Network Monitoring For IGP
draft-gu-opsawg-network-monitoring-igp-01

Abstract

To evolve towards automated network OAM (Operations, administration and management), the monitoring of control plane protocols is a fundamental necessity. This document proposes network monitoring for IGP to facilitate troubleshooting by collecting the IGP monitoring data and reporting it to the network monitoring server in real-time. In this document, the operations of network monitoring for ISIS are described, and the corresponding network monitoring message types and message formats are defined.

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 August 23, 2021.
1. Introduction

1.1. Motivation

The requirement for better network OAM approaches has been greatly driven by the network evolvement. The concept of network Telemetry has been proposed to meet the current and future OAM demands w.r.t.,
massive and real-time data storage, collection, process, export, and analysis, and an architectural framework of existing Telemetry approaches is introduced in [I-D.song-ntf]. Network Telemetry provides visibility to the network health conditions, and is beneficial for faster network troubleshooting, network OpEx (operating expenditure) reduction, and network optimization. Telemetry can be applied to the data plane, control plane and management plane. There have been various methods proposed for each plane:

- **Management plane**: For example, SNMP (Simple Network Management Protocol) [RFC1157], NETCONF (Network Configuration Protocol) [RFC6241] and gNMI (gRPC Network Management Interface) [I-D.openconfig-rtgwg-gnmi-spec] are three typical widely adopted management plane Telemetry approaches. Various YANG modules are defined for network operational state retrieval and configuration management. Subscription to specific YANG datastore can be realized in combination with gRPC/NETCONF.

- **Data plane**: For example, In-situ OAM (iOAM) [I-D.brockners-inband-oam-requirements] embeds an instruction header to the user data packets, and collects the requested data and adds it to the use packet at each network node along the forwarding path. Applications such as path verification, SLA (service-level agreement) assurance can be enabled with iOAM.

- **Control Plane**: BGP monitoring protocol (BMP) [RFC7854] is proposed to monitor BGP sessions and intended to provide a convenient interface for obtaining BGP route views. Data collected using BMP can be further analyzed with big data platforms for network health condition visualization, diagnose and prediction applications.

The general idea of most Telemetry approaches is to collect various information from devices and export to the centralized server for further analysis, and thus providing more network insight. It should not be surprising that any future and even current Telemetry applications may require the fusion of data acquired from more than one single approach/one single plane. For example, for network troubleshooting purposes, it requires the collection of comprehensive information from devices, such system ID/router ID, interface status, PDUs (protocol data units), device/protocol statistics and so on. Information such as system ID/router ID can be reported by management plane Telemetry approaches, while the protocol related data (especially PDUs) are more fit to be monitored using the control plane Telemetry. With rich information collected in real time at the centralized server, network issues can be localized faster and more accurately, and the root cause analysis can be also provided.
The conventional troubleshooting logic is to log in a faulty router, physically or through Telnet, and by using CLI to display related information/logs for fault source localization and further analysis. There are several concerns with the conventional troubleshooting methods:

1. It requires rich OAM experience for the OAM operator to know what information to check on the device, and the operation is complex;

2. In a multi-vendor network, it requires the understanding and familiarity of vendor specific operations and configurations;

3. Locating the fault source device could be non-trivial work, and is often realized through network-wide device-by-device check, which is both time-consuming and labor-consuming; and finally,

4. The acquisition of troubleshooting data can be difficult under some cases, e.g., when auto recovery is used.

This document proposes the network monitoring for IGP to monitor the running state of IGP, e.g., PDUs, protocol statistics and peer status, which have not been systematically covered by any other Telemetry approach, to facilitate network troubleshooting.

1.2. Overview

Like BMP, a networking monitoring session is established between each monitored router (NM client) and the NM monitoring station (NM server) through TCP connection. Information are collected directly from each monitored router and reported to the NM server. The NM message can be both periodic and event-triggered, depending on the message type.

