";
  }

  feature l2cp-control {
    description
      "Indicates the support of L2CP control.";
  }

  feature output-bw {
    description
      "Indicates the support of Output Bandwidth in
       a VPN";
  }
```

```
  feature uni-list {
    description
      "Indicates thesupport of UNI list in a VPN.";
  }

  feature oam-3ah {
    description
      "Indicates the support of OAM 802.3ah.";
  }

  feature micro-bfd {
    description
      "Indicates the support of Micro-BFD.";
  }

  feature signaling-options {
    description
      "Indicates the support of signalling option.";
  }

  feature always-on {
    description
      "Indicates the support for always-on access
       constraint.";
  }

  feature requested-type {
    description
      "Indicates the support for requested-type access
       constraint.";
  }

  feature vlan {
    description
      "Indicates the support of VLAN.";
  }

  feature sub-inf {
    description
      "Indicates the support of Sub Interface.";
  }

  feature atm {
    description
      "Indicates the support of ATM.";
  }

  feature vxlan {
```

```
    description
      "Indicates the support of VxLAN.";
  }

  feature lan-tag {
    description
      "Indicates the LAN Tag support in a VPN.";
  }

  /* Typedefs */
  /* Identities */

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

  identity protection-mode {
    description
      "Identity of protection mode";
  }

  identity oneplusone {
    base protection-mode;
    description
      "In this scheme, the primary circuit will be
       protected by a backup circuit, typically meeting certain
       diverse path/fiber/site/node criteria.  Both primary and
       protection circuits are provisioned to be in the active
       forward ing state. The subscriber may choose to send the
       same service frames across both circuits simultaneously.";
  }

  identity one-to-one {
    base protection-mode;
    description
      "In this scheme, a backup circuit to the primary
       circuit is provisioned.  Depending on the implementation
       agreement, the protection circuits may either always be
       in active forwarding state, or may only become active when
       a faulty state is detected on the primary circuit.";
  }

  identity bundling-type {
    description
      "The base identity for the bundling type.  It supports
       multiple CE-VLANs associated with an L2VPN service or
```

```
       all CE-VLANs associated with an L2VPN service.";
  }

  identity multi-svc-bundling {
    base bundling-type;
    description
      "Identity for multi-service bundling, i.e.,
       multiple CE-VLAN IDs can be associated with an
       L2VPN service at a site.";
  }

  identity one2one-bundling {
    base bundling-type;
    description
      "Identity for one-to-one service bundling, i.e.,
       each L2VPN can be associated with only one CE-VLAN ID
       at a site.";
  }

  identity all2one-bundling {
    base bundling-type;
    description
      "Identity for all-to-one bundling, i.e., all CE-VLAN IDs
       are mapped to one L2VPN service.";
  }

  identity color-id {
    description
      "Base identity of the color ID.";
  }

  identity color-id-cvlan {
    base color-id;
    description
      "Identity of the color ID based on a CVLAN.";
  }

  identity color-type {
    description
      "Identity of color types.";
  }

  identity green {
    base color-type;
    description
      "Identity of the 'green' color type.";
  }
```

```
  identity yellow {
    base color-type;
    description
      "Identity of the 'yellow' color type.";
  }

  identity red {
    base color-type;
    description
      "Identity of the 'red' color type.";
  }

  identity perf-tier-opt {
    description
      "Identity of performance tier option.";
  }

  identity metro {
    base perf-tier-opt;
    description
      "Identity of metro";
  }

  identity regional {
    base perf-tier-opt;
    description
      "Identity of regional";
  }

  identity continental {
    base perf-tier-opt;
    description
      "Identity of continental";
  }

  identity global {
    base perf-tier-opt;
    description
      "Identity of global";
  }

  identity policing {
    description
      "Identity of policing type";
  }

  identity one-rate-two-color {
    base policing;
```

```
    description
      "Identity of one-rate, two-color (1R2C)";
  }

  identity two-rate-three-color {
    base policing;
    description
      "Identity of two-rate, three-color (2R3C)";
  }

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

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

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

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

  identity vpws-type {
    base t-ldp-pwe-type;
    description
      "Identity for VPWS";
  }

  identity vpls-type {
    base t-ldp-pwe-type;
    description
      "Identity for vpls";
  }

  identity hvpls {
    base t-ldp-pwe-type;
    description
      "Identity for h-vpls";
```

```
  }

  identity l2vpn-type {
    description
      "Layer 2 VPN types";
  }

  identity l2vpn-vpws {
    base l2vpn-type;
    description
      "VPWS L2VPN type.";
  }

  identity l2vpn-vpls {
    base l2vpn-type;
    description
      "VPLS L2VPN type.";
  }

  identity distribute-vpls {
    base l2vpn-type;
    description
      "distribute VPLS L2VPN type.";
  }

  identity evpn-type {
    description
      "Ethernet VPN types";
  }

  identity evpn-vpws {
    base evpn-type;
    description
      "VPWS support in EVPN.";
  }

  identity evpn-pbb {
    base evpn-type;
    description
      " Provider Backbone Bridging Support in EVPN.";
  }

  identity pm-type {
    description
      "Performance-monitoring type.";
  }

  identity loss {
```

```
    base pm-type;
    description
      "Loss measurement.";
  }

  identity delay {
    base pm-type;
    description
      "Delay measurement.";
  }

  identity mac-learning-mode {
    description
      "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
      "Identity for dropping a packet.";
  }

  identity flood {
    base mac-action;
    description
      "Identity for packet flooding.";
  }

  identity warning {
    base mac-action;
    description
```

```
      "Identity for sending a warning log message.";
  }

  identity load-balance-method {
    description
      "Base identity for load balance method.";
  }

  identity fat-pw {
    base load-balance-method;
    description
      "Identity for Fat PW. Fat label is
       applied to Pseudowires across MPLS
       network.";
  }

  identity entropy-label {
    base load-balance-method;
    description
      "Identity for entropy label.Entropy label
       is applied to IP forwarding,
       L2VPN or L3VPN across MPLS network";
  }

  identity vxlan-source-port {
    base load-balance-method;
    description
      "Identity for vxlan source port.VxLAN
       Source Port is one load balancing method.";
  }

  identity precedence-type {
    description
      "Redundancy type. The service can be created
       with active and bakcup signalization.";
  }

  identity primary {
    base precedence-type;
    description
      "Identifies the Main L2VPN.";
  }

  identity backup {
    base precedence-type;
    description
      "Identifies the Backup L2VPN.";
  }
```

```
  /* Groupings */

  grouping cfm-802-grouping {
    leaf maid {
      type string;
      description
        "MA ID";
    }
    leaf mep-id {
      type uint32;
      description
        "Local MEP ID";
    }
    leaf mep-level {
      type uint32;
      description
        "MEP level";
    }
    leaf mep-up-down {
      type enumeration {
        enum up {
          description
            "MEP up";
        }
        enum down {
          description
            "MEP down";
        }
      }
      description
        "MEP up/down";
    }
    leaf remote-mep-id {
      type uint32;
      description
        "Remote MEP ID";
    }
    leaf cos-for-cfm-pdus {
      type uint32;
      description
        "COS for CFM PDUs";
    }
    leaf ccm-interval {
      type uint32;
      description
        "CCM interval";
    }
    leaf ccm-holdtime {
```

```
      type uint32;
      description
        "CCM hold time";
    }
    leaf ccm-p-bits-pri {
      type vpn-common:ccm-priority-type;
      description
        "The priority parameter for CCMs transmitted by the MEP";
    }
    description
      "Grouping for 802.1ag CFM attribute";
  }

  grouping y-1731 {
    list y-1731 {
      key "maid";
      leaf maid {
        type string;
        description
          "MA ID ";
      }
      leaf mep-id {
        type uint32;
        description
          "Local MEP ID";
      }
      leaf type {
        type identityref {
          base pm-type;
        }
        description
          "Performance monitor types";
      }
      leaf remote-mep-id {
        type uint32;
        description
          "Remote MEP ID";
      }
      leaf message-period {
        type uint32;
        description
          "Defines the interval between OAM messages. The message
           period is expressed in milliseconds";
      }
      leaf measurement-interval {
        type uint32;
        description
          "Specifies the measurement interval for statistics. The
```

```
          measurement interval is expressed in seconds";
      }
      leaf cos {
        type uint32;
        description
          "Class of service";
      }
      leaf loss-measurement {
        type boolean;
        description
          "Whether enable loss measurement";
      }
      leaf synthethic-loss-measurement {
        type boolean;
        description
          "Indicate whether enable synthetic loss measurement";
      }
      container delay-measurement {
        leaf enable-dm {
          type boolean;
          description
            "Whether to enable delay measurement";
        }
        leaf two-way {
          type boolean;
          description
            "Whether delay measurement is two-way (true) of one-
             way (false)";
        }
        description
          "Container for delay measurement";
      }
      leaf frame-size {
        type uint32;
        description
          "Frame size";
      }
      leaf session-type {
        type enumeration {
          enum proactive {
            description
              "Proactive mode";
          }
          enum on-demand {
            description
              "On demand mode";
          }
        }
```

```
        description
          "Session type";
      }
      description
        "List for y-1731.";
    }
    description
      "Grouping for y.1731";
  }

  /* MAIN L2VPN SERVICE */

  container l2vpn-ntw {
    container vpn-profiles {
      uses vpn-common:vpn-profile-cfg;
      description
        "Container for VPN Profiles.";
    }
    container vpn-services {
      list vpn-service {
        key "vpn-id";
        uses vpn-common:service-status;
        uses vpn-common:vpn-description;
        leaf l2sm-vpn-id {
          type vpn-common:vpn-id;
          description
            "Pointer to the L2SM service.";
        }
        leaf vpn-svc-type {
          type identityref {
            base vpn-common:vpn-signaling-type;
          }
          description
            "Service type";
        }
        leaf svc-topo {
          type identityref {
            base vpn-common:vpn-topology;
          }
          description
            "Defining service topology, such as
             any-to-any, hub-spoke, etc.";
        }
        container multicast-like {
          if-feature "vpn-common:multicast";
          leaf enabled {
            type boolean;
            default "false";
```

```
            description
              "Enables multicast.";
          }
          container customer-tree-flavors {
            leaf-list tree-flavor {
              type identityref {
                base vpn-common:multicast-tree-type;
              }
              description
                "Type of tree to be used.";
            }
            description
              "Type of trees used by customer.";
          }
          description
            "Multicast like container";
        }
        container extranet-vpns {
          if-feature "vpn-common:extranet-vpn";
          list extranet-vpn {
            key "vpn-id";
            leaf vpn-id {
              type vpn-common:vpn-id;
              description
                "Identifies the target VPN.";
            }
            leaf local-sites-role {
              type identityref {
                base vpn-common:role;
              }
              default "vpn-common:any-to-any-role";
              description
                "This describes the role of the
                 local sites in the target VPN topology.";
            }
            description
              "List of extranet VPNs the local VPN is attached to.";
          }
          description
            "Container for extranet VPN configuration.";
        }
        leaf svc-mtu {
          type uint32;
          description
            "SVC MTU, it is also known as the maximum transmission unit
             or maximum frame size,When a frame is larger than the MTU,
             it is broken down, or fragmented, into smaller pieces by the
             network protocol to accommodate the MTU of the network";
```

```
        }
        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 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.";
        }
        leaf ce-vlan-cos-perservation {
          type boolean;
          description
            "CE vlan CoS preservation. 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.";
        }
        uses vpn-common:svc-transport-encapsulation;
        container vpn-nodes {
          list vpn-node {
            key "vpn-node-id ne-id";
            leaf vpn-node-id {
              type vpn-common:vpn-id;
              description
                "";
            }
            leaf description {
              type string;
              description
                "Textual description of a VPN node.";
            }
            leaf node-role {
              type identityref {
                base vpn-common:role;
              }
              default "vpn-common:any-to-any-role";
              description
                "Role of the vpn-node in the IP VPN.";
            }
            leaf ne-id {
              type string;
              description
                "NE IP address";
            }
            leaf port-id {
```

```
              type string;
              description
                "NE Port-id";
            }
            uses vpn-common:service-status;
            list signaling-options {
              key "type";
              leaf type {
                type identityref {
                  base vpn-common:vpn-signaling-type;
                }
                description
                  "VPN signaling types";
              }
              container l2vpn-bgp {
                when "/l2vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/sign
aling-options/type = 'vpn-common:l2vpn-bgp'" {
                  description
                    "Only applies when vpn signaling type is l2vpn
                     BGP protocol.";
                }
                leaf pwe-encapsulation-type {
                  type identityref {
                    base vpn-common:encapsulation-type;
                  }
                  description
                    "PWE Encapsulation Type";
                }
                uses vpn-common:vpn-route-targets;
                container pwe-mtu {
                  leaf allow-mtu-mismatch {
                    type boolean;
                    description
                      "Allow MTU mismatch";
                  }
                  description
                    "Container of PWE MTU configurations";
                }
                leaf address-family {
                  type vpn-common:address-family;
                  description
                    "Address family used for router-id information.";
                }
                description
                  "Container for MP BGP L2VPN";
              }
              container evpn-bgp {
                when "/l2vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/sign
aling-options/type = 'vpn-common:evpn-bgp'" {
                  description
```

```
                    "Only applies when vpn signaling type is EVPN
                     BGP protocol.";
                  }
                  leaf vpn-id {
                    type leafref {
                      path "/l2vpn-ntw/vpn-services/vpn-service/vpn-id";
                    }
                    description
                      "Identifies the target EVPN";
                  }
                  leaf type {
                    type identityref {
                      base evpn-type;
                    }
                    description
                      "L2VPN types";
                  }
                  leaf address-family {
                    type vpn-common:address-family;
                    description
                      "Address family used for router-id information.";
                  }
                  leaf mac-learning-mode {
                    type identityref {
                      base mac-learning-mode;
                    }
                    description
                      "Indicates through which plane MAC addresses are
                       advertised.";
                  }
                  leaf arp-suppress {
                    type boolean;
                    default "false";
                    description
                      "Indicates whether to suppress ARP broadcast.";
                  }
                  description
                    "Container for MP BGP L2VPN";
                }
                container t-ldp-pwe {
                  when "/l2vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/sign
aling-options/type = 'vpn-common:t-ldp'" {
                    description
                      "Only applies when vpn signaling type is Target LDP.";
                  }
                  leaf type {
                    type identityref {
                      base t-ldp-pwe-type;
                    }
```

```
                  description
                    "T-LDP PWE type";
                }
                leaf pwe-encapsulation-type {
                  type identityref {
                    base vpn-common:encapsulation-type;
                  }
                  description
                    "PWE Encapsulation Type.";
                }
                leaf pwe-mtu {
                  type uint16;
                  description
                    "Allow MTU mismatch";
                }
                list ac-pw-list {
                  key "peer-addr vc-id";
                  leaf peer-addr {
                    type inet:ip-address;
                    description
                      "Peer IP address.";
                  }
                  leaf vc-id {
                    type vpn-common:vpn-id;
                    description
                      "VC lable used to identify PW.";
                  }
                  leaf pw-type {
                    type identityref {
                      base vpn-common:vpn-topology;
                    }
                    description
                      "PW topology type";
                  }
                  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.";
                  }
                  description
                    "List of AC and PW bindings.";
                }
                container qinq {
                  when "/l2vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/si
gnaling-options/type = 'vpn-common:h-vpls'" {
                    description
                      "Only applies when t-ldp pwe type is h-vpls.";
```

```
                  }
                  leaf s-tag {
                    type uint32;
                    description
                      "S-TAG";
                  }
                  leaf c-tag {
                    type uint32;
                    description
                      "C-TAG";
                  }
                  description
                    "Container for QinQ";
                }
                description
                  "Container of T-LDP PWE configurations";
              }
              container l2tp-pwe {
                when "/l2vpn-ntw/vpn-services/vpn-service/vpn-nodes/vpn-node/sign
aling-options/type = 'vpn-common:l2tp'" {
                  description
                    "Applies when vpn signaling type is L2TP protocol.";
                }
                leaf type {
                  type identityref {
                    base t-ldp-pwe-type;
                  }
                  description
                    "T-LDP PWE type";
                }
                leaf encapsulation-type {
                  type identityref {
                    base vpn-common:encapsulation-type;
                  }
                  description
                    "Encapsulation type";
                }
                list ac-pw-list {
                  key "peer-addr vc-id";
                  leaf peer-addr {
                    type inet:ip-address;
                    description
                      "Peer IP address.";
                  }
                  leaf vc-id {
                    type string;
                    description
                      "VC lable used to identify PW.";
                  }
```

```
            leaf pw-priority {
              type uint32;
              description
                "PW priority";
            }
            description
              "List of AC and PW bindings.";
          }
          description
            "Container for l2tp pw";
        }
        description
          "List of VPN Signaling Option.";
      }
      container vpn-network-accesses {
        list vpn-network-access {
          key "id";
          leaf id {
            type vpn-common:vpn-id;
            description
              "Identifier of network access";
          }
          leaf description {
            type string;
            description
              "String to describe the element.";
          }
          leaf Interface-mtu {
            type uint32;
            description
              "Interface MTU, it is also known as the maximum
               transmission unit or maximum frame size. When a
               frame is larger than the MTU, it is broken down,
               or fragmented, into smaller pieces by the
               network protocol to accommodate the MTU of the
               network";
          }
          uses vpn-common:service-status;
          container access-diversity {
            if-feature "vpn-common:placement-diversity";
            container groups {
              leaf fate-sharing-group-size {
                type uint16;
                description
                  "Fate sharing group size.";
              }
              leaf group-color {
                type string;
```

```
        description
          "Group color associated with a particular VPN.";
      }
      list group {
        key "group-id";
        leaf group-id {
          type string;
          description
            "Group-id the site network access
             is belonging to";
        }
        description
          "List of group-id";
      }
      description
        "Groups the fate sharing group member
         is belonging to";
    }
    container constraints {
      list constraint {
        key "constraint-type";
        leaf constraint-type {
          type identityref {
            base vpn-common:placement-diversity;
          }
          description
            "Diversity constraint type.";
        }
        container target {
          choice target-flavor {
            case id {
              list group {
                key "group-id";
                leaf group-id {
                  type string;
                  description
                    "The constraint will apply
                     against this particular
                     group-id";
                }
                description
                  "List of groups";
              }
            }
            case all-accesses {
              leaf all-other-accesses {
                type empty;
                description
```