IS-IS [RFC1195], as one of the most commonly adopted network layer protocols, builds the fundamental network connectivity of an autonomous system (AS). The disfunction of IS-IS, e.g., IS-IS neighbor down, route flapping, MTU mismatch, and so on, could lead to network-wide instability and service interruption. Thus, it is critical to keep track of the health condition of IS-IS, and the availability of information, related to IS-IS running status, is the fundamental requirement. In this document, typical network issues are identified as the use cases of network monitoring. Then the operations and the message formats of network monitoring for IS-IS are defined. Network monitoring for OSPF will be included in the future version.
2. Terminology

IGP: Interior Gateway Protocol
IS-IS: Intermediate System to Intermediate System
NM: Network Monitoring
IMP: Network Monitoring for IGP
BMP: BGP monitoring protocol
IIH: IS-IS Hello Packet
LSP: Link State Packet
CSNP: Complete Sequence Number Packet
NSNP: Partial Sequence Number Packet

3. Use Cases

We have identified two typical network issues due to IS-IS disfunction that are currently difficult to detect or localize.

3.1. IS-IS Route Flapping

The IS-IS Route Flapping refers to the situation that one or more routes appear and then disappear in the routing table repeatedly. Route flapping usually comes with massive PDUs interactions (e.g., LSP, LSP purge...), which consume excessive network bandwidth, and excessive CPU processing. In addition, the impact is often network-wide. The localizing of the flapping source and the identifying of root causes haven’t been easy work due to various reasons.

The flapping can be caused by system ID conflict, IS-IS neighborship flapping, route source flapping (caused by import route policy misconfiguration), device clock dis-function with abnormal LSP purge (e.g., 100 times faster) and so on.

- The system ID conflict check is a network-wide work. If such information is collected centrally to a controller/server, the issues can be identified in seconds, and more importantly, in advance of the actual flapping event.

- The IS-IS neighborship flapping is typically caused by interface flapping, BFD flapping, CPU high and so on. Conventionally, to located the issue, operators typically identify the target...
device(s), and then log in the devices to check related statistics, parsed protocol PDU data and configurations. The manual check often requires a combination of multiple CLIs (check cost/next hop/exit interface/LSP age...) in a repeated manner, which is time-consuming and requires rich OAM experience. If such statistics and configuration data were collected at the server in real-time, the server may analyze them automatically or semi-automatically with troubleshooting algorithms implemented at the server.

- In the case that route policies are misconfigured, which then causes the route flapping, it’s typically difficult to directly identify the responsible policy in a short time. Thus, if the route change history is recorded in correlation with the route policy, then with such record collected at the server, the server can directly identify the responsible policy with the one-to-one mapping between policy processing and the route attribute change.

- In the case that flapping comes with abnormal LSP purges, it may be due to continuous LSP corruptions with falsified shorter Remaining Lifetime, or the clock running 100 times faster with 100 times more purge LSPs generated. In order to identify the purge originator, RFC 6232 [RFC6232] proposes to carry the Purge Originator Identification (POI) TLV in IS-IS. However, to analyze the root cause of such abnormal purges, the collection and analysis of LSP PDUs are needed.

3.2. IS-IS LSDB Synchronization Failure

During the IS-IS flooding, sometimes the LSP synchronization failure happens. The synchronization failure causes can be generally classified into three cases:

- Case 1, the LSP is not correctly advertised. For example, an LSP sent by Router A fails to be synchronized at Router B. It can be due to incorrect route export policy, or too many prefixes being advertised which exceeds the LSP/MTU threshold, and so on at Router A.

- Case 2, LSP transmission error, which is typically caused by IS-IS adjacency failure, e.g., link down/BFD down/authentication failure.

- Case 3, the LSP is received but not correctly processed. The problem that happens at Router B can be faulty route import policy, or Router B being in Overload mode, or the hardware/software bugs.
With sufficient ISIS PDU related statistics and parsed PDU information recorded at the device, the neighborship failure in Case 2 can be typically diagnosed at Router A or Router B independently. With such diagnosing information collected (e.g., in the format of reason code) in real-time, the server can identify the root synchronization issue with much less time and labor consumption compared with conventional methods. In Case 1 & 3, the failure is mostly caused by incorrect route policy and software/hardware issue. By comparing the LSDB with the sent/received LSP, differences can be recognized. Then the difference may further guide the localization of the root cause. Thus, by collecting the LSDBs and sent/received LSPs from the two affected neighbors, the server can have more insights at the synchronization failure.

4. Message Format

4.1. Protocol Selection Options

Regarding the network monitoring data export, BMP has been a good option. First of all, BMP serves similar purposes of network monitoring for IGP that reports routes, route statistics and peer status. In addition, BMP has already been implemented in major vendor devices and utilized by operator.

4.2. Message Types

The variety of IS-IS troubleshooting use cases requires a systematic information report of network monitoring, so that the NM server or any third party analyzer could efficiently utilize the reported messages to localize and recover various network issues. We define NM messages for IS-IS uses the following types:

- **Initiation Message**: A message used for the monitored device to inform the NM monitoring station of its capabilities, vendor, software version and so on. For example, the link MTU can be included within the message. The initiation message is sent once the TCP connection between the monitoring station and monitored router is set up. During the monitoring session, any change of the initiation message could trigger an Initiation Message update.

- **Adjacency Status Change Notification Message**: A message used to inform the NM monitoring station of the adjacency status change of the monitored device, i.e., from up to down, from down/initiation to up, with possible alarms/logs recorded in the device. This message notifies the NM server of the ongoing IS-IS adjacency change event and possible reasons. If no reason is provided or the provided reason is not specific enough, the NM server can further analyze the IS-IS PDU or the IS-IS statistics.
- Statistic Report Message: A message used to report the statistics of the ongoing IS-IS process at the monitored device. For example, abnormal LSP count of the monitored device can be a sign of route flapping. This message can be sent periodically or event triggered. If sent periodically, the frequency can be configured by the operator depending on the monitoring requirement. If it’s event triggered, it could be triggered by a counter/timer exceeding the threshold.

- IS-IS PDU Monitoring Message: A message used to update the NM server of any PDU sent from and received at the monitored device. For example, the IIHs collected from two neighbors can be used for analyzing the adjacency set up failure issue. The LSPs collected from two neighbors can be analyzed for the LSP synchronization issue.

- Termination Message: A message for the monitored router to inform the monitoring station of why it is closing the NM session. This message is sent when the monitoring session is to be closed.

4.3. Message Format

4.3.1. Common Header

The common header is encapsulated in all messages of network monitoring for IGP. It includes the Version, Message Length and Message Type fields. The common header can reuse the common header of BMP and new message types should defined for IGP monitoring.

- Type = TBD: Adjacency Status Change Notification
- Type = TBD: ISIS Statistic Report
- Type = TBD: IS-IS PDU Monitoring

4.3.2. Per Adjacency Header

Except the Initiation and Termination Message, all the rest messages are per adjacency based. Thus, a per adjacency header is defined as follows.
4.3.3. Initiation Message

Three new types of Router Capability TLVs should be defined for IGP monitoring:

- **Type = TBD: Local System ID.** The corresponding Router Capability Value field SHALL indicate the router’s System ID
- **Type = TBD: Link MTU.** The corresponding Router Capability Value field SHALL indicate the router’s link MTU.