```
                        "The constraint will apply
                         against all other site network
                         access of this site";
                    }
                  }
                  case all-groups {
                    leaf all-other-groups {
                      type empty;
                      description
                        "The constraint will apply
                         against all other groups the
                         customer is managing";
                    }
                  }
                  description
                    "Choice for the group definition";
                }
                description
                  "The constraint will apply against
                   this list of groups";
              }
              description
                "List of constraints";
            }
            description
              "Constraints for placing this site
                network access";
          }
          description
            "Diversity parameters.";
        }
        container connection {
          leaf encapsulation-type {
            type identityref {
              base vpn-common:encapsulation-type;
            }
            description
              "Encapsulation Type";
          }
          leaf-list eth-inf-type {
            type identityref {
              base vpn-common:encapsulation-type;
            }
            description
              "Ethernet Interface Type";
          }
          container dot1q-interface {
            leaf l2-access-type {
```

```
        type identityref {
          base vpn-common:encapsulation-type;
        }
        description
          "L2 Access Encapsulation Type";
      }
      container dot1q {
        when "../l2-access-type='vpn-common:dot1q'";
        if-feature "vpn-common:dot1q";
        leaf physical-inf {
          type string;
          description
            "Physical Interface";
        }
        leaf c-vlan-id {
          type uint32;
          description
            "VLAN identifier";
        }
        description
          "Qot1q";
      }
      container qinq {
        when "../l2-access-type='vpn-common:qinq'";
        if-feature "vpn-common:qinq";
        leaf s-vlan-id {
          type uint32;
          description
            "S-VLAN Identifier";
        }
        leaf c-vlan-id {
          type uint32;
          description
            "C-VLAN Identifier";
        }
        description
          "QinQ";
      }
      container qinany {
        if-feature "vpn-common:qinany";
        leaf s-vlan-id {
          type uint32;
          description
            "S-Vlan ID";
        }
        description
          "Container for Q in Any";
      }
```

```
      container vxlan {
        when "../l2-access-type='vpn-common:vxlan'";
        if-feature "vxlan";
        leaf vni-id {
          type uint32;
          description
            "VNI Identifier";
        }
        leaf peer-mode {
          type identityref {
            base vpn-common:vxlan-peer-mode;
          }
          description
            "specify the vxlan access mode";
        }
        list peer-list {
          key "peer-ip";
          leaf peer-ip {
            type inet:ip-address;
            description
              "Peer IP";
          }
          description
            "List for peer IP";
        }
        description
          "QinQ";
      }
      description
        "Container for dot1Q Interface";
    }
    container phy-interface {
      leaf port-number {
        type uint32;
        description
          "Port number";
      }
      leaf port-speed {
        type uint32;
        description
          "Port speed";
      }
      leaf mode {
        type vpn-common:neg-mode;
        description
          "Negotiation mode";
      }
      leaf phy-mtu {
```

```
            type uint32;
            description
              "PHY MTU";
          }
          leaf flow-control {
            type string;
            description
              "Flow control";
          }
          container oam-802.3ah-link {
            if-feature "oam-3ah";
            leaf enable {
              type boolean;
              description
                "Indicate whether support oam 802.3 ah link";
            }
            description
              "Container for oam 802.3 ah link.";
          }
          leaf uni-loop-prevention {
            type boolean;
            description
              "If this leaf set to truth that the port automatically
               goes down when a physical loopback is detect.";
          }
          description
            "Container of PHY Interface Attributes configurations";
        }
        container lag-interface {
          if-feature "vpn-common:lag-interface";
          list lag-interface {
            key "lag-interface-number";
            leaf lag-interface-number {
              type uint32;
              description
                "LAG interface number";
            }
            container lacp {
              leaf lacp-state {
                type boolean;
                description
                  "LACP on/off";
              }
              leaf lacp-mode {
                type boolean;
                description
                  "LACP mode";
              }
```

```
leaf lacp-speed {
  type boolean;
  description
    "LACP speed";
}
leaf mini-link {
  type uint32;
  description
    "The minimum aggregate bandwidth for a LAG";
}
leaf system-priority {
  type uint16;
  description
    "Indicates the LACP priority for the system.
     The range is from 0 to 65535.
     The default is 32768.";
}
container member-link-list {
  list member-link {
    key "name";
    leaf name {
      type string;
      description
        "Member link name";
    }
    leaf port-speed {
      type uint32;
      description
        "Port speed";
    }
    leaf mode {
      type vpn-common:neg-mode;
      description
        "Negotiation mode";
    }
    leaf link-mtu {
      type uint32;
      description
        "Link MTU size.";
    }
    container oam-802.3ah-link {
      if-feature "oam-3ah";
      leaf enable {
        type boolean;
        description
          "Indicate whether support oam 802.3 ah link";
      }
      description
```

```
                        "Container for oam 802.3 ah link.";
                      }
                      description
                        "Member link";
                    }
                    description
                      "Container of Member link list";
                  }
                  leaf flow-control {
                    type string;
                    description
                      "Flow control";
                  }
                  leaf lldp {
                    type boolean;
                    description
                      "LLDP";
                  }
                  description
                    "LACP";
                }
                description
                  "List of LAG interfaces";
              }
              description
                "Container of LAG interface attributes configuration";
            }
            list cvlan-id-to-svc-map {
              key "svc-id";
              leaf svc-id {
                type leafref {
                  path "/l2vpn-ntw/vpn-services/vpn-service/vpn-id";
                }
                description
                  "VPN Service identifier";
              }
              list cvlan-id {
                key "vid";
                leaf vid {
                  type uint32;
                  description
                    "CVLAN ID";
                }
                description
                  "List of CVLAN-ID to SVC Map configurations";
              }
              description
                "List for cvlan-id to L2VPn Service map configurations";
```

```
        }
        container split-horizon {
          leaf group-name {
            type string;
            description
              "group-name of the Split Horizon";
          }
          description
            "Configuration with split horizon enabled";
        }
        description
          "Container for bearer";
      }
      container availability {
        leaf access-priority {
          type uint32;
          description
            "Access priority";
        }
        choice redundancy-mode {
          case single-active {
            leaf single-active {
              type boolean;
              description
                "Single active";
            }
            description
              "Single active case";
          }
          case all-active {
            leaf all-active {
              type boolean;
              description
                "All active";
            }
            description
              "All active case";
          }
          description
            "Redundancy mode choice";
        }
        description
          "Container of availability optional configurations";
      }
      container service {
        container svc-input-bandwidth {
          if-feature "vpn-common:input-bw";
          list input-bandwidth {
```

```
                    key "type";
                    leaf type {
                      type identityref {
                        base vpn-common:bw-type;
                      }
                      description
                        "Bandwidth Type";
                    }
                    leaf cos-id {
                      type uint8;
                      description
                        "Identifier of Class of Service
                         , indicated by DSCP or a CE-CLAN
                         CoS(802.1p)value in the service frame.";
                    }
                    leaf cir {
                      type uint64;
                      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;
                      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;
                      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;
                      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;
                      description
```

```
                    "Peak Information Rate, i.e., maixmum frame delivery
                     allowed. It is equal to or less than sum of cir and
                     eir.";
                }
                leaf pbs {
                  type uint64;
                  description
                    "Peak Burst Size. It is measured in bytes per second.";
                }
                description
                  "List for input bandwidth";
              }
              description
                "From the PE perspective, the service input
                 bandwidth of the connection.";
            }
            container svc-output-bandwidth {
              if-feature "output-bw";
              list output-bandwidth {
                key "type";
                leaf type {
                  type identityref {
                    base vpn-common:bw-type;
                  }
                  description
                    "Bandwidth Type";
                }
                leaf cos-id {
                  type uint8;
                  description
                    "Identifier of Class of Service
                     , indicated by DSCP or a CE-CLAN
                     CoS(802.1p)value in the service frame.";
                }
                leaf cir {
                  type uint64;
                  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;
                  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;
              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;
              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;
              description
                "Peak Information Rate, i.e., maixmum frame delivery
                 allowed. It is equal to or less than sum of cir and
                 eir.";
            }
            leaf pbs {
              type uint64;
              description
                "Peak Burst Size. It is measured in bytes per second.";
            }
            description
              "List for output bandwidth";
          }
          description
            "From the PE perspective, the service output
             bandwidth of the connection.";
        }
        container qos {
          if-feature "vpn-common:qos";
          container qos-classification-policy {
            uses vpn-common:qos-classification-policy;
            description
              "Configuration of the traffic classification
               policy.";
          }
          container qos-profile {
            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.";
                  }
                }
                description
                  "QoS profile configuration.";
              }
              description
                "QoS configuration.";
            }
            container precedence {
              leaf precedence {
                type identityref {
                  base precedence-type;
                }
                description
                  "Defining service redundancy in transport
                   network.";
              }
              description
                "Transport netowrk precedence selector
                 Primary or Secondary tunnel.";
            }
            description
              "Container for service";
          }
          container broadcast-unknown-unicast-multicast {
            leaf multicast-site-type {
              type enumeration {
                enum receiver-only {
                  description
```

```
                        "The site only has receivers.";
                    }
                    enum source-only {
                      description
                        "The site only has sources.";
                    }
                    enum source-receiver {
                      description
                        "The site has both sources and receivers.";
                    }
                  }
                  default "source-receiver";
                  description
                    "Type of multicast site.";
                }
                list multicast-gp-address-mapping {
                  key "id";
                  leaf id {
                    type uint16;
                    description
                      "Unique identifier for the mapping.";
                  }
                  leaf vlan-id {
                    type uint32;
                    description
                      "The VLAN ID of the Multicast group.";
                  }
                  leaf mac-gp-address {
                    type yang:mac-address;
                    description
                      "The MAC address of the Multicast group.";
                  }
                  leaf port-lag-number {
                    type uint32;
                    description
                      "The ports/LAGs belonging to the Multicast group.";
                  }
                  description
                    "List of Port to group mappings.";
                }
                leaf bum-overall-rate {
                  type uint32;
                  description
                    "overall rate for BUM";
                }
                description
                  "Container of broadcast, unknown unicast, and multicast
                   configurations";
```

```
        }
        container ethernet-service-oam {
          leaf md-name {
            type string;
            description
              "Maintenance domain name";
          }
          leaf md-level {
            type uint8;
            description
              "Maintenance domain level";
          }
          container cfm-802.1-ag {
            list n2-uni-c {
              key "maid";
              uses cfm-802-grouping;
              description
                "List of UNI-N to UNI-C";
            }
            list n2-uni-n {
              key "maid";
              uses cfm-802-grouping;
              description
                "List of UNI-N to UNI-N";
            }
            description
              "Container of 802.1ag CFM configurations.";
          }
          uses y-1731;
          description
            "Container for Ethernet service OAM.";
        }
        container mac-loop-prevention {
          leaf frequency {
            type uint32;
            description
              "Frequency";
          }
          leaf protection-type {
            type identityref {
              base loop-prevention-type;
            }
            description
              "Protection type";
          }
          leaf number-retries {
            type uint32;
            description
```

```
                    "Number of retries";
                  }
                  description
                    "Container of MAC loop prevention.";
                }
                container access-control-list {
                  list mac {
                    key "mac-address";
                    leaf mac-address {
                      type yang:mac-address;
                      description
                        "MAC address.";
                    }
                    description
                      "List for MAC.";
                  }
                  description
                    "Container for access control List.";
                }
                container mac-addr-limit {
                  leaf mac-num-limit {
                    type uint16;
                    description
                      "maximum number of MAC addresses learned from
                       the subscriber for a single service instance.";
                  }
                  leaf time-interval {
                    type uint32;
                    units "milliseconds";
                    description
                      "The aging time of the mac address.";
                  }
                  leaf action {
                    type identityref {
                      base mac-action;
                    }
                    description
                      "specify the action when the upper limit is
                       exceeded: drop the packet, flood the
                       packet, or simply send a warning log message.";
                  }
                  description
                    "Container of MAC-Addr limit configurations";
                }
                description
                  "List of VPN Network Accesses.";
              }
              description
```

```
                "List of VPN Nodes.";
            }
            description
              "Container of VPN Nodes.";
          }
          description
            "List of vpn-svc";
        }
        description
          "Container of port configurations";
      }
      description
        "Container for L2VPN service";
    }
    description
      "Container for VPN services.";
  }
}