4.3.4. Adjacency Status Change Notification

The Adjacency Status Change Notification Message indicates an IS-IS adjacency status change: from up to down or from initiation/down to up. It consists of the Common Header, Per Adjacency Header and the Reason TLV. The Notification is triggered whenever the status...
changes. The Reason TLV is optional, and is defined as follows. More Reason types can be defined if necessary.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-------------------------------+
|    Reserved |S|   Reason Type |        Reason Length          |
+-------------------------------+-------------------------------+
+                        Reason Value (variable)                +
```

- **Reason Flags (1 byte):** The S flag (1 bit) indicates if the Adjacency status is from up to down (set to 0) or from down/initial to up (set to 1). The rest bits of the Flag field are reserved. When the S flag is set to 1, the Reason Type SHALL be set to all zeroes (i.e., Type 0), the Reason Length fields SHALL be set to all zeroes, and the Reason Value field SHALL be set empty.

- **Reason Type (1 byte):** indicates the possible reason that caused the adjacency status change. Currently defined types are:
  - **Type = 0:** Adjacency Up. This type indicates the establishment of an adjacency. For this reason type, the S flag MUST be set to 1, indicating it’s a adjacency-up event. There’s no further reason to be provided. The reason Length field SHALL be set to all zeroes, and the Reason Value field SHALL be set empty.
  - **Type = 1:** Circuit Down. For this data type, the S flag MUST be set to 0, indicating it’s a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.
  - **Type = 2:** Memory Low. For this data type, the S flag MUST be set to 0, indicating it’s a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.
  - **Type = 3:** Hold timer expired. For this data type, the S flag MUST be set to 0, indicating it’s a adjacency-down event. The length field is set to all zeroes, and the value field is set empty.
  - **Type = 4:** String. For this data type, the S flag MUST be set to 0, indicating it’s a adjacency-down event. The corresponding Reason Value field indicates the reason specified by the monitored router in a free-form UTF-8 string whose length is given by the Reason Length field.
4.3.5. ISIS Statistic Report Message

The ISIS Statistic Report Message reports the statistics of the parameters that are of interest to the operator. The message consists of the Common Header, the Per Adjacency Header and the Statistic TLV. The message include both per-adjacency based statistics and non per-adjacency based statistics. For example, the received/sent LSP counts are per-adjacency based statistics, and the local LSP change times count and the number of established adjacencies are non per-adjacency based statistics. For the non per-adjacency based statistics, the CT Flag (2 bits) in the Per Adjacency Header MUST be set to 00. Upon receiving any message with CT flag set to 00, the Per Adjacency Header SHALL be ignored (the total length of the Per Adjacency Header is 18 bytes as defined in Section 3.2.2, and the message reading/analysis SHALL resume from the Statistic TLV part.

The Statistic TLV is defined as follows.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|   Reserved  |T| Statistic Type|        Statistic Length       |
+---------------------------------------------------------------+
|                       Statistic  Value                        |
+---------------------------------------------------------------+
```

* **Statistic Flags (1 byte):** provides information for the reported statistics.
  
  * **T flag (1 bit):** indicates if the statistic is for the received-from direction (set to 1) or sent-to direction the neighbor (set to 0)

* **Statistic Type (1 byte):** specifies the statistic type of the counter. Currently defined types are:

  * **Type = 0:** IIH count. The T flag indicates if it’s a sent or received Hello PDU. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.
* Type = 1: Incorrect IIH received count. For this type, the T flag MUST be set to 1. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 2: LSP count. The T flag indicates if it’s a sent or received LSP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 3: Incorrect LSP received count. For this type, the T flag MUST be set to 1. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 4: Retransmitted LSP count. For this type, the T flag MUST be set to 0. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 5: CSNP count. The T flag indicates if it’s a sent or received CSNP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 6: PSNP count. The T flag indicates if it’s a sent or received PSNP. It is a per-adjacency based statistic type, and the CT flag in the Per Adjacency Header MUST NOT be set to 00.

* Type = 7: Number of established adjacencies. It’s a non per-adjacency based statistic type, and thus for the monitoring station to recognize this type, the CT flag in the Per Adjacency Header MUST be set to 00.

* Type = 8: LSP change time count. It’s a non per-adjacency based statistic type, and thus for the monitoring station to recognize this type, the CT flag in the Per Adjacency Header MUST be set to 00.

- Statistic Length (2 bytes): indicates the length of the Statistic Value field.

- Statistic Value (4 bytes): specifies the counter value, which is a non-negative integer.

4.3.6. IS-IS PDU Monitoring Message

The IS-IS PDU Monitoring Message is used to update the monitoring station of any PDU sent from and received at the monitored device per neighbor. Following the Common Header and the Per Adjacency Header
is the IS-IS PDU. To tell whether it’s a sent or received PDU, the monitoring station can analyze the source and destination addresses in the reported PDUs.

4.3.7. Termination Message

This document does not change the Termination Message defined by RFC7854.

5. IANA

TBD

6. Contributors

TBD

7. Acknowledgments

TBD

8. 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.chen-npm-use-cases]

[I-D.ietf-netconf-yang-push]

[I-D.openconfig-rtgwg-gnmi-spec]
Yunan Gu
Huawei
156 Beiqing Road
Beijing, 100095
China
Email: guyunan@huawei.com

Shuanglong Chen
Huawei
156 Beiqing Road
Beijing, 100095
China
Email: chenshuanglong@huawei.com

Yingzhen Qu
Futurewei
United States
Email: yingzhen.qu@futurewei.com

Huanan Chen
China Telecom
109 West Zhongshan Ave
Guangzhou
China
Email: chenhuanan@189.com

Zhenbin Li
Huawei
156 Beiqing Road
Beijing, 100095
China
Email: lizhenbin@huawei.com
A YANG Network Data Model for Layer 2 VPNs
draft-ietf-opsawg-l2nm-19

Abstract

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

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

Editorial Note (To be removed by RFC Editor)

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

* "This version of this YANG module is part of RFC XXXX;"
* "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs";
* reference: RFC XXXX

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Table of Contents

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

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

Also, this document defines the initial versions of two IANA-maintained modules that define a set of identities of BGP Layer 2 encapsulation types (Section 8.1) and pseudowire types (Section 8.2). These types are used in the L2NM to identify a Layer 2 encapsulation type as a function of the signalling option used to deliver an L2VPN service. Relying upon these IANA-maintained modules is meant to provide more flexibility in handling new types rather than being limited by a set of identities defined in the L2NM itself.

Section 8.3 defines another YANG module to manage Ethernet Segments (ESes) that are required for instantiating Ethernet VPNs (EVPNs). References to Ethernet segments that are created using the module in Section 8.3 can be included in the L2NM for EVPNs.
The L2NM (Section 8.4) can be exposed, for example, by a network controller to a service controller within the service provider’s network. In particular, the model can be used in the communication interface between the entity that interacts directly with the customer (i.e., the service orchestrator) and the entity in charge of network orchestration and control (a.k.a., network controller/orchestrator) by allowing for more network-centric information to be included.

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

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

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

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

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

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

2. Terminology

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

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

The meanings of the symbols in YANG tree diagrams is defined in [RFC8340].
This document makes use of the following terms:

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

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

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

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

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

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

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

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

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

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

3. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document:

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

4. Reference Architecture

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

Similar to Section 3 of [RFC8466], CE to PE attachment is achieved through a bearer with a Layer 2 connection on top. The bearer refers to properties of the attachment that are below Layer 2, while the connection refers to Layer 2 protocol-oriented properties.
The reader may refer to [RFC8309] for the distinction between the "Customer Service Model", the "Service Delivery Model", the "Network Configuration Model", and the "Device Configuration Model". The "Domain Orchestration" and "Config Manager" roles may be performed by "SDN Controllers".
Figure 1: L2SM and L2NM Interaction

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

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

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

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

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

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

L2SM: The L2NM is not a customer service model.

The internal view of the service (i.e., the L2NM) may be mapped to an external view which is visible to customers: L2VPN Service Model (L2SM) [RFC8466].

The L2NM can be fed with inputs that are requested by customers, typically, relying upon an L2SM template. Concretely, some parts of the L2SM module can be directly mapped into the L2NM while other parts are generated as a function of the requested service and local guidelines. Finally, there are parts local to the service provider and do not map directly to the L2SM.

Note that using the L2NM within a service provider does not assume, nor does it preclude, exposing the VPN service via the L2SM. This is deployment specific. Nevertheless, the design of L2NM tries to align as much as possible with the features supported by the L2SM to ease the grafting of both the L2NM and the L2SM for the sake of highly automated VPN service provisioning and delivery.

Network Topology Modules: An L2VPN involves nodes that are part of a
topology managed by the service provider network. Such a topology can be represented using the network topology module in [RFC8345] or its extension, such as a network YANG module for Service Attachment Points (SAPs) [I-D.ietf-opsawg-sap].

Device Modules: The L2NM is not a device model.

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

* Interfaces [RFC8343].
* BGP [I-D.ietf-idr-bgp-model].
* MPLS [RFC8960].
* Access Control Lists (ACLs) [RFC8519].

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

6. Description of the Ethernet Segment YANG Module

The ‘ietf-ethernet-segment’ module (Figure 3) is used to manage a set of Ethernet segments in the context of an EVPN service.
module: ietf-ethernet-segment
  +--rw ethernet-segments
  +--rw ethernet-segment* [name]
    +--rw name string
    +--rw esi-type? identityref
    +--rw (esi-choice)?
      |  +--:(directly-assigned)
      |  |  +--rw ethernet-segment-identifier? yang:hex-string
      +--:(auto-assigned)
        +--rw esi-auto
          +--rw (auto-mode)?
            |  +--:(from-pool)
            |     +--rw esi-pool-name? string
            |     +--:(full-auto)
            +--ro auto? empty
            +--ro auto-ethernet-segment-identifier? yang:hex-string
    +--rw esi-redundancy-mode? identityref
    +--rw df-election
      +--rw df-election-method? identityref
      +--rw revertive? boolean
      +--rw election-wait-time? uint32
    +--rw split-horizon-filtering? boolean
    +--rw pbb
      |  +--rw backbone-src-mac? yang:mac-address
    +--rw member* [ne-id interface-id]
      +--rw ne-id string
      +--rw interface-id string