<CODE ENDS>
```

Figure 11

6. Acknowledgements

The authors would like to thank Tom Petch for the comments to improve the document.

7. Contributors

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

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

Zhang Guiyu
China Unicom
Email: zhanggy113@chinaunicom.cn

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

8. 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-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 module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

name: ietf-l2vpn-ntw

namespace: urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw

maintained by IANA: N

prefix: l2vpn-ntw

reference: RFC XXXX

9. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040] . The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8466].

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

The ietf-l2vpn-ntw module is used to manage L2 VPNs in a service provider backbone network. Hence, the module can be used to request, modify, or retrieve L2VPN services. There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes MAY be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a

negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability in the ietf-l2vpn-ntw module:

- vpn-service: An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L2 VPN Service, interrupting all the traffic of a client.

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

- customer-name: 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.

10. References

10.1. Normative References

[I-D.ietf-opsawg-vpn-common]
barguil, s., Dios, O., Boucadair, M., and Q. WU, "A Layer 2/3 VPN Common YANG Model", draft-ietf-opsawg-vpn-common-02 (work in progress), October 2020.

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

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

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

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

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

[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>.

[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>.

[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>.

[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>.

[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>.

10.2. Informative References

[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>.

[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>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <https://www.rfc-editor.org/info/rfc8453>.

Authors' Addresses

Samier Barguil (editor)
Telefonica
Madrid
ES

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair
Orange
France

Email: "mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone
ES

Email: luis-angel.munoz@vodafone.com

Luay Jalil
Verizon
USA

Email: luay.jalil@verizon.com

Jichun Ma
China Unicom
China

Email: majc16@chinaunicom.cn

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2021

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

October 16, 2020

A Layer 3 VPN Network YANG Model
draft-ietf-opsawg-l3sm-l3nm-05

Abstract

This document defines a L3VPN Network YANG Model (L3NM) that can be used to manage the provisioning of Layer 3 Virtual Private Network (VPN) services within a Service Provider's network. The model provides a network-centric view of L3VPN services.

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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 April 19, 2021.

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

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

2. Requirements Language

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

3. Terminology

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

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

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

This document makes use of the following terms:

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

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

- Service Orchestrator: A functional entity that interacts with the customer of a L3VPN. The Service Orchestrator interacts with the customer using L3SM. The Service Orchestrator is responsible of the Customer Edge (CE) - the 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 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.

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

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

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

4. L3NM Reference Architecture

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

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

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

The terminology from [RFC8309] is introduced to show the distinction between the "Customer Service Model", the "Service Delivery Model", the "Network Configuration Model", and the "Device Configuration Model". In that context, the "Domain Orchestration" and "Config Manager" roles may be performed by "Controllers".

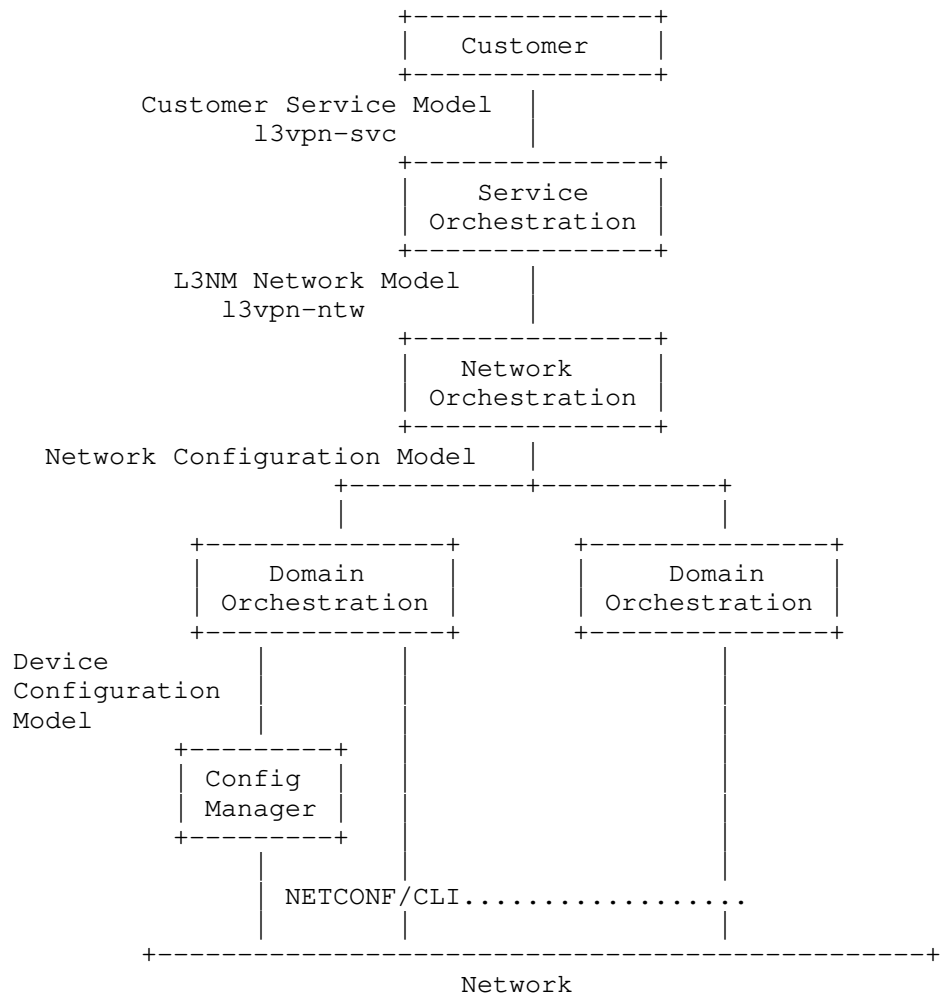


Figure 1: Reference Architecture

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

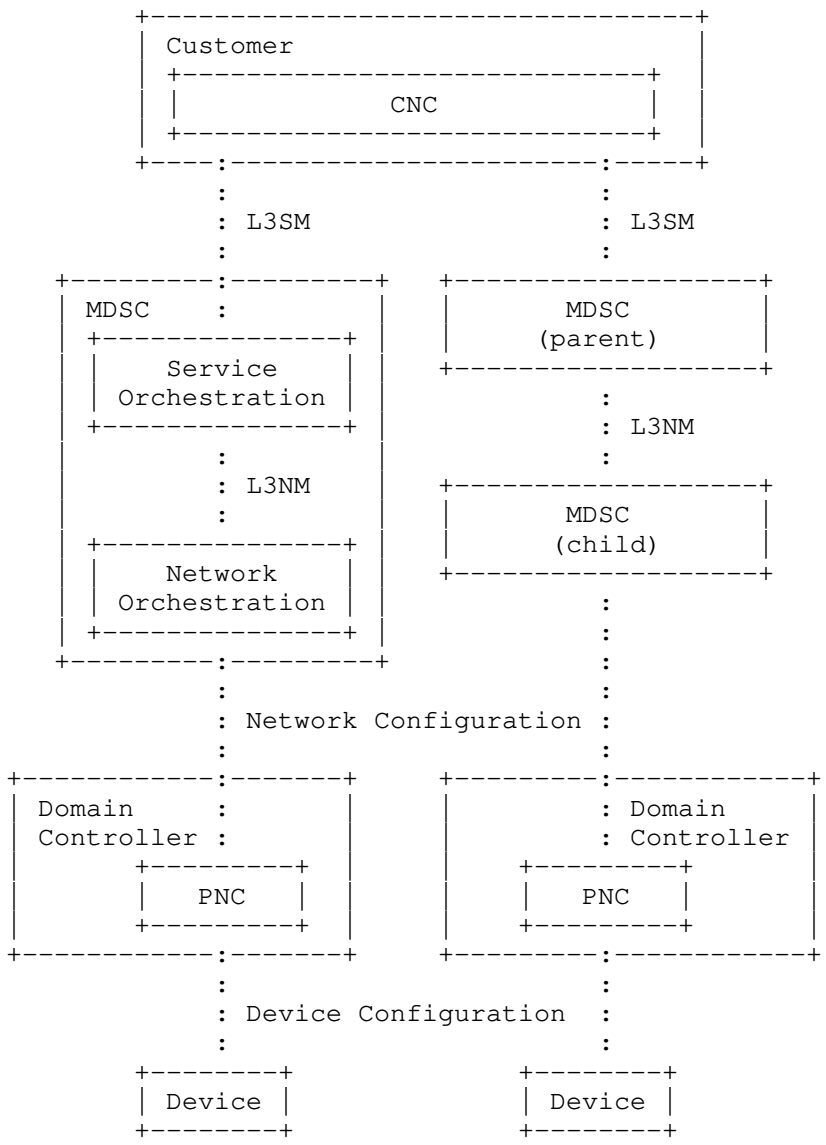


Figure 2: L3SM and L3NM in the Context of ACTN

5. Relation with other YANG Models

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 grouping.

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

L3SM: L3NM is not a Customer Service Model.

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

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

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

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

Device Modules: L3NM is not a device model.

Once a global VPN service is captured by means of L3NM, the actual activation and provisioning of the VPN service will involve a

variety of device modules to tweak the required functions for the delivery of the service. These functions are supported by the VPN nodes and can be managed using device YANG modules. A non-comprehensive list of such device YANG modules is provided below:

* Routing management [RFC8349].

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

* PIM [I-D.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

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 tasks of provisioning and maintenance of these VPNs. Templates and batch processes can be built, and as a result many parameters are needed for the creation from scratch of a VPN that can be abstracted to the upper SDN layer and little manual intervention will be still required.

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

6.2. Multi-Domain Resource Management

The implementation of L3VPN services which span across administratively separated domains (i.e., that are under the administration of different management systems or controllers) requires some network resources to be synchronized between systems. Particularly, there are two resources that must be orchestrated and manage to avoid asymmetric (non-functional) configuration, or the usage of unavailable resources.

For example, RTs shall be synchronized between PEs. When every PE is controlled by the same management system, RT allocation can be

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

6.3. Management of Multicast Services

Multicast services over L3VPN can be implemented either using dual PIM MVPNs (also known as, Draft Rosen model) [RFC4364] or multiprotocol BGP (MBGP)-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 backbone. 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 the PE routers using the same architecture that is used for unicast VPNs.

On the other hand, Draft Rosen has limitations such as reduced options for transport, control plane scalability, availability, operational inconsistency, and the need of maintaining state in the backbone. Because of this, MBGP MVPN is the architectural model that has been taken as the base for implementing multicast service on L3VPN. In this scenario, BGP auto discovery is used to discover MVPN PE members and the customer PIM signaling is sent across provider core through MP-BGP. The multicast traffic is transported on MPLS P2MP LSPs. All of the previous information is carried in the MCAST-VPN BGP NRLI.

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.

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

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).

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

Figure 3: Overall L3NM Tree Structure

7.2. VPN Profiles

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

This document does not make any assumption about the exact definition of these profiles. How such profiles are defined is deployment specific. The model only includes an identifier to these profiles to ease identifying local policies when building a VPN service. As shown in Figure 4, the following identifiers can be included:

- 'cloud-identifier': This identifier refers to a cloud service.
- 'encryption-profile-identifier': An encryption profile refers to a set of policies related to the encryption scheme(s) and setup that can be applied when building and offering a VPN service.
- 'qos-profile-identifier': A QoS profile refers to as set of policies such as classification, marking, and actions (e.g., [RFC3644]).
- 'bfd-profile-identifier': A Bidirectional Forwarding Detection (BFD) profile refers to a set of BFD [RFC5880] policies that can be invoked when building a VPN service.
- 'forwarding-profile-identifier': A forwarding profile refers to the policies that apply to the forwarding of packets conveyed within a VPN. Such policies may consist at applying Access Control Lists (ACLs).

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

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

Figure 4: VPN Profiles Subtree Structure

7.3. Modeling a Layer 3 VPN Service

The 'vpn-service' is the data structure that abstracts a VPN service in the service provider network. Each 'vpn-service' is uniquely identified by an identifier: 'vpn-id'. Such 'vpn-id' is only meaningful locally within the Network controller.

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

The main 'vpn-service' parameters are:

- 'status': Allows the control of the operative and administrative status of the service as a whole.

- 'vpn-id': Is an identifier that is used to uniquely identify the L3VPN Service within L3NM scope.

- 'l3sm-vpn-id': Refers to an identifier of L3SM service. This identifier allows to easily correlate the (network) service as built in the network with a service order.

- 'vpn-service-topology': Indicates the network topology for the service: Hub-Spoke, Any-to-Any, and Custom. The deployment on the

network is defined by the correct usage of import and export profiles

- 'vpn-type': Indicate the VPN service signaling type.
- 'ie-profiles': Defines reusable import/export policies for the same 'vpn-service'. More details are provided in Section 7.3.2.
- 'underlay-transport': Describes the preference for the transport technology to carry the traffic of the VPN service (Section 7.3.3).

The 'vpn-node' is an abstraction that represents a set of policies applied to a network node and that belong to a single 'vpn-service'. A VPN 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.

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

Figure 5: VPN Services Subtree Structure

7.3.1. Service Status

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

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

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           +--rw status
           |  +--rw admin-status
           |  |  +--rw status?         identityref
           |  |  +--rw last-updated?   yang:date-and-time
           |  +--ro oper-status
           |     +--ro status?         identityref
           |     +--ro last-updated?   yang:date-and-time
           ...
```

Figure 6: VPN Service Status Subtree Structure

7.3.2. Concept of Import/Export Profiles

The import and export profiles construct contains a list with information related with route target and distinguishers (RTs and RDs), grouped and identified by 'ie-profile-id'. The identifier is then referenced in one or multiple 'vpn-nodes' so the controller can identify RTs and RDs to be configured for a given VRF. The subtree is shown in Figure 7.

```
+--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
      +--rw vpn-service* [vpn-id]
         +--rw vpn-id                  vpn-common:vpn-id
         +  ...
         +--rw ie-profiles
         |  +--rw ie-profile* [ie-profile-id]
         |     +--rw ie-profile-id    string
         |     +--rw rd?              union
         |     +--rw vpn-targets
         |        +--rw vpn-target* [id]
         |        |  +--rw id                    int8
         |        |  +--rw route-targets* [route-target]
         |        |  |  +--rw route-target    rt-types:route-target
         |        |  +--rw route-target-type
         |        |           rt-types:route-target-type
         |        +--rw vpn-policies
         |           +--rw import-policy?   string
         |           +--rw export-policy?   string
         +--rw vpn-nodes
            +--rw vpn-node* [ne-id]
               +--rw ne-id                      string
               ...
               +--rw vpn-targets
               |  +--rw vpn-target* [id]
               |  |  +--rw id                    int8
               |  |  +--rw route-targets* [route-target]
               |  |  |  +--rw route-target    rt-types:route-target
               |  |  +--rw route-target-type
               |  |           rt-types:route-target-type
               |  +--rw vpn-policies
               |     +--rw import-policy?   string
               |     +--rw export-policy?   string
               ...
```

Figure 7: Subtree Structure of Import/Export Profiles

7.3.3. Underlay Transport

The model allows to indicate a preference for the underlay transport technology when activating a L3VPN service (Figure 8). This preference is especially useful in networks with multiple domains and NNI types. This version of the YANG module supports these options: BGP, LDP, GRE, SR, SR-TE, and RSVP-TE as underlay transport mechanisms. Other profiles can be defined in the future.

```
+--rw l3vpn-ntw
   +--rw vpn-profiles
   |  ...
   +--rw vpn-services
      +--rw vpn-service* [vpn-id]
         +--rw vpn-id                  vpn-common:vpn-id
         +  ...
         +--rw underlay-transport
         |  +--rw type*   identityref
         +--rw vpn-nodes
            +--rw vpn-node* [ne-id]
            ...
```

Figure 8: Subtree Structure of the Underlying Transport

7.3.4. VPN Node

The 'vpn-node' is an abstraction that represents a set of common policies applied on a given network node (tipcally, a PE) and belong to one L3VPN service. In order to indicate the network nodes where the 'vpn-node' applies, the 'ne-id' must be indicated. The 'vpn-node' includes a parameter to indicate the network node on which it is applied. In the case that the 'ne-id' points to a specific PE, the 'vpn-node' will likely be mapped into a VRF in the node. However, the model also allows to point to an abstract node. In this case, the network controller will decide how to split the 'vpn-node' into VRFs. Some 'vpn-node' parameters are listed below:

- local-autonomous-system: Refers to the autonomous system number that is locally configured in the instance. It can be overwritten for specific purposes in the CE-PE BGP session.

- maximum-routes: Set the maximum number of prefixes allowed in the 'vpn-node' instance. This value is typically set in the service request.

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

- Multicast: Enable multicast traffic inside the VPN. Refer to Section 7.3.4.3.

Under the VPN Node ('vpn-node') container, VPN Network Accesses ('vpn-network-access') can be created. The VPN Network Access represents the point to which sites are connected. Note that, unlike in L3SM, the L3NM does not need to model the customer site, only the

points where the traffic from the site are received (i.e., the PE side of PE-CE connections). Hence, the VPN Network access contains the connectivity information between the provider's network and the customer premises. The VPN profiles ('vpn-profiles') have a set of routing policies than can be applied during the service creation.

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

```
  +--rw l3vpn-ntw
     +--rw vpn-profiles
     |  ...
     +--rw vpn-services
        +--rw vpn-service* [vpn-id]
           +--rw vpn-id                  vpn-common:vpn-id
           ...
           +--rw vpn-nodes
              +--rw vpn-node* [ne-id]
                 +--rw vpn-node-id?               union
                 +--rw local-autonomous-system?   inet:as-number
                 +--rw description?               string
                 +--rw ne-id                      string
                 +--rw router-id?                 inet:ip-address
                 +--rw address-family?
                 |       vpn-common:address-family
                 +--rw node-role?                 identityref
                 +--rw rd?                        union
                 +--rw vpn-targets
                 |  +--rw vpn-target* [id]
                 |  |  +--rw id                   int8
                 |  |  +--rw route-targets* [route-target]
                 |  |  |  +--rw route-target    rt-types:route-target
                 |  |  +--rw route-target-type
                 |  |           rt-types:route-target-type
                 |  +--rw vpn-policies
                 |     +--rw import-policy?   string
                 |     +--rw export-policy?   string
                 +--rw status
                 |  +--rw admin-status
                 |  |  +--rw status?         identityref
                 |  |  +--rw last-updated?   yang:date-and-time
                 |  +--ro oper-status
                 |     +--ro status?         identityref
                 |     +--ro last-updated?   yang:date-and-time
                 +--rw node-ie-profile?           leafref
                 +--rw groups
```

```
|  +--rw group* [group-id]
|     +--rw group-id    string
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     ...
+--rw maximum-routes
|  +--rw address-family* [af]
|     +--rw af
|     |       vpn-common:address-family
|     +--rw maximum-routes?   uint32
+--rw multicast {vpn-common:multicast}?
   ...
```

Figure 9: VPN Node Subtree Structure

7.3.4.1. RT/RD Assignment/auto-assignment

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

Three possible behaviors are needed to fulfil the identified use cases:

- The network controller auto-assigns logical resources (RTs, RDs). This can apply for new services deployment.

- The Network Operator/Service orchestrator assigns explicitly the RTs and RDs. This case will fit with a brownfield scenario where some existing services needs to be updated by the network operators.

- The Network Operator/Service orchestrator explicitly wants NO RT/RD to be assigned. This case will fit in VRF-Lite scenarios, CE testing inside the Network or just for troubleshooting pruposes.

Thus a union between two yang data types are included in order to support this scenarios. So, if the leaf is not created in the Yang the expected behavior is the auto-assigns. If the Leaf is created with a valid rd value it will be explicitly assign in the VPN Node and if the leaf is created with an empty value, the RD value will not be deployed in the VPN node.

7.3.4.2. VPN Network Access

A 'vpn-network-access' represents an entry point to a VPN service (Figure 10). In other words, this container encloses the parameters that describe the access information for the traffic that belongs to a particular L3VPN. As such, every 'vpn-network-access' MUST belong to one and only one 'vpn-node'.

A 'vpn-network-access' includes information such as the connection on which the access is defined (see Section 7.3.4.2.1), the encapsulation of the traffic, policies that are applied on the access, etc.

Each 'vpn-network-access' SHOULD have a 'vpn-network-access-type' to select the type of network interface to be deployed in the devices. The available options are:

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

- Multipoint: This option represents a broadcast connection between end-points. It implies the controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.

- Pseudowire: Represent a connection coming from an L2VPN service. It implies the controller must keep the relationship between the logical tunnels or bridges on the devices with the 'id' of the' vpn-network-access'.

- Loopback: It represents the creation of a logical interface on the devices.

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

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

Figure 10: VPN Network Access Tree Structure

7.3.4.2.1. Connection

The definition of a L3VPN is commonly specified not only at the IP layer, but also requires to identify parameters at the Ethernet layer, such as encapsulation type (e.g., VLAN, QinQ, QinAny, VxLAN, etc.). The 'connection' container represents and groups the set of Layer 2 connectivity from where the traffic of the L3VPN in a particular VPN Network access is coming.

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

```
...
+--rw vpn-services
   +--rw vpn-service* [vpn-id]
      +--rw vpn-id                  vpn-common:vpn-id
      +  ...
       +--rw vpn-node* [ne-id]
          +--rw ne-id                       string
          + ...
          +--rw vpn-network-accesses
          |  +--rw vpn-network-access* [id]
          |     +--rw id
          |     |       vpn-common:vpn-id
          |     ...
          |     +--rw connection
          |     |  +--rw encapsulation-type?   identityref
          |     |  +--rw logical-interface
          |     |  |  +--rw peer-reference?   uint32
          |     |  +--rw tagged-interface
          |     |  |  +--rw type?                identityref
          |     |  |  +--rw dot1q-vlan-tagged
          |     |  |  |       {vpn-common:dot1q}?
          |     |  |  |  +--rw tag-type?   identityref
          |     |  |  |  +--rw cvlan-id?   uint16
          |     |  |  +--rw priority-tagged
          |     |  |  |  +--rw tag-type?   identityref
          |     |  |  +--rw qinq {vpn-common:qinq}?
          |     |  |  |  +--rw tag-type?   identityref
          |     |  |  |  +--rw svlan-id    uint16
          |     |  |  |  +--rw cvlan-id    uint16
          |     |  |  +--rw qinany {vpn-common:qinany}?
          |     |  |  |  +--rw tag-type?   identityref
          |     |  |  |  +--rw svlan-id    uint16
          |     |  |  +--rw vxlan {vpn-common:vxlan}?
          |     |  |     +--rw vni-id       uint32
          |     |  |     +--rw peer-mode?   identityref
          |     |  |     +--rw peer-list* [peer-ip]
          |     |  |        +--rw peer-ip    inet:ip-address
          |     |  +--rw bearer
          |     |     ...
          |     ...
```

Figure 11: Encapsulation Subtree Structure

Additionally, the 'bearer-reference' and the pseudowire termination are supported (see Figure 12). A site, as per [RFC4176] represents a VPN customer's location that is connected to the Service Provider network via a CE-PE link, which can access at least one VPN. The connection from the site to the Service Provider network is the

bearer. Every site is associated with a list of bearers. A bearer is the layer two connections with the site. In the module it is assumed that the bearer has been allocated by the Service Provider at the service orchestration step. The bearer is associated to a network element and a port. Hence, a bearer is just a bearer-reference to allow the translation between L3SM and L3NM.

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw vpn-network-access-type?   identityref
|     +--rw connection
|     |  ...
|     |  +--rw bearer
|     |     +--rw bearer-reference?   string
|     |     |       {vpn-common:bearer-reference}?
|     |     +--rw pseudowire
|     |     |  +--rw vcid?      uint32
|     |     |  +--rw far-end?   union
|     |     +--rw vpls
|     |        +--rw vcid?      union
|     |        +--rw far-end?   union
|  ...
```

Figure 12: Bearer Subtree Structure

7.3.4.2.2. IP Connections

IP connection container (Figure 13) has the parameters of the 'vpn-network-access' addressing information. The address allocated in this container would represent the PE interface address configuration. The IP connection container is designed to support both IPv4 and IPv6. It also supports three IP address assignment modes: SLAAC [RFC7527], Provider DHCP, DHCP relay, and static addressing. Only one of them is enabled for a given service.

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw vpn-network-access-type?   identityref
|     +--rw connection
|     |  ...
```

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

```
|      |   |      |           +--:(number)
|      |   |      |           |  +--rw number-of-dynamic-address?
|      |   |      |           |          uint16
|      |   |      |           +--:(explicit)
|      |   |      |              +--rw customer-addresses
|      |   |      |                 +--rw address-group*
|      |   |      |                          [group-id]
|      |   |      |                    +--rw group-id
|      |   |      |                    |       string
|      |   |      |                    +--rw start-address?
|      |   |      |                    |       inet:ipv6-address
|      |   |      |                    +--rw end-address?
|      |   |      |                            inet:ipv6-address
|      |   |      |
|      |   |      +--:(dhcp-relay)
|      |   |      |  +--rw dr-provider-address?
|      |   |      |  |       inet:ipv6-address
|      |   |      |  +--rw dr-prefix-length?
|      |   |      |  |       uint8
|      |   |      |  +--rw customer-dhcp-servers
|      |   |      |     +--rw server-ip-address*
|      |   |      |             inet:ipv6-address
|      |   |      +--:(static-addresses)
|      |   |         ...
```

Figure 13: IP Connection Subtree Structure

In the case of the static addressing (Figure 14), 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 vpn-network-accesses
   |  +--rw vpn-network-access* [id]
   |     +--rw id
   |     |       vpn-common:vpn-id
   |     ...
   |     +--rw vpn-network-access-type?   identityref
   |     +--rw connection
   |     |  ...
   |     +--rw ip-connection
   |     |  +--rw ipv4 {vpn-common:ipv4}?
   |     |  |  +--rw address-allocation-type?
   |     |  |  |       identityref
   |     |  |  +--rw (allocation-type)?
   |     |  |     ...
   |     |  |     +--:(static-addresses)
   |     |  |        +--rw primary-address?
   |     |  |        |       -> ../address/address-id
   |     |  |        +--rw address* [address-id]
   |     |  |           +--rw address-id
   |     |  |           |       string
   |     |  |           +--rw s-provider-address?
   |     |  |           |       inet:ipv4-address
   |     |  |           +--rw s-customer-address?
   |     |  |           |       inet:ipv4-address
   |     |  |           +--rw s-prefix-length?
   |     |  |                   uint8
   |     |  +--rw ipv6 {vpn-common:ipv6}?
   |     |  |  +--rw address-allocation-type?
   |     |  |  |       identityref
   |     |  |  +--rw (allocation-type)?
   |     |  |     ...
   |     |  |     +--:(static-addresses)
   |     |  |        +--rw s-primary-address?
   |     |  |        |       -> ../s-address/address-id
   |     |  |        +--rw s-address* [address-id]
   |     |  |           +--rw address-id
   |     |  |           |       string
   |     |  |           +--rw provider-address?
   |     |  |           |       inet:ipv6-address
   |     |  |           +--rw customer-address?
   |     |  |           |       inet:ipv6-address
   |     |  |           +--rw prefix-length?      uint8
   |     ...
```

Figure 14: IP Connection Subtree Structure: Static Mode

7.3.4.2.3. Security

The 'security' container specifies the authentication and the encryption to be applied for a given VPN network access (Figure 15).

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     +  ...
|     +--rw connection
|     |  ...
|     +--rw ip-connection
|     |  ...
|     +--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 algorithm?       string
|     |     +--rw (key-type)?
|     |        +--:(psk)
|     |           +--rw preshared-key?   string
|     +--rw routing-protocols
|     |  ...
|     +--rw service
|        ...
|  ...
```

Figure 15: Security Subtree Structure

7.3.4.2.4. CE-PE Routing Protocols

The model allows the Provider to configure one or more routing protocols associated with a particular 'vpn-network-access' (Figure 16). This protocol will run between the PE and the CE. A routing protocol instance MUST have a type (e.g., bgp, ospf) and an identifier. The identifier is necessary when multiple instances of the same protocol have to be configured.

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw ip-connection
|     |  ...
|     +--rw routing-protocols
|     |  +--rw routing-protocol* [id]
|     |     +--rw id                  string
|     |     +--rw type?               identityref
|     |     +--rw routing-profiles* [id]
|     |     |  +--rw id      leafref
|     |     |  +--rw type?   identityref
|     |     +--rw ospf {vpn-common:rtg-ospf}?
|     |     |  ...
|     |     +--rw bgp {vpn-common:rtg-bgp}?
|     |     |  ...
|     |     +--rw isis {vpn-common:rtg-isis}?
|     |     |  ...
|     |     +--rw static
|     |     |  ...
|     |     +--rw rip {vpn-common:rtg-rip}?
|     |     |  +--rw address-family*
|     |     |           vpn-common:address-family
|     |     +--rw vrrp {vpn-common:rtg-vrrp}?
|     |        +--rw address-family*
|     |                 vpn-common:address-family
|     +--rw service
|        ...
...
```

Figure 16: Routing Subtree Structure

Routing configuration does not include low-level policies. These policies are low level device configurations that must not be part of an abstracted model. A provider's internal policies (such as security filters) will be implemented as part of the device configuration but does not require any input from this model. Some policies like primary/backup or load-balancing can be inferred from 'access-priority'.

When configuring multiple instances of the same routing protocol, this does not automatically imply that, from a device configuration perspective, there will be parallel instances (multiple processes) running. It will be up to the implementation to use the most appropriate deployment model. As an example, when multiple BGP peers

need to be implemented, multiple instances of BGP must be configured as part of this model. However, from a device configuration point of view, this could be implemented as:

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

To be aligned with [RFC8299], this model supports the following CE-PE routing protocols:

- OSPF: The model (Figure 17) allows the user to configure OSPF to run as routing protocol on the 'vpn-network-access interface'. An OSPF instance can be bound to IPv4, IPv6 or both. When only IPv4 address-family is requested, it will be up to the implementation to drive whether OSPFv2 or OSPFv3 is used.

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw ip-connection
|     |  ...
|     +--rw routing-protocols
|     |  +--rw routing-protocol* [id]
|     |     +--rw id                    string
|     |     +--rw type?                 identityref
|     |     +--rw routing-profiles* [id]
|     |     |  +--rw id      leafref
|     |     |  +--rw type?   identityref
|     |     +--rw ospf {vpn-common:rtg-ospf}?
|     |     |  +--rw address-family*
|     |     |  |       vpn-common:address-family
|     |     |  +--rw area-address
|     |     |  |       yang:dotted-quad
|     |     |  +--rw metric?           uint16
|     |     |  +--rw mtu?              uint16
|     |     |  +--rw process-id?       uint16
|     |     |  +--rw security
|     |     |  |  +--rw auth-key?   string
|     |     |  +--rw sham-links
|     |     |       {vpn-common:rtg-ospf-sham-link}?
|     |     |     +--rw sham-link* [target-site]
|     |     |        +--rw target-site
|     |     |        |       vpn-common:vpn-id
|     |     |        +--rw metric?        uint16
|     |     +--rw bgp {vpn-common:rtg-bgp}?
|     |     |  ...
|     |     +--rw isis {vpn-common:rtg-isis}?
|     |     |  ...
|     |     +--rw static
|     |     |  ...
|     |     +--rw rip {vpn-common:rtg-rip}?
|     |     |  ...
|     |     +--rw vrrp {vpn-common:rtg-vrrp}?
|     |        ...
|     +--rw service
|        ...
...
```

Figure 17: OPSF Routing Subtree Structure

- BGP: The model (Figure 18) allows to configure a BGP neighbor, including a set for parameters that are pertinent to be tweaked at the network level for service customization purposes. This container does not aim to include every BGP parameter; a comprehensive set of parameters belongs more to the BGP device model. The following parameters are captured in Figure 18. It is up to the implementation to drive the corresponding BGP device configuration.

  * 'peer-autonomous-system': This parameter conveys the Customer's AS Number (ASN).

  * 'local-autonomous-system': This parameter is set of AS override is activated for this peer.

  * 'address-family': This attribute indicates the address-family of the peer. It can be set to IPv4, IPv6, or both address-families.

  * 'neighbor': The module supports supplying two neighbors (each for a given address-family) or one neighbor (if 'address-family' attribute is set to both IPv4 and IPv6 address-families). A list of IP address(es) of the BGP neighbor can be then conveyed in this parameter.

  * 'multihop': This attribute indicates the number of allowed IP hops between a BGP peer and a PE.

  * 'security': The authentication type will be driven by the implementation but the module supports any authentication that uses a key as a parameter.

  * 'as-override': If set, this parameter indicates whether AS override is enabled, i.e., replace the ASN of the peer specified in the AS Path attribute with the ASN identified by the 'local-autonomous-system' attribute.

  * 'default-route': This attribute controls whether default route(s) can be advertised to the peer.

  * 'bgp-max-prefix': This attribute is used to control how many prefixes can be received from a neighbor. If reached, the BGP session will be teared down.

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

the KeepAlive timer between a PE and a BGP peer (Section 4.4 of [RFC4271]).

```
                ...
                +--rw vpn-network-accesses
                |  +--rw vpn-network-access* [id]
                |     +--rw id
                |     |       vpn-common:vpn-id
                |     ...
                |     +--rw ip-connection
                |     |  ...
                |     +--rw routing-protocols
                |     |  +--rw routing-protocol* [id]
                |     |     +--rw id                    string
                |     |     +--rw type?                 identityref
                |     |     +--rw routing-profiles* [id]
                |     |     |  +--rw id      leafref
                |     |     |  +--rw type?   identityref
                |     |     +--rw ospf {vpn-common:rtg-ospf}?
                |     |     |  ...
                |     |     +--rw bgp {vpn-common:rtg-bgp}?
                |     |     |  +--rw peer-autonomous-system
                |     |     |  |           inet:as-number
                |     |     |  +--rw local-autonomous-system?
                |     |     |  |       inet:as-number
                |     |     |  +--rw address-family*
                |     |     |  |      vpn-common:address-family
                |     |     |  +--rw neighbor*
                |     |     |  |       inet:ip-address
                |     |     |  +--rw multihop?            uint8
                |     |     |  +--rw security
                |     |     |  |  +--rw auth-key?   string
                |     |     |  +--rw status
                |     |     |  |  +--rw admin-status
                |     |     |  |  |  +--rw status?       identityref
                |     |     |  |  |  +--rw last-updated?
                |     |     |  |  |  |            yang:date-and-time
                |     |     |  |  +--ro oper-status
                |     |     |  |     +--ro status?       identityref
                |     |     |  |     +--ro last-updated?
                |     |     |  |                   yang:date-and-time
                |     |     |  +--rw description?             string
                |     |     |  +--rw as-override?            boolean
                |     |     |  +--rw default-route?          boolean
                |     |     |  +--rw bgp-max-prefix
                |     |     |  |  +--rw max-prefix?           uint32
                |     |     |  |  +--rw warning-threshold? decimal64
                |     |     |  |  +--rw violate-action?  enumeration
```

```
|      |      |  |  +--rw restart-interval?     uint16
|      |      |  +--rw bgp-timer
|      |      |     +--rw keep-alive?   uint16
|      |      |     +--rw hold-time?    uint16
|      |      +--rw isis {vpn-common:rtg-isis}?
|      |      |  ...
|      |      +--rw static
|      |      |  ...
|      |      +--rw rip {vpn-common:rtg-rip}?
|      |      |  ...
|      |      +--rw vrrp {vpn-common:rtg-vrrp}?
|      |         ...
|      +--rw service
|         ...
...
```

Figure 18: BGP Routing Subtree Structure

- IS-IS: The model (Figure 19) allows the user to configure IS-IS to run on the 'vpn-network-access' interface. An IS-IS instance can run L1, L2, or both levels.

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw ip-connection
|     |  ...
|     +--rw routing-protocols
|     |  +--rw routing-protocol* [id]
|     |     +--rw id                  string
|     |     +--rw type?
|     |     |        identityref
|     |     +--rw routing-profiles* [id]
|     |     |  +--rw id      leafref
|     |     |  +--rw type?   identityref
|     |     +--rw ospf {vpn-common:rtg-ospf}?
|     |     |  ...
|     |     +--rw bgp {vpn-common:rtg-bgp}?
|     |     |  ...
|     |     +--rw isis {vpn-common:rtg-isis}?
|     |     |  +--rw address-family*
|     |     |  |       vpn-common:address-family
|     |     |  +--rw area-address
|     |     |  |       yang:dotted-quad
|     |     |  +--rw level?            identityref
```

```
|      |      |  +--rw metric?           uint16
|      |      |  +--rw process-id?       uint16
|      |      |  +--rw mode?             enumeration
|      |      |  +--rw status
|      |      |     +--rw admin-status
|      |      |     |  +--rw status?
|      |      |     |  |       identityref
|      |      |     |  +--rw last-updated?
|      |      |     |          yang:date-and-time
|      |      |     +--ro oper-status
|      |      |        +--ro status?
|      |      |        |       identityref
|      |      |        +--ro last-updated?
|      |      |                yang:date-and-time
|      |      +--rw static
|      |      |  ...
|      |      +--rw rip {vpn-common:rtg-rip}?
|      |      |  ...
|      |      +--rw vrrp {vpn-common:rtg-vrrp}?
|      |         ...
|      +--rw service
|         ...
...
```

Figure 19: IS-IS Routing Subtree Structure

- RIP: The module covers only a list of address-family as parameter.
- VRRP: The module covers only a list of address-family as parameter.

The module allows a user to configure one or more IPv4 and/or IPv6 static routes as depicted in Figure 20.

```
                ...
                +--rw vpn-network-accesses
                |  +--rw vpn-network-access* [id]
                |     +--rw id
                |     |       vpn-common:vpn-id
                |     ...
                |     +--rw ip-connection
                |     |  ...
                |     +--rw routing-protocols
                |     |  +--rw routing-protocol* [id]
                |     |     +--rw id              string
                |     |     +--rw type?      identityref
                |     |     +--rw routing-profiles* [id]
                |     |     |  +--rw id      leafref
                |     |     |  +--rw type?   identityref
                |     |     +--rw ospf {vpn-common:rtg-ospf}?
                |     |     |  ...
                |     |     +--rw bgp {vpn-common:rtg-bgp}?
                |     |     |  ...
                |     |     +--rw isis {vpn-common:rtg-isis}?
                |     |     |  ...
                |     |     +--rw static
                |     |     |  +--rw cascaded-lan-prefixes
                |     |     |     +--rw ipv4-lan-prefixes*
                |     |     |     |       [lan next-hop]
                |     |     |     |       {vpn-common:ipv4}?
                |     |     |     |  +--rw lan
                |     |     |     |  |       inet:ipv4-prefix
                |     |     |     |  +--rw lan-tag?    string
                |     |     |     |  +--rw next-hop
                |     |     |     |          inet:ipv4-address
                |     |     |     +--rw ipv6-lan-prefixes*
                |     |     |             [lan next-hop]
                |     |     |             {vpn-common:ipv6}?
                |     |     |        +--rw lan
                |     |     |        |       inet:ipv6-prefix
                |     |     |        +--rw lan-tag?    string
                |     |     |        +--rw next-hop
                |     |     |                inet:ipv6-address
                |     |     +--rw rip {vpn-common:rtg-rip}?
                |     |     |  ...
                |     |     +--rw vrrp {vpn-common:rtg-vrrp}?
                |     |        ...
                |     +--rw service
                |        ...
                ...
```

Figure 20: Static Routing Subtree Structure

7.3.4.2.5. Services

The 'services' container specifies the service parameters to apply for a given VPN network access (Figure 21).

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     |       vpn-common:vpn-id
|     ...
|     +--rw service
|        +--rw svc-input-bandwidth     uint64
|        +--rw svc-output-bandwidth    uint64
|        +--rw svc-mtu                 uint16
|        +--rw qos {vpn-common:qos}?
|        |  +--rw qos-classification-policy
|        |  |  +--rw rule* [id]
|        |  |     +--rw id
|        |  |     |       string
|        |  |     +--rw (match-type)?
|        |  |     |  +--:(match-flow)
|        |  |     |  |  +--rw (l3)?
|        |  |     |  |  |  +--:(ipv4)
|        |  |     |  |  |  |  ...
|        |  |     |  |  |  +--:(ipv6)
|        |  |     |  |  |     ...
|        |  |     |  |  +--rw (l4)?
|        |  |     |  |     +--:(tcp)
|        |  |     |  |     |  ...
|        |  |     |  |     +--:(udp)
|        |  |     |  |        ...
|        |  |     |  +--:(match-application)
|        |  |     |     +--rw match-application?
|        |  |     |             identityref
|        |  |     +--rw target-class-id?
|        |  |             string
|        |  +--rw qos-profile
|        |     +--rw qos-profile* [profile]
|        |        +--rw profile      leafref
|        |        +--rw direction?   identityref
|        +--rw carrierscarrier
|        |       {vpn-common:carrierscarrier}?
|        |  +--rw signalling-type?   enumeration
|        +--rw multicast {vpn-common:multicast}?
|           +--rw site-type?        enumeration
|           +--rw address-family?
|           |       vpn-common:address-family
|           +--rw protocol-type?    enumeration
|           +--rw remote-source?    boolean
...
```

Figure 21: Services Subtree Structure

The following attributes are defined:

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

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

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

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

- 'multicast': Specifies the multicast mode and other service-related attributes such as the address-family.

- 'qos': Is used to define QoS policies to apply on a given connection. Classification can be based on many criteria such as:

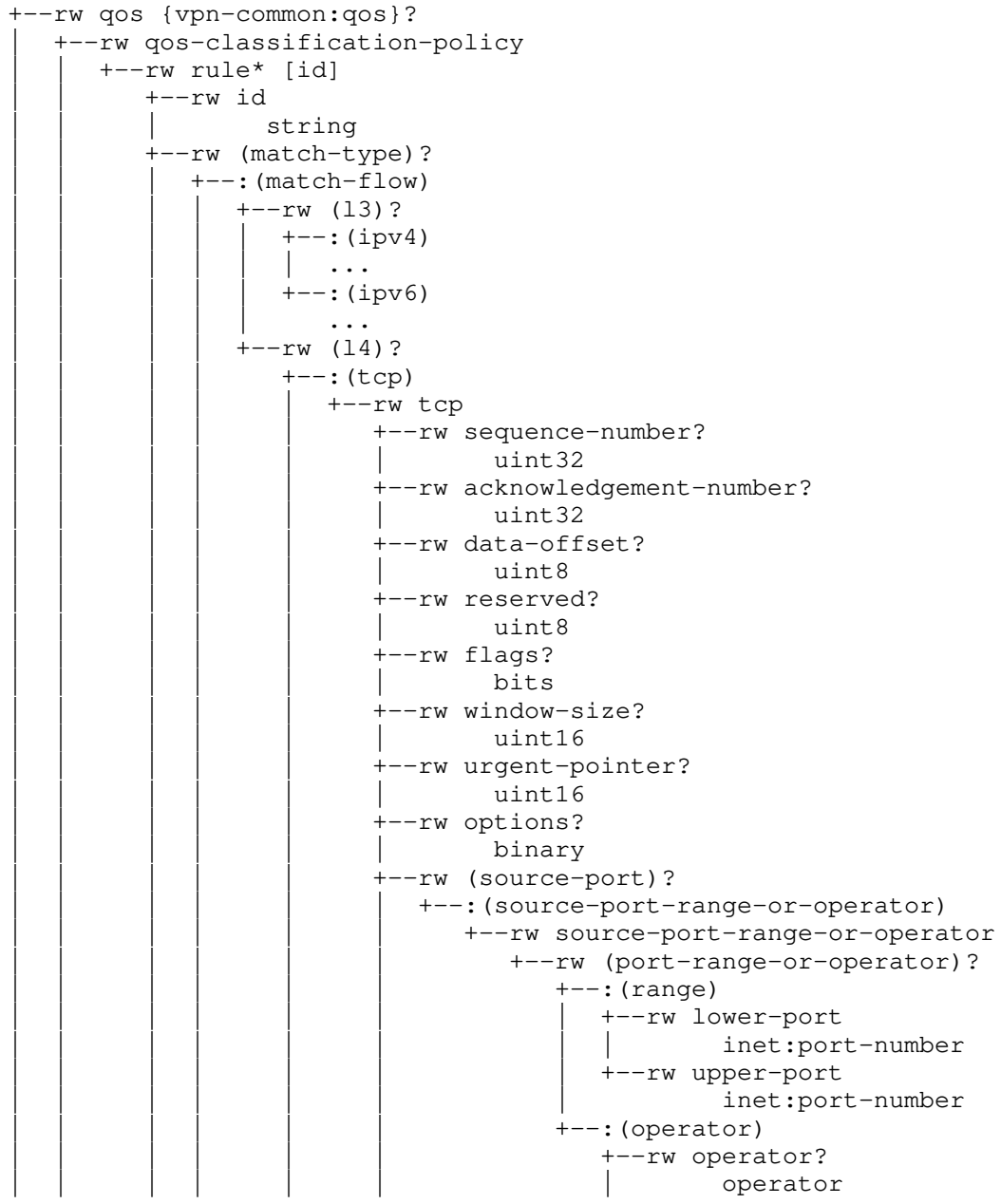
  * Layer 3: As shown in Figure 23, the model allow to classify based on any IP header field or a combination thereof. Both IPv4 and IPv6 are supported.

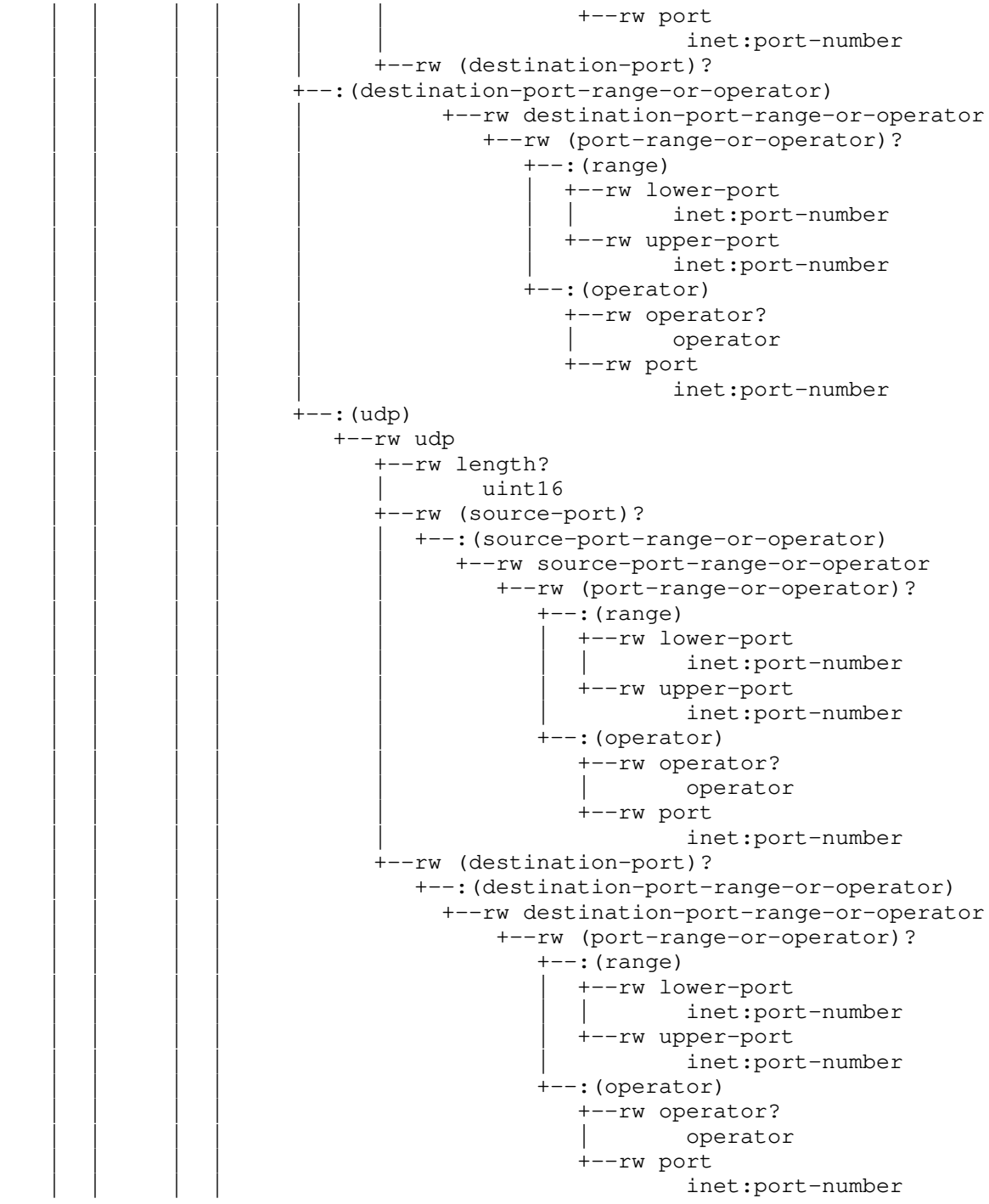
```
 +--rw qos {vpn-common:qos}?
 |  +--rw qos-classification-policy
 |  |  +--rw rule* [id]
 |  |     +--rw id
 |  |     |       string
 |  |     +--rw (match-type)?
 |  |     |  +--:(match-flow)
 |  |     |  |  +--rw (l3)?
 |  |     |  |  |  +--:(ipv4)
 |  |     |  |  |  |  ...
 |  |     |  |  |  +--:(ipv6)
 |  |     |  |  |     ...
 |  |     |  |  +--rw (l4)?
 |  |     |  |  +--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
        |  |      |  +--rw (l4)?
        |  |      |     +--:(tcp)
        |  |      |     |  ...
        |  |      |     +--:(udp)
        |  |      |        ...
        |  |      |  |
       ...
```

Figure 22: QoS Subtree Structure (L3)

* Layer 4: As shown in Figure 23, TCP or UDP-related match crietria can be specified.




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

```
...
```

Figure 23: QoS Subtree Structure (L4)

* Application match

7.3.4.3. Multicast

Multicast MAY be enabled for a particular vpn-network-node (see Figure 24).

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

When ASM is used, the model supports the configuration of rendez-vous points (RPs). RP discovery may be 'static', 'bsr-rp', or 'auto-rp'. When set to 'static', RP to multicast grouping mapping MUST be configured as part of the 'rp-group-mappings' container. The RP MAY be a provider node or a customer node. When the RP is a customer node, the RP address must be configured using the 'rp-address' leaf otherwise no RP address is needed.

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

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

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

```
...
+--rw vpn-network-accesses
|  +--rw vpn-network-access* [id]
|     +--rw id
|     ..
+--rw multicast {vpn-common:multicast}?
   +--rw enabled?       boolean
   +--rw tree-flavor*   identityref
```

```
                    +--rw rp
                    |  +--rw rp-group-mappings
                    |  |  +--rw rp-group-mapping* [id]
                    |  |     +--rw id                    uint16
                    |  |     +--rw provider-managed
                    |  |     |  +--rw enabled?
                    |  |     |  |       boolean
                    |  |     |  +--rw rp-redundancy?
                    |  |     |  |       boolean
                    |  |     |  +--rw optimal-traffic-delivery?
                    |  |     |  |       boolean
                    |  |     |  +--rw anycast
                    |  |     |     +--rw local-address?
                    |  |     |     |       inet:ip-address
                    |  |     |     +--rw rp-set-address*
                    |  |     |             inet:ip-address
                    |  |     +--rw rp-address
                    |  |     |       inet:ip-address
                    |  |     +--rw groups
                    |  |        +--rw group* [id]
                    |  |           +--rw id
                    |  |           |       uint16
                    |  |           +--rw (group-format)
                    |  |              +--:(group-prefix)
                    |  |              |  +--rw group-address?
                    |  |              |          inet:ip-prefix
                    |  |              +--:(startend)
                    |  |                 +--rw group-start?
                    |  |                 |       inet:ip-address
                    |  |                 +--rw group-end?
                    |  |                         inet:ip-address
                    |  +--rw rp-discovery
                    |     +--rw rp-discovery-type?   identityref
                    |     +--rw bsr-candidates
                    |        +--rw bsr-candidate-address*
                    |                inet:ip-address
                    +--rw msdp {msdp}?
                       +--rw enabled?         boolean
                       +--rw peer?            inet:ip-address
                       +--rw local-address?   inet:ip-address
```

Figure 24: Multicast Subtree Structure

8. Layer 3 Network Model

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

```
<CODE BEGINS>  file "ietf-l3vpn-ntw@2020-10-16.yang"
module ietf-l3vpn-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-l3vpn-ntw";
  prefix l3nm;

  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC UUUU: A Layer 2/3 VPN Common YANG Model";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-packet-fields {
    prefix pf;
    reference
      "RFC 8519: YANG Data Model for Network Access
                 Control Lists (ACLs)";
  }

  organization
    "IETF OPSA (Operations and Management Area) Working Group ";
  contact
    "WG Web:   <http://tools.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Editor:   Samier Barguil
               <mailto:samier.barguilgiraldo.ext@telefonica.com>
     Editor:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Editor:   Mohamed Boucadair
              <mailto:mohamed.boucadair@orange.com>
     Author:   Luis Angel Munoz
               <mailto:luis-angel.munoz@vodafone.com>
     Author:   Alejandro Aguado
               <mailto:alejandro.aguado_martin@nokia.com>
    ";
  description
    "This YANG module defines a generic network-oriented model
     for the configuration of Layer 3 Virtual Private Networks.
```

```
    Copyright (c) 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 2020-10-16 {
    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)";
  }

  /* Typedefs */

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

  /* Identities */

  identity address-allocation-type {
    description
      "Base identity for address-allocation-type for
       PE-CE link.";
```

```
  }

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

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

  identity provider-dhcp-slaac {
    base address-allocation-type;
    description
      "The Provider's network provides a DHCP service to
       the customer, as well as IPv6 Stateless Address
       Autoconfiguration (SLAAC).";
    reference
      "RFC 7527: IPv6 Stateless Address Autoconfiguration";
  }

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

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

  identity isis-level {
    description
      "Defines the IS-IS level for interface
       and system.";
  }

  identity level1 {
    base isis-level;
```

```
    description
      "IS-IS level 1.";
  }

  identity level2 {
    base isis-level;
    description
      "IS-IS level 2.";
  }

  identity level1-2 {
    base isis-level;
    description
      "IS-IS levels 1 and 2.";
  }

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

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

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

  /* Groupings */

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

```
      "Grouping to define a set of security parameters";
  }

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

  /* Main Blocks */
  /* Main l3nm */

  container l3vpn-ntw {
    container vpn-profiles {
      uses vpn-common:vpn-profile-cfg;
      description
        "Contains a set of valid VPN Profiles to
         reference in the VPN service.";
    }
    container vpn-services {
      list vpn-service {
        key "vpn-id";
        uses vpn-common:service-status;
        uses vpn-common:vpn-description;
        leaf l3sm-vpn-id {
          type vpn-common:vpn-id;
          description
```

```
            "Pointer to the parent L3SM service,
             if any.";
        }
        leaf vpn-type {
          type identityref {
            base vpn-common:vpn-signaling-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.";
        }
        container ie-profiles {
          list ie-profile {
            key "ie-profile-id";
            leaf ie-profile-id {
              type string;
              description
                "IE profile id.";
            }
            uses vpn-common:rt-rd;
            description
              "List for Imort/Export profile.";
          }
          description
            "Container for Import/Export profiles.";
        }
        uses vpn-common:svc-transport-encapsulation;
        container vpn-nodes {
          description
            "Container for VPN nodes.";
          list vpn-node {
            key "vpn-node-id";
            leaf vpn-node-id {
              type union {
                type vpn-common:vpn-id;
                type uint32;
              }
              description
                "Type STRING or NUMBER Service-Id.";
            }
            leaf local-autonomous-system {
```

```
            type inet:as-number;
            description
              "Provider's AS number in case the customer
               requests BGP routing.";
          }
          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 router-id {
            type inet:ip-address;
            description
              "The router-id information can be an IPv4
               or IPv6 address.";
          }
          leaf address-family {
            type vpn-common:address-family;
            description
              "The address family used for router-id
               information.";
          }
          leaf node-role {
            type identityref {
              base vpn-common:role;
            }
            default "vpn-common:any-to-any-role";
            description
              "Role of the VPN node in the IP VPN.";
          }
          uses vpn-common:rt-rd;
          uses vpn-common:service-status;
          leaf node-ie-profile {
            type leafref {
              path "/l3vpn-ntw/vpn-services/"
                 + "vpn-service/ie-profiles/"
                 + "ie-profile/ie-profile-id";
            }
            description
              "Node's Import/Export profile.";
          }
          uses vpn-common:vpn-node-group;
```

```
            container vpn-network-accesses {
              list vpn-network-access {
                key "id";
                leaf id {
                  type vpn-common:vpn-id;
                  description
                    "Identifier for the access.";
                }
                leaf port-id {
                  type vpn-common:vpn-id;
                  description
                    "Identifier for the network access.";
                }
                leaf description {
                  type string;
                  description
                    "Textual description of a network access.";
                }
                uses vpn-common:service-status;
                leaf vpn-network-access-type {
                  type identityref {
                    base vpn-common:site-network-access-type;
                  }
                  default "vpn-common:point-to-point";
                  description
                    "Describes the type of connection, e.g.,
                     point-to-point or multipoint.";
                }
                container connection {
                  leaf encapsulation-type {
                    type identityref {
                      base vpn-common:encapsulation-type;
                    }
                    default "vpn-common:untagged-int";
                    description
                      "Encapsulation type.  By default,
                       the encapsulation type is set to
                       'untagged'.";
                  }
                  container logical-interface {
                    leaf peer-reference {
                      type uint32;
                      description
                        "Specify the associated logical peer
                         interface";
                    }
                    description
                      "Reference of a logical interface
```

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

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

```
            description
              "Peer IP.";
          }
          description
            "List of peer IP addresses.";
        }
        description
          "QinQ.";
      }
      description
        "Container for tagged interfaces.";
    }
    container bearer {
      leaf bearer-reference {
        if-feature "vpn-common:bearer-reference";
        type string;
        description
          "This is an internal reference for*
           the SP.";
      }
      container pseudowire {
        leaf vcid {
          type uint32;
          description
            "PW or VC identifier.";
        }
        leaf far-end {
          type union {
            type uint32;
            type inet:ipv4-address;
          }
          description
            "SDP/Far End/LDP neighbour reference.";
        }
        description
          "Pseudowire termination parameters";
      }
      container vpls {
        leaf vcid {
          type union {
            type uint32;
            type string;
          }
          description
            "VCID identifier, IRB/RVPPLs interface
             supported using string
             format.";
        }
```

```
            leaf far-end {
              type union {
                type uint32;
                type inet:ipv4-address;
              }
              description
                "SDP/Far End/LDP Neighbour reference.";
            }
            description
              "Pseudowire termination parameters";
          }
          description
            "Defines physical properties of a site
             attachment.";
        }
        description
          "Encapsulation types";
      }
      container ip-connection {
        container ipv4 {
          if-feature "vpn-common:ipv4";
          leaf address-allocation-type {
            type identityref {
              base address-allocation-type;
            }
            must "not(derived-from-or-self(current(), "
              + "'slaac') or derived-from-or-self(current(),"
              + " 'provider-dhcp-slaac'))" {
              error-message "SLAAC is only applicable to
                             IPv6";
            }
            description
              "Defines how addresses are allocated.
               If there is no value for the address
               allocation type, then IPv4 is not enabled.";
          }
          choice allocation-type {
            case provider-dhcp {
              when "derived-from-or-self(./address-"
                 + "allocation-type, 'provider-dhcp')" {
                description
                  "Only applies when addresses are
                   allocated by DHCP.";
              }
              leaf provider-address {
                type inet:ipv4-address;
                description
                  "Address of provider side.
```

```
        If provider-address is not specified,
        then prefix length should not be
        specified either.

        It also implies provider-dhcp
        allocation is not enabled.

        If provider-address is specified,
        then the prefix length may or
        may not be specified.";
    }
    leaf prefix-length {
      type uint8 {
        range "0..32";
      }
      must '(../provider-address)' {
        error-message
          "If the prefix length is specified,
           provider-address must also be
           specified.";
        description
          "If the prefix length is specified,
           provider-address must also be
           specified.";
      }
      description
        "Subnet prefix length expressed in bits.
         If not specified, or specified as zero,
         this means the customer leaves the actual
         prefix length value to the provider.";
    }
    choice address-assign {
      default "number";
      case number {
        leaf number-of-dynamic-address {
          type uint16;
          default "1";
          description
            "Describes the number of IP
             addresses the customer requires.";
        }
      }
      case explicit {
        container customer-addresses {
          list address-group {
            key "group-id";
            leaf group-id {
              type string;
```

```
                    description
                      "Group-id for the address range
                       from start-address to
                       end-address.";
                  }
                  leaf start-address {
                    type inet:ipv4-address;
                    description
                      "First address.";
                  }
                  leaf end-address {
                    type inet:ipv4-address;
                    description
                      "Last address.";
                  }
                  description
                    "Describes IP addresses allocated by
                     DHCP.

                     When only start-address or only
                     end-address is present, it
                     represents a single address.
                     When both start-address and
                     end-address are specified, it
                     implies a range inclusive of
                     both addresses.  If no address
                     is specified, it implies customer
                     addresses group is not supported.";
                }
                description
                  "Container for customer addresses is
                   allocated by DHCP.";
              }
            }
            description
              "Choice for the way to assign
               addresses.";
          }
          description
            "DHCP allocated addresses related
             parameters.";
        }
        case dhcp-relay {
          when "derived-from-or-self(./address-allocation"
             + "-type, 'provider-dhcp-relay')" {
            description
              "Only applies when provider is required to
               implement DHCP relay function.";
```

```
                    }
                    leaf dr-provider-address {
                      type inet:ipv4-address;
                      description
                        "Address of provider side.

                         If provider-address is
                         not specified, then prefix length
                         should not be specified either.

                         It also implies provider-dhcp
                         allocation is not enabled.

                         If provider-address is specified,
                         then prefix length may or may
                         not be specified.";
                    }
                    leaf dr-prefix-length {
                      type uint8 {
                        range "0..32";
                      }
                      must '(../dr-provider-address)' {
                        error-message
                          "If prefix length is specified,
                           provider-address must also be
                           specified.";
                        description
                          "If prefix length is specified,
                           provider-address must also be
                           specified.";
                      }
                      description
                        "Subnet prefix length expressed in bits.

                         If not specified, or specified as zero,
                         this means the customer leaves the
                         actual prefix  length value
                         to the provider.";
                    }
                    container customer-dhcp-servers {
                      leaf-list server-ip-address {
                        type inet:ipv4-address;
                        description
                          "IP address of customer DHCP
                           server.";
                      }
                      description
                        "Container for list of customer
```

```
            DHCP servers.";
        }
        description
          "DHCP relay provided by operator.";
      }
      case static-addresses {
        when "derived-from-or-self(./address-allocation"
           + "-type, 'static-address')" {
          description
            "Only applies when address allocation
            type is static.";
        }
        leaf primary-address {
          type leafref {
            path "../address/address-id";
          }
          description
            "Principal address of the connection.";
        }
        list address {
          key "address-id";
          leaf address-id {
            type string;
            description
              "IPv4 Address";
          }
          leaf s-provider-address {
            type inet:ipv4-address;
            description
              "IPv4 Address List of the provider side.
               When the protocol allocation type is
               static, the provider address must be
               configured.";
          }
          leaf s-customer-address {
            type inet:ipv4-address;
            description
              "IPv4 Address of customer side.";
          }
          leaf s-prefix-length {
            type uint8 {
              range "0..32";
            }
            description
              "Subnet prefix length expressed
               in bits. It is applied to both
               provider-address and customer-address.";
          }
```

```
            description
              "Describes IPv4 addresses used.";
          }
          description
            "Describes IPv4 addresses used.";
        }
        description
          "Choice the address allocation.";
      }
      description
        "IPv4-specific parameters.";
    }
    container ipv6 {
      if-feature "vpn-common:ipv6";
      leaf address-allocation-type {
        type identityref {
          base address-allocation-type;
        }
        description
          "Defines how addresses are allocated.
           If there is no value for the address
           allocation type, then IPv6 is
           not enabled.";
      }
      choice allocation-type {
        choice provider-dhcp {
          when "derived-from-or-self(./address-allo"
             + "cation-type, 'provider-dhcp') "
             + "or derived-from-or-self(./address-allo"
             + "cation-type, 'provider-dhcp-slaac')" {
            description
              "Only applies when addresses are
               allocated by DHCP.";
          }
          leaf provider-address {
            type inet:ipv6-address;
            description
              "Address of the provider side.

               If provider-address is not specified,
               then prefix length should not be
               specified either.  It also implies
               provider-dhcp allocation is not
               enabled.

               If provider-address is
               specified, then prefix length may
               or may not be specified.";
```

```
                        }
                        leaf prefix-length {
                          type uint8 {
                            range "0..128";
                          }
                          must '(../provider-address)' {
                            error-message
                              "If prefix length is specified,
                               provider-address
                               must also be specified.";
                            description
                              "If prefix length is specified,
                                provider-address
                               must also be specified.";
                          }
                          description
                            "Subnet prefix length expressed in
                             bits.

                             If not specified, or specified as
                             zero, this means the customer leaves
                             the actual prefix length value to
                             the provider.";
                        }
                        choice address-assign {
                          default "number";
                          case number {
                            leaf number-of-dynamic-address {
                              type uint16;
                              default "1";
                              description
                                "Describes the number of IP
                                 addresses required by the
                                 customer.";
                            }
                          }
                          case explicit {
                            container customer-addresses {
                              list address-group {
                                key "group-id";
                                leaf group-id {
                                  type string;
                                  description
                                    "Group-id for the address range
                                     from start-address to
                                     end-address.";
                                }
                                leaf start-address {
```

```
                    type inet:ipv6-address;
                    description
                      "First address.";
                  }
                  leaf end-address {
                    type inet:ipv6-address;
                    description
                      "Last address.";
                  }
                  description
                    "Describes IP addresses allocated
                     by DHCP.

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

                     When both start-address and
                     end-address are specified, it
                     implies a range inclusive of
                     both addresses.

                     If no address is specified, it
                     implies customer addresses group
                     is not supported.";
                }
                description
                  "Container for customer addresses
                   allocated by DHCP.";
              }
            }
            description
              "Choice for the way to assign addresses.";
          }
          description
            "DHCP allocated addresses related
             parameters.";
        }
        case dhcp-relay {
          when "derived-from-or-self(./address-allo"
             + "cation-type, 'provider-dhcp-relay')" {
            description
              "Only applies when the provider is required
               to implement DHCP relay function.";
          }
          leaf dr-provider-address {
            type inet:ipv6-address;
            description
```

```
        "Address of the provider side.

         If provider-address is not specified,
         then prefix length should not be
         specified either.  It also implies
         provider-dhcp allocation is not enabled.

         If provider address is specified, then
         prefix length may or may not be
         specified.";
      }
      leaf dr-prefix-length {
        type uint8 {
          range "0..128";
        }
        must '(../dr-provider-address)' {
          error-message
            "If prefix length is specified,
             provider-address must also be
             specified.";
          description
            "If prefix length is specified,
             provider-address must also be
             specified.";
        }
        description
          "Subnet prefix length expressed in bits.

           If not specified, or specified as zero,
           this means the customer leaves the
           actual prefix length value to the
           provider.";
      }
      container customer-dhcp-servers {
        leaf-list server-ip-address {
          type inet:ipv6-address;
          description
            "This node contains the IP address of
             the customer DHCP server.  If the DHCP
             relay function is implemented by the
             provider, this node contains the
             configured value.";
        }
        description
          "Container for list of customer DHCP
           servers.";
      }
      description
```

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

```
            description
              "IPv6-specific parameters.";
          }
          description
            "IPv6 allocation type.";
        }
        description
          "IPv6-specific parameters.";
      }
      container oam {
        container bfd {
          if-feature "vpn-common:bfd";
          leaf enabled {
            type boolean;
            default "false";
            description
              "If true, BFD activation is required.";
          }
          choice holdtime {
            default "fixed";
            case fixed {
              leaf fixed-value {
                type uint32;
                units "msec";
                description
                  "Expected BFD holdtime.

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

                   If the provider doesn't allow the
                   customer to use this function,
                   the fixed-value will not be set.";
              }
            }
            case profile {
              leaf profile-name {
                type leafref {
                  path "/l3vpn-ntw/vpn-profiles/"
                     + "valid-provider-identifiers/"
                     + "bfd-profile-identifier/id";
                }
                description
                  "Well-known SP profile name.

                   The provider can propose some profiles
```

```
                        to the customer, depending on the
                        service level the customer wants to
                        achieve.

                        Profile names must be communicated to
                        the customer.";
                    }
                    description
                      "Well-known SP profile.";
                  }
                  description
                    "Choice for holdtime flavor.";
                }
                description
                  "Container for BFD.";
              }
              description
                "Defines the Operations, Administration,
                 and Maintenance (OAM)mechanisms used on
                 the connection.

                 BFD is set as a fault detection mechanism,
                 but the 'oam' container can easily
                 be augmented by other mechanisms";
            }
            description
              "Defines connection parameters.";
          }
          container security {
            container encryption {
              if-feature "vpn-common:encryption";
              leaf enabled {
                type boolean;
                default "false";
                description
                  "If true, traffic encryption on the
                   connection is required. It is
                   disabled, otherwise.";
              }
              leaf layer {
                when "../enabled = 'true'" {
                  description
                    "Require a value for layer when
                     enabled is true.";
                }
                type enumeration {
                  enum layer2 {
                    description
```

```
                        "Encryption will occur at Layer 2.";
                      }
                      enum layer3 {
                        description
                          "Encryption will occur at Layer 3.
                           For example, IPsec may be used when
                           a customer requests Layer 3
                           encryption.";
                      }
                    }
                    description
                      "Layer on which encryption is applied.";
                  }
                  description
                    "Container for CE-PE security encryption.";
                }
                container encryption-profile {
                  choice profile {
                    case provider-profile {
                      leaf profile-name {
                        type leafref {
                          path "/l3vpn-ntw/vpn-profiles"
                             + "/valid-provider-identifiers"
                             + "/encryption-profile-identifier/id";
                        }
                        description
                          "Name of the SP profile to be applied.";
                      }
                    }
                    case customer-profile {
                      leaf algorithm {
                        type string;
                        description
                          "Encryption algorithm to be used.";
                      }
                    }
                    description
                      "Choice for encryption profile.";
                  }
                  choice key-type {
                    default "psk";
                    case psk {
                      leaf preshared-key {
                        type string;
                        description
                          "Pre-Shared Key (PSK) coming from the
                           customer.";
                      }
```

```
          }
          description
            "Choice of encryption profile.
             The encryption profile can be the
             provider profile or customer profile.";
        }
        description
          "Container for encryption profile.";
      }
      description
        "Site-specific security parameters.";
    }
    container routing-protocols {
      list routing-protocol {
        key "id";
        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";
          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.";
          }
          description
            "Routing profiles.";
        }
```

```
container ospf {
  when "derived-from-or-self(../type, "
     + "'vpn-common:ospf')" {
    description
      "Only applies when protocol is OSPF.";
  }
  if-feature "vpn-common:rtg-ospf";
  leaf-list address-family {
    type vpn-common:address-family;
    min-elements 1;
    description
      "If OSPF is used on this site, this node
       contains a configured value.  This node
       contains at least one address family
       to be activated.";
  }
  leaf area-address {
    type yang:dotted-quad;
    mandatory true;
    description
      "Area address.";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the PE-CE link.  It is used
       in the routing state calculation and
       path selection.";
  }
  leaf mtu {
    type uint16;
    description
      "Maximum transmission unit for a given
       OSPF link.";
  }
  leaf process-id {
    type uint16;
    description
      "Process id of the OSPF CE-PE connection.";
  }
  uses security-params;
  container sham-links {
    if-feature "vpn-common:rtg-ospf-sham-link";
    list sham-link {
      key "target-site";
      leaf target-site {
        type vpn-common:vpn-id;
```

```
            description
              "Target site for the sham link connection.
               The site is referred to by its ID.";
          }
          leaf metric {
            type uint16;
            default "1";
            description
              "Metric of the sham link.  It is used in
               the routing state calculation and path
               selection.  The default value is set
               to 1.";
          }
          description
            "Creates a sham link with another site.";
        }
        description
          "List of sham links.";
      }
      description
        "OSPF-specific configuration.";
    }
    container bgp {
      when "derived-from-or-self(../type, "
         + "'vpn-common:bgp')" {
        description
          "Only applies when protocol is BGP.";
      }
      if-feature "vpn-common:rtg-bgp";
      leaf peer-autonomous-system {
        type inet:as-number;
        mandatory true;
        description
          "Indicates the Customer's AS Number (ASN) in
           case the Customer requests BGP routing.";
      }
      leaf local-autonomous-system {
        type inet:as-number;
        description
          "Is set to the ASN to override a peers' ASN
           if such feature is requested by the
           Customer.";
      }
      leaf-list address-family {
        type vpn-common:address-family;
        min-elements 1;
        description
          "This node contains at least one
```

```
        address-family to be activated.";
    }
    leaf-list neighbor {
      type inet:ip-address;
      description
        "IP address(es) of the BGP neighbor. IPv4
         and IPv6 neighbors may be indicated if
         two sessions will be used for IPv4 and
         IPv6.";
    }
    leaf multihop {
      type uint8;
      description
        "Describes the number of IP hops allowed
         between a given BGP neighbor and the PE.";
    }
    uses security-params;
    uses vpn-common:service-status;
    leaf description {
      type string;
      description
        "Includes a description of the BGP session.
         Such description is meant to be used for
         diagnosis purposes. The semantic of the
         description is local to an
         implementation.";
    }
    leaf as-override {
      type boolean;
      default "false";
      description
        "Defines whether AS override is enabled,
         i.e., replace the ASN of the peer specified
         in the AS Path attribute with the local
         AS number.";
    }
    leaf default-route {
      type boolean;
      default "false";
      description
        "Defines whether default route(s) can be
         advertised to its peer. If set, the
         default route(s) is advertised to its
         peer.";
    }
    container bgp-max-prefix {
      leaf max-prefix {
        type uint32;
```

```
        default "5000";
        description
          "Indicates the maximimum number of BGP
           prefixes allowed in the BGP session.

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

           If the limit is exceeded, the session
            can be teared down.";
        reference
          "RFC4271, Section 8.2.2.";
      }
      leaf warning-threshold {
        type decimal64 {
          fraction-digits 5;
          range "0..100";
        }
        units "percent";
        default "75";
        description
          "When this value is reached, a warning
           notification will be triggered.";
      }
      leaf violate-action {
        type enumeration {
          enum warning {
            description
              "Only a warning message is sent to
               the peer when the limit is
               exceeded.";
          }
          enum discard-extra-paths {
            description
              "Discards extra paths when the
               limit is exceeded.";
          }
          enum restart {
            description
              "Restart after a time interval.";
          }
        }
        description
          "BGP neighbour max-prefix violate
           action";
      }
      leaf restart-interval {
        type uint16;
```

```
        units "minutes";
        description
          "Time interval (min) after which the
           BGP session will be reestablished.";
      }
      description
        "Controls the behavior when a prefix
         maximum is reached.";
    }
    container bgp-timer {
      description
        "Includes two BGP timers that can be
         customized when building a VPN service
         with BGP used as CE-PE routing
         protocol.";
      leaf keep-alive {
        type uint16 {
          range "0..21845";
        }
        units "seconds";
        default "30";
        description
          "This timer indicates the KEEPALIVE
           messages'  frequency between a PE
           and a BGP peer.

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

           It is suggested that the maximum time
           between  KEEPALIVEmessages would be
           one third of the Hold Time interval.";
        reference
          "Section 4.4 of RFC 4271";
      }
      leaf hold-time {
        type uint16 {
          range "0 | 3..65535";
        }
        units "seconds";
        default "90";
        description
          "It indicates the maximum number of
          seconds that may elapse between the
          receipt of successive KEEPALIVE
          and/or UPDATE   messages from the peer.

          The Hold Time must be either zero or
```

```
            at least three seconds.";
          reference
            "Section 4.2 of RFC 4271";
        }
      }
      description
        "BGP-specific configuration.";
    }
    container isis {
      when "derived-from-or-self(../type, "
         + "'vpn-common:isis')" {
        description
          "Only applies when protocol is IS-IS.";
      }
      if-feature "vpn-common:rtg-isis";
      leaf-list address-family {
        type vpn-common:address-family;
        min-elements 1;
        description
          "If ISIS is used on this site, this node
           contains a configured value.  This node
           contains at least one address family
           to be activated.";
      }
      leaf area-address {
        type yang:dotted-quad;
        mandatory true;
        description
          "Area address.";
      }
      leaf level {
        type identityref {
          base isis-level;
        }
        description
          "level1, level2 or level1-2";
      }
      leaf metric {
        type uint16;
        default "1";
        description
          "Metric of the PE-CE link.  It is used
           in the routing state calculation and
           path selection.";
      }
      leaf process-id {
        type uint16;
        description
```

```
            "Process id of the IS-IS CE-PE
             connection.";
          }
          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.";
          }
          uses vpn-common:service-status;
          description
            "IS-IS specific configuration.";
        }
        container static {
          when "derived-from-or-self(../type, "
             + "'vpn-common:static')" {
            description
              "Only applies when protocol is static.
               BGP activation requires the SP to know
               the address of the customer peer.  When
               BGP is enabled, the 'static-address'
               allocation type for the IP connection
               must be used.";
          }
          container cascaded-lan-prefixes {
            list ipv4-lan-prefixes {
              if-feature "vpn-common:ipv4";
              key "lan next-hop";
              leaf lan {
                type inet:ipv4-prefix;
                description
                  "LAN prefixes.";
              }
              leaf lan-tag {
                type string;
                description
```

```
                    "Internal tag to be used in VPN
                     policies.";
                }
                leaf next-hop {
                  type inet:ipv4-address;
                  description
                    "Next-hop address to use on the
                     customer side.";
                }
                description
                  "List of LAN prefixes for the site.";
              }
              list ipv6-lan-prefixes {
                if-feature "vpn-common:ipv6";
                key "lan next-hop";
                leaf lan {
                  type inet:ipv6-prefix;
                  description
                    "LAN prefixes.";
                }
                leaf lan-tag {
                  type string;
                  description
                    "Internal tag to be used in VPN
                     policies.";
                }
                leaf next-hop {
                  type inet:ipv6-address;
                  description
                    "Next-hop address to use on the
                     customer side.";
                }
                description
                  "List of LAN prefixes for the site.";
              }
              description
                "LAN prefixes from the customer.";
            }
            description
              "Configuration specific to static routing.";
          }
          container rip {
            when "derived-from-or-self(../type, "
               + "'vpn-common:rip')" {
              description
                "Only applies when the protocol is RIP.
                 For IPv4, the model assumes that RIP
                 version 2 is used.";
```

```
        }
        if-feature "vpn-common:rtg-rip";
        leaf-list address-family {
          type vpn-common:address-family;
          min-elements 1;
          description
            "If RIP is used on this site, this node
             contains a configured value.  This node
             contains at least one address family
             to be activated.";
        }
        description
          "Configuration specific to RIP routing.";
      }
      container vrrp {
        when "derived-from-or-self(../type, "
           + "'vpn-common:vrrp')" {
          description
            "Only applies when protocol is VRRP.";
        }
        if-feature "vpn-common:rtg-vrrp";
        leaf-list address-family {
          type vpn-common:address-family;
          min-elements 1;
          description
            "If VRRP is used on this site, this node
             contains a configured value.  This node
             contains at least one address family to
             be activated.";
        }
        leaf vrrp-group {
          type uint8 {
            range "1..255";
          }
          description
            "VRRP group number";
        }
        leaf backup-peer {
          type inet:ip-address;
          description
            "IP address of the peer";
        }
        leaf priority {
          type uint8;
          description
            "Local priority of the VRRP speaker";
        }
        leaf ping-reply {
```

```
            type boolean;
            description
              "Whether the VRRP speaker should answer
               to ping requests";
          }
          description
            "Configuration specific to VRRP.";
        }
        description
          "List of routing protocols used on
           the site.  This list can be augmented.";
      }
      description
        "Defines routing protocols.";
    }
    container service {
      leaf svc-input-bandwidth {
        type uint64;
        units "bps";
        mandatory true;
        description
          "From the customer site's perspective, the
           service input bandwidth of the connection
           or download bandwidth from the SP to
           the site.";
      }
      leaf svc-output-bandwidth {
        type uint64;
        units "bps";
        mandatory true;
        description
          "From the customer site's perspective,
           the service output bandwidth of the
           connection or upload bandwidth from
           the site to the SP.";
      }
      leaf svc-mtu {
        type uint16;
        units "bytes";
        mandatory true;
        description
          "MTU at service level.  If the service is IP,
           it refers to the IP MTU.  If CsC is enabled,
           the requested 'svc-mtu' leaf will refer
            to the MPLS MTU and not to the IP MTU.";
      }
      container qos {
        if-feature "vpn-common:qos";
```

```
                container qos-classification-policy {
                  list rule {
                    key "id";
                    ordered-by user;
                    leaf id {
                      type string;
                      description
                        "A description identifying the
                         qos-classification-policy rule.";
                    }
                    choice match-type {
                      default "match-flow";
                      case match-flow {
                        choice l3 {
                          container ipv4 {
                            uses pf:acl-ip-header-fields;
                            uses pf:acl-ipv4-header-fields;
                            description
                              "Rule set that matches IPv4 header.";
                          }
                          container ipv6 {
                            uses pf:acl-ip-header-fields;
                            uses pf:acl-ipv6-header-fields;
                            description
                              "Rule set that matches IPv6 header.";
                          }
                          description
                            "Either IPv4 or IPv6.";
                        }
                        choice l4 {
                          container tcp {
                            uses pf:acl-tcp-header-fields;
                            uses ports;
                            description
                              "Rule set that matches TCP header.";
                          }
                          container udp {
                            uses pf:acl-udp-header-fields;
                            uses ports;
                            description
                              "Rule set that matches UDP header.";
                          }
                          description
                            "Can be TCP or UDP";
                        }
                      }
                      case match-application {
                        leaf match-application {
```

```
                type identityref {
                  base vpn-common:customer-application;
                }
                description
                  "Defines the application to match.";
              }
            }
            description
              "Choice for classification.";
          }
          leaf target-class-id {
            type string;
            description
              "Identification of the class of service.
               This identifier is internal to the
               administration.";
          }
          description
            "List of marking rules.";
        }
        description
          "Configuration of the traffic classification
           policy.";
      }
      container qos-profile {
        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.";
```

```
          }
        }
        description
          "QoS profile configuration.";
      }
      description
        "QoS configuration.";
    }
    container carrierscarrier {
      if-feature "vpn-common:carrierscarrier";
      leaf signalling-type {
        type enumeration {
          enum ldp {
            description
              "Use LDP as the signalling protocol
               between the PE and the CE.  In this
               case, an IGP routing protocol must
               also be activated.";
          }
          enum bgp {
            description
              "Use BGP as the signalling protocol
               between the PE and the CE.
               In this case, BGP must also be configured
               as the routing protocol.";
            reference
              "RFC 8277: Using BGP to Bind MPLS Labels
                         to Address Prefixes";
          }
        }
        default "bgp";
        description
          "MPLS signalling type.";
      }
      description
        "This container is used when the customer
         provides MPLS-based services.  This is
         only used in the  case of CsC (i.e., a
         customer builds an MPLSservice using an
         IP VPN to carry its traffic).";
    }
    container multicast {
      if-feature "vpn-common:multicast";
      leaf site-type {
        type enumeration {
          enum receiver-only {
            description
              "The site only has receivers.";
```

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

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

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

```
                    "Multicast protocol type to be used with
                     the customer site.";
                }
                leaf remote-source {
                  type boolean;
                  default "false";
                  description
                    "When true, there is no PIM adjacency on
                     the interface.";
                }
                description
                  "Multicast parameters for the site.";
              }
              description
                "Service parameters on the attachment.";
            }
            description
              "List of accesses for a site.";
          }
          description
            "List of accesses for a site.";
        }
        container maximum-routes {
          list address-family {
            key "af";
            leaf af {
              type vpn-common:address-family;
              description
                "Indicates the address family
                (IPv4 or IPv6).";
            }
            leaf maximum-routes {
              type uint32;
              description
                "Indicates the maximum prefixes the VRF
                 can accept for this address family.";
            }
            description
              "List of address families.";
          }
          description
            "Defines 'maximum-routes' for the VRF.";
        }
        container multicast {
          if-feature "vpn-common:multicast";
          leaf enabled {
            type boolean;
            default "false";
```

```
            description
              "Enables multicast.";
          }
          leaf-list tree-flavor {
            type identityref {
              base vpn-common:multicast-tree-type;
            }
            description
              "Type of tree to be used.";
          }
          container rp {
            container rp-group-mappings {
              list rp-group-mapping {
                key "id";
                leaf id {
                  type uint16;
                  description
                    "Unique identifier for the mapping.";
                }
                container provider-managed {
                  leaf enabled {
                    type boolean;
                    default "false";
                    description
                      "Set to true if the Rendezvous Point (RP)
                       must be a provider-managed node.  Set to
                       false if it is a customer-managed node.";
                  }
                  leaf rp-redundancy {
                    type boolean;
                    default "false";
                    description
                      "If true, a redundancy mechanism for the
                       RP is required.";
                  }
                  leaf optimal-traffic-delivery {
                    type boolean;
                    default "false";
                    description
                      "If true, the SP must ensure that
                       traffic uses an optimal path.  An SP may
                       use Anycast RP or RP-tree-to-SPT
                       switchover architectures.";
                  }
                  container anycast {
                    when "../rp-redundancy = 'true' and
                          ../optimal-traffic-delivery = 'true'" {
                      description
```

```
            "Only applicable if
             RP redundancy is
             enabled and delivery through
             optimal path is activated.";
        }
        leaf local-address {
          type inet:ip-address;
          description
            "IP local address for PIM RP.
             Usually, it corresponds to router
             ID or primary address";
        }
        leaf-list rp-set-address {
          type inet:ip-address;
          description
            "Address other RP routers
             that share the same RP IP address.";
        }
        description
          "PIM Anycast-RP parameters.";
      }
      description
        "Parameters for a provider-managed RP.";
    }
    leaf rp-address {
      when "../provider-managed/enabled = 'false'" {
        description
          "Relevant when the RP is not
           provider-managed.";
      }
      type inet:ip-address;
      mandatory true;
      description
        "Defines the address of the RP.
         Used if the RP is customer-managed.";
    }
    container groups {
      list group {
        key "id";
        leaf id {
          type uint16;
          description
            "Identifier for the group.";
        }
        choice group-format {
          mandatory true;
          case group-prefix {
            leaf group-address {
```

```
                        type inet:ip-prefix;
                        description
                          "A single multicast group prefix.";
                      }
                    }
                    case startend {
                      leaf group-start {
                        type inet:ip-address;
                        description
                          "The first multicast group address in
                           the multicast group address range.";
                      }
                      leaf group-end {
                        type inet:ip-address;
                        description
                          "The last multicast group address in
                           the multicast group address range.";
                      }
                    }
                    description
                      "Choice for multicast group format.";
                  }
                  description
                    "List of multicast groups.";
                }
                description
                  "Multicast groups associated with the RP.";
              }
              description
                "List of RP-to-group mappings.";
            }
            description
              "RP-to-group mappings parameters.";
          }
          container rp-discovery {
            leaf rp-discovery-type {
              type identityref {
                base vpn-common:multicast-rp-discovery-type;
              }
              default "vpn-common:static-rp";
              description
                "Type of RP discovery used.";
            }
            container bsr-candidates {
              when "derived-from-or-self(../rp-discovery-type, "
                 + "'vpn-common:bsr-rp')" {
              description
                "Only applicable if discovery type
```

```
                    is BSR-RP.";
                }
                leaf-list bsr-candidate-address {
                  type inet:ip-address;
                  description
                    "Address of candidate Bootstrap Router
                    (BSR).";
                }
                description
                  "Container for List of Customer
                   BSR candidate's addresses.";
              }
              description
                "RP discovery parameters.";
            }
            description
              "RP parameters.";
          }
          container msdp {
            if-feature "msdp";
            leaf enabled {
              type boolean;
              default "false";
              description
                "If true, MSDP is activated.";
            }
            leaf peer {
              type inet:ip-address;
              description
                "IP address of the MSDP peer.";
            }
            leaf local-address {
              type inet:ip-address;
              description
                "IP address of the local end. This local
                 address must be configured on the
                 node.";
            }
            description
              "MSDP parameters.";
          }
          description
            "Multicast global parameters for the VPN
             service.";
        }
        description
          "List for VPN node.";
      }
```

```
        }
        description
          "List of VPN services.";
      }
      description
        "Top-level container for the VPN services.";
    }
    description
      "Main container for L3VPN services management.";
  }
}
<CODE ENDS>
```

Figure 25

9. IANA Considerations

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

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

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

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

10. Security Considerations

The YANG module specified in this document defines schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040] . The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8466].

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

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

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

- 'vpn-service': An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L3VPN Service, interrupting all the traffic of a client. In addition, an attacker may modify the attributes of a running service (e.g., QoS, bandwidth, routing protocols), leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create a L3VPN Service or adding a new network access. Such activity can be detected by adequately monitoring and tracking network configuration changes.

Some of the readable data nodes in the "ietf-l3vpn-ntw" module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

- Malicious clients attempting to delete or modify VPN services.

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

- Unauthorized clients attempting to read VPN service related information.

11. Acknowledgements

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

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

12. Contributors

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

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

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

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

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

Stephane Litkowski
Cisco
Email: slitkows@cisco.com>

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

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

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

13. References

13.1. Normative References

[I-D.ietf-opsawg-vpn-common]
barguil, s., Dios, O., Boucadair, M., and Q. WU, "A Layer 2/3 VPN Common YANG Model", draft-ietf-opsawg-vpn-common-01 (work in progress), September 2020.

[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>.

[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>.

[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>.

[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>.

[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>.

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

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

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

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

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

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

13.2. Informative References

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

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-09 (work in progress), June 2020.

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

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

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

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

[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>.

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 26, 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'.

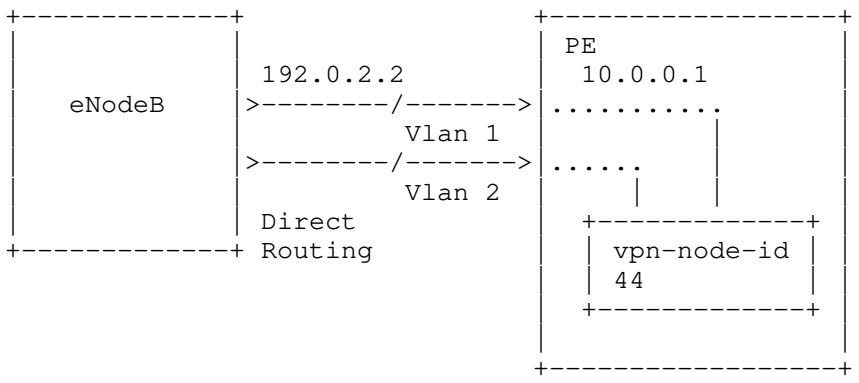


Figure 26: Mobile Backhaul Example

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

First: Create the 4G VPN Service (Figure 27).

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/vpn-services
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "4G",
        "customer-name": "mycustomer",
        "vpn-service-topology": "custom",
        "description": "VPN to deploy 4G services"
      }
    ]
  }
}
```

Figure 27: Create VPN Service

Second: Create a VPN Node as depicted in Figure 28. In this type of service, the VPN Node is equivalent to the VRF configured in the physical device ('ne-id'=10.0.0.1).

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-nodes": {
    "vpn-node": [
      {
        "vpn-node-id": "44",
        "ne-id": "10.0.0.1",
        "local-autonomous-system": "65550",
        "rd": "0:65550:1",
        "vpn-targets": {
          "vpn-target": [
            {
              "id": "1",
              "route-targets": [
                "0:65550:1"
              ],
              "route-target-type": "both"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 28: Create VPN Node

Finally, two VPN Network Accesses are created using the same physical port ('port-id'=1/1/1). Each 'vpn-network-access' has a particular VLAN (1,2) to differentiate the traffic between: Sync and data (Figure 29).

```
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/\
      vpn-services/vpn-service=4G/vpn-nodes/vpn-node=44
content-type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "vpn-network-access-id": "1/1/1.1",
        "port-id": "1/1/1",
        "description": "Interface SYNC to eNODE-B",
        "status": {
```

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

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

Figure 29: Create VPN Network Access

A.2. Multicast VPN Provisioning Example

IPTV is mainly distributed through multicast over the LANs. In the following example, PIM-SM is enabled and functional between the PE and the CE. The PE receives multicast traffic from a CE that is directly connected to the multicast source. The signaling between PE and CE is achieved using BGP. Also, RP is statically configured for a multicast group.

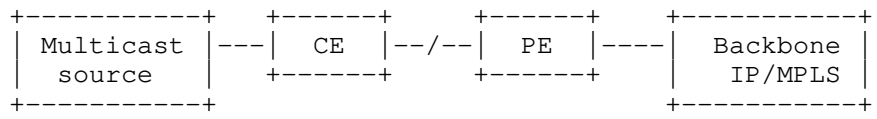


Figure 30: Multicast L3VPN Service Example

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

First, the multicast service is created (see the excerpt of the request message body shown in Figure 31)

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

Figure 31: 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 32). In this example, the VPN Node will represent VRF configured in the physical device.

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

Figure 32: 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 33).

```
 {
  "ietf-l3vpn-ntw:vpn-network-access": {
    "vpn-network-access-id": "1/1/1",
```

```
      "description": "Connected_to_source",
      "status": {
        "admin-enabled": "true"
      },
      "vpn-network-access-type": "vpn-common:point-to-point",
      "ip-connection": {
        "ipv4": {
          "address-allocation-type": "static-address",
          "static-addresses": {
            "primary-address": "1",
            "address": [
              {
                "address-id": "1",
                "s-provider-address": "172.19.48.1",
                "s-prefix-length": 30
              }
            ]
          }
        }
      },
      "routing-protocols": {
        "routing-protocol": [
          {
            "id": "1",
            "type": "vpn-common:bgp",
            "bgp": {
              "peer-autonomous-system": "6500",
              "local-autonomous-system": "3816",
              "address-family": "ipv4",
              "neighbor": "172.19.48.2",
              "description": "Connected to CE"
            }
          }
        ]
      },
      "service": {
        "multicast": {
          "multicast-site-type": "source-only",
          "multicast-address-family": {
            "ipv4": "true"
          },
          "protocol-type": "router"
        }
      }
    }
  }
```

Figure 33: Create VPN Network Access (Excerpt of the Message Request Body)

Appendix B. Implementation Status

This section records the status of known implementations of the Yang module defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. 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 [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note the RFC Editor: As per [RFC6982] guidelines, please remove this Implementation Status apendix prior publication.

B.1. Nokia Implementation

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

B.2. Huawei Implementation

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

B.3. Infinera Implementation

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

B.4. Ribbon-ECI Implementation

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

Authors' Addresses

Samier Barguil
Telefonica
Madrid
ES

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone
ES

Email: luis-angel.munoz@vodafone.com

Alejandro Aguado
Nokia
Madrid
ES

Email: alejandro.aguado_martin@nokia.com

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.

Table of Contents

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 versionioning 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:

- on devices themselves
- on a web site (e.g., via URI)
- 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.

- Is the model structured correctly?

- Are there other retrieval mechanisms that need to be specified?
- Do we need to be more specific in how to authenticate and retrieve SBOMs?
- 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-acldns: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:

- 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

Soure 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 eihter automated implemetation 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.

Table of Contents

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 validataion. 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 granualarity 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) boarder 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 imcoming interface is different from the out-interface of the packet source IP address, using the longest prefix match , also refered to as asymmetric routing of data packets, Strict uRPF exibits 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 algorihtms (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 exibit false postive 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 exibits 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.

- 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.

- 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 soltuion, one should never rely on a single point but a systematic SAV technique combination depoyled 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 administrive 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 filetering. In the rest of this document, we use the term SAVA (SAV architecture) to refer to the discussion of the systematic SAV solution.

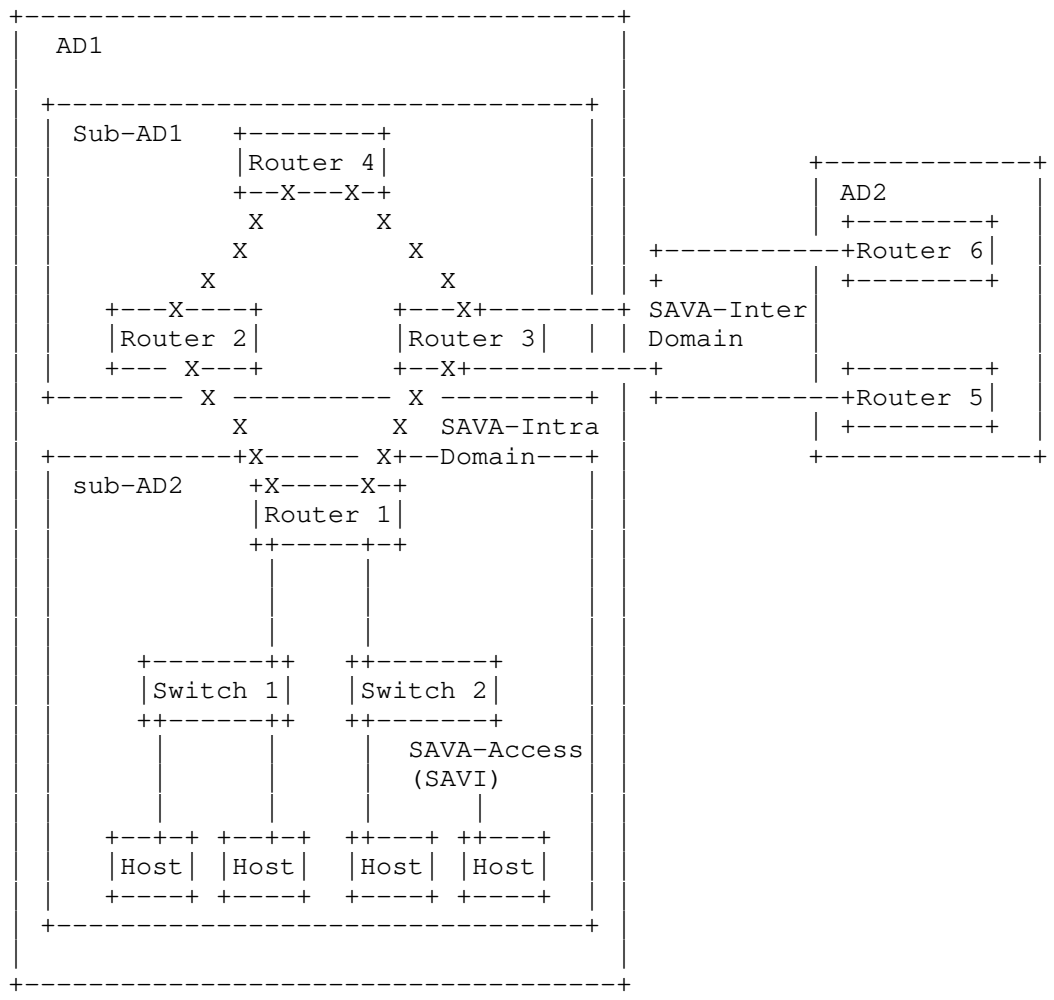


Figure 1: SAVA

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: Boarder 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. Supposely, 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.

- Loose uRPF: works for Flow 1, but fails for Flow 2.
- Strict uRPF: works for Flow 2, but fails for Flow 1 (the incoming interface does not match P1/P2's out-interface).
- FP-uRFP: 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.

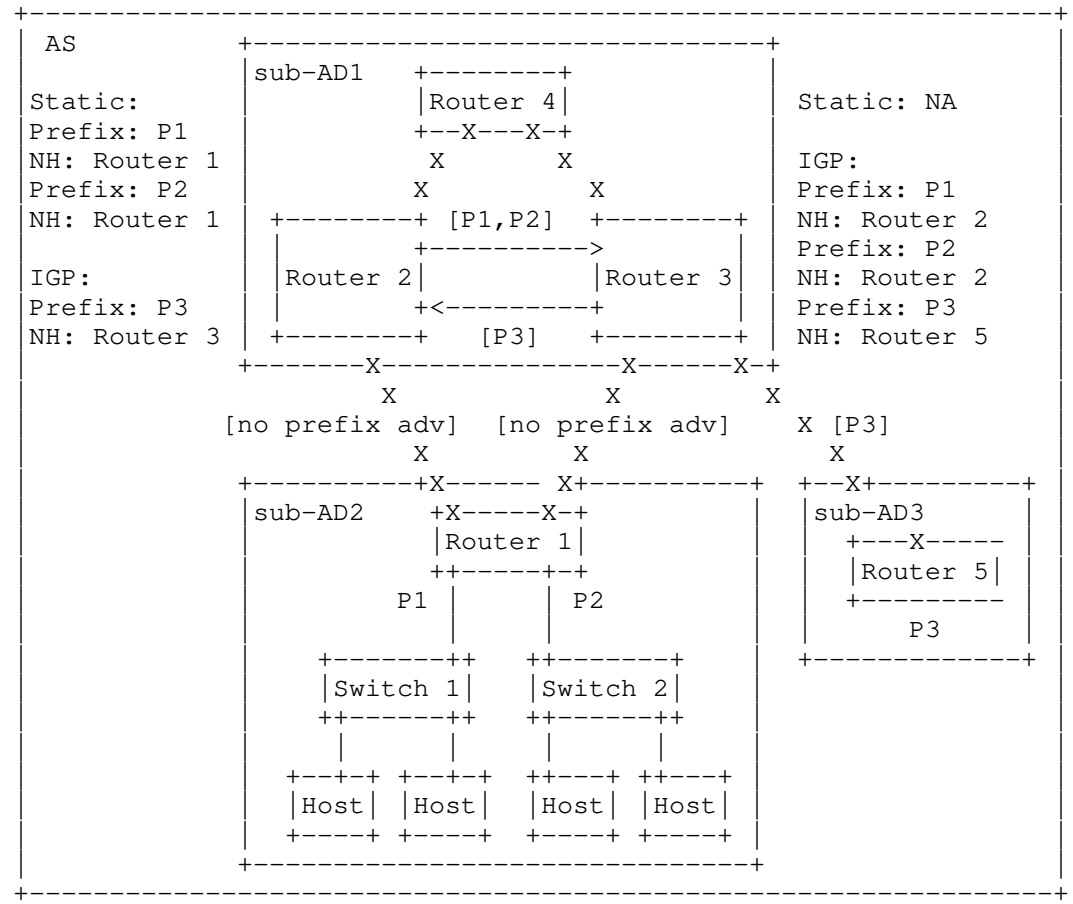


Figure 2: Asymmetric data flow in the Intra-AS scenario

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.

- Loose uRPF: works for Flow 1, but fails for Flow 2.
- Strict uRPF: works for Flow 2, but fails for Flow 1 (the incoming interface does not match P1/P2's out-interface).
- FP-uRFP: works for Flow 2, but fails for Flow 1 (no feasible path for P1/P2 other than the best route exists).
- 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.

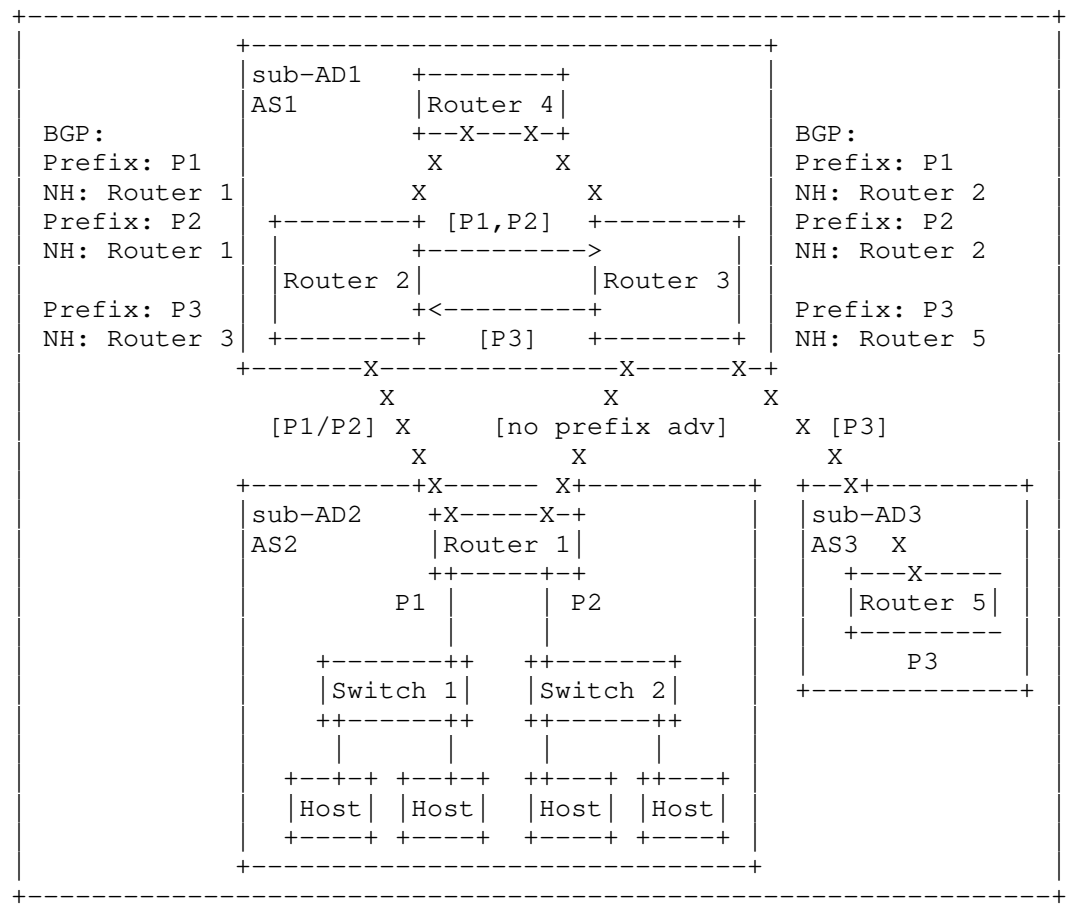


Figure 3: Asymmetric data flow in the Inter-AS scenario

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 databeses provides a way for the multiple transit providers of the

same multihomed network to share such information without extra way of data syncronization.

In addition, although ERP-uRPF is striving for more accurate RPF list construction, there's still currenly 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 publiclly 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-rtgwg-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", draft-openconfig-rtgwg-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: March 16, 2021

T. Graf
Swisscom

September 12, 2020

Export of MPLS Segment Routing Label Type Information in IP Flow Information Export (IPFIX) draft-tgraf-ipfix-mpls-sr-label-type-05

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 March 16, 2021.

Table of Contents

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 dataplane [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 #46 mplsTopLabelType 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 IANA Information Element #46 SubRegistry [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 #46 mplsTopLabelType 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 dataplane to above discribed use case.

Both use cases can be verified by looking at IE46, mplsTopLabelType, IE47, mplsTopLabelIPv4Address, IE70, mplsTopLabelStackSection and IE89 forwardingStatus dimensions. Giving insights into the MPLS dataplane for which MPLS provider edge loopback address, which label protocol has been used and how many packets are forwarded or dropped and when dropped why they have been dropped.

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
TBD1	OSPFv2 Segment Routing	RFC8665
TBD2	OSPFv3 Segment Routing	RFC8666
TBD3	IS-IS Segment Routing	RFC8667
TBD4	BGP Segment Routing Prefix-SID	RFC8669

Figure 1: Updates to "IPFIX Information Element #46" SubRegistry

4. Security Considerations

The same security considerations apply as for the IPFIX Protocol RFC7012 [RFC7012].

5. Acknowledgements

I would like to thank Paul Aitken, 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., and J. Uttaro, "Seamless Segment Routing", draft-hegde-spring-mpls-seamless-sr-01 (work in progress), July 2020.

[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.

Table of Contents

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.

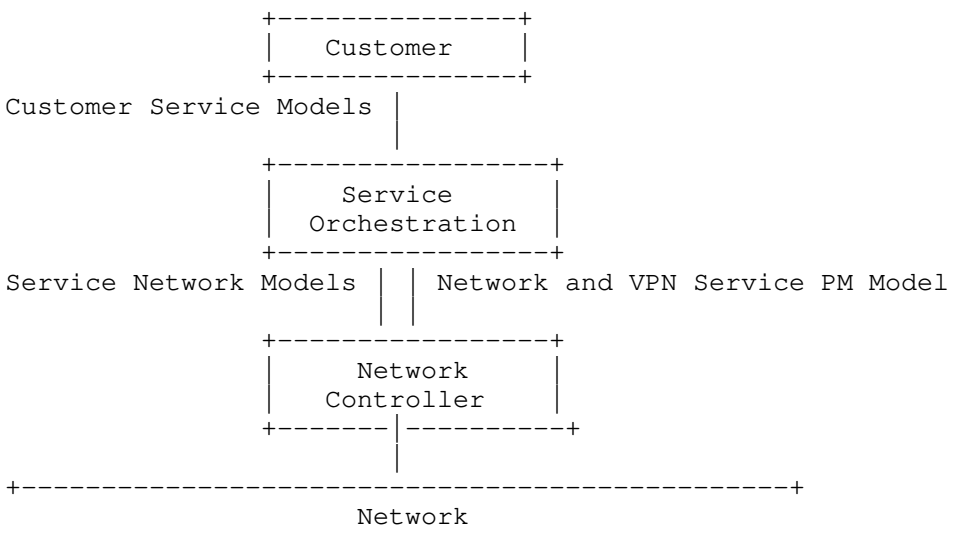


Figure 1: Reference Architecture

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.

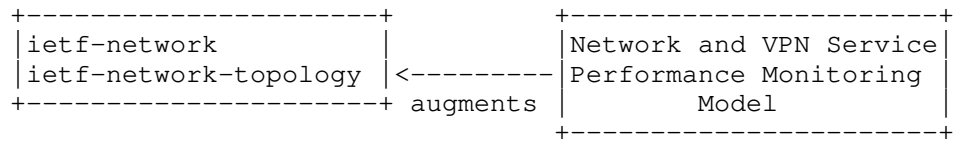


Figure 2: Module Augmentation

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:

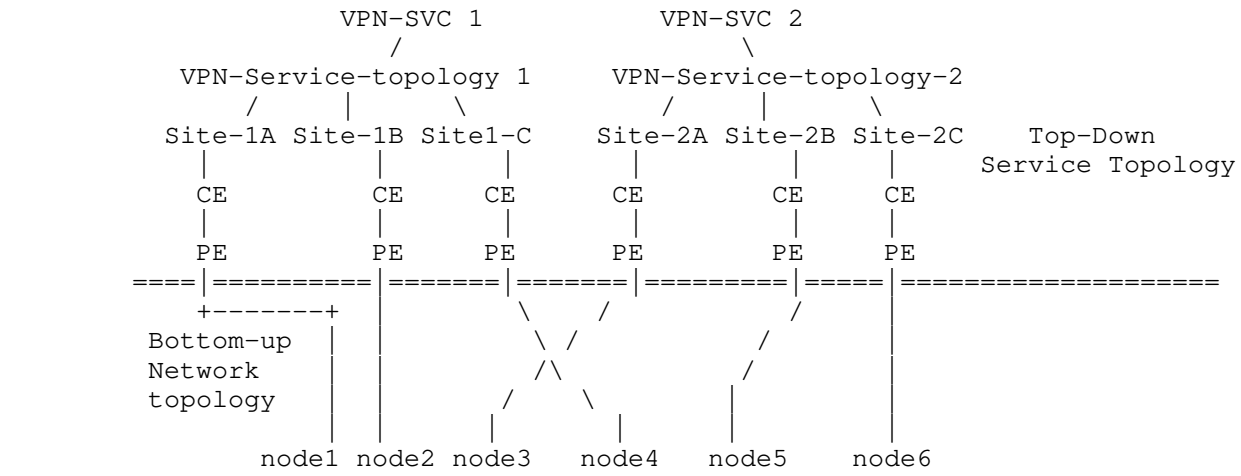


Figure 3: Example of topology mapping between VPN Service Topo and Underlying network

As shown in Figure 3, two VPN services topologies are both built on top of one common underlying physical network:

- 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.

- 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:

- "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.

- "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:

- "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.

- "site-id" (Constraint): Uniquely identifies the site within the overall network infrastructure.

- "site-role" (Constraint): Defines the role of the site in a particular VPN topology.

- "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:

- "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.

- 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.

- 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.

- 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.

- 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:

- 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.

- 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-attribtues>
                  <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-attribtues>
                    <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-attribtues>
           <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-attribtues>
           <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
             <lana.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:

- /nw:networks/nw:network/svc-topo:svc-telemetry-attributes

- /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: lana.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