Figure 3: Ethernet Segments Tree Structure

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

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

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

'esi-type': Indicates the Ethernet Segment Identifier (ESI) type as discussed in Section 5 of [RFC7432]. ESIs can be automatically assigned either with or without indicating a pool from which an ESI should be taken ('esi-pool-name'). The following types are supported:
'esi-type-0-operator': The ESI is directly configured by the VPN service provider. The configured value is provided in 'ethernet-segment-identifier'.

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

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

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

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

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

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

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

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

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

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

'split-horizon-filtering': Controls the activation of the split-
horizon filtering for an ES (Section 8.3 of [RFC7432]).

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

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

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

7. Description of the L2NM YANG Module

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

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

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

7.1. Overall Structure of the Module

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

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

The 'vpn-services' container maintains the set of L2VPN services managed in the service provider network. The module allows creating a new L2VPN service by adding a new instance of 'vpn-service'. The 'vpn-service' is the data structure that abstracts the VPN service (Section 7.3).
module: ietf-l2vpn-ntw
   +--rw l2vpn-ntw
      +--rw vpn-profiles
         |  ...
         +--rw vpn-services
            +--rw vpn-service* [vpn-id]
                ...
                +--rw vpn-nodes
                   +--rw vpn-node* [vpn-node-id]
                       ...
                   +--rw vpn-network-accesses
                      +--rw vpn-network-access* [id]
                          ...

Figure 4: Overall L2NM Tree Structure

7.2. VPN Profiles

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

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

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

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

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

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

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

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

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

7.3. VPN Services

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

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

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

‘vpn-description’: Includes a textual description of the service.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

'12vpn-bgp': The service is a Multipoint VPLS that uses a BGP control plane as described in [RFC4761] and [RFC6624].

'evpn-bgp': The service is a Multipoint VPLS that uses also a
BGP control plane, but also includes the additional EVPN features and related parameters [RFC7432] and [RFC7209].

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

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

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

<table>
<thead>
<tr>
<th>VPN Type</th>
<th>Signaling Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpls</td>
<td>l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)</td>
</tr>
<tr>
<td>vpws</td>
<td>l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)</td>
</tr>
<tr>
<td>vpws-evpn</td>
<td>bgp-signaling (evpn-bgp)</td>
</tr>
<tr>
<td>pbb-evpn</td>
<td>bgp-signaling (evpn-bgp)</td>
</tr>
<tr>
<td>mpls-evpn</td>
<td>bgp-signaling (evpn-bgp)</td>
</tr>
<tr>
<td>vxlan-evpn</td>
<td>bgp-signaling (evpn-bgp)</td>
</tr>
</tbody>
</table>

Table 1: Signaling Options per VPN Service Type

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

More details are provided in Section 7.4.

'underlay-transport': Describes the preference for the transport technology to carry the traffic of the VPN service. This preference is especially useful in networks with multiple domains and Network-to-Network Interface (NNI) types. The underlay transport can be expressed as an abstract transport instance (e.g., an identifier of a VPN+ instance, a virtual network identifier, or a network slice name) or as an ordered list of the actual protocols to be enabled in the network.
A rich set of protocol identifiers that can be used to refer to an underlay transport (or how such an underlay is set up) are defined in [RFC9181].

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

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

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

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

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

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

More details are provided in Section 7.5.

7.4. Global Parameters Profiles

The ‘global-parameters-profile’ defines reusable parameters for the same L2VPN service instance (‘vpn-service’). Global parameters profiles are defined at the VPN service level, activated at the VPN node level, and then an activated VPN profile may be used at the VPN network access level. Each VPN instance profile is identified by ‘profile-id’. Some of the data nodes can be adjusted at the VPN node or VPN network access levels. These adjusted values take precedence
over the global values. The subtree of 'global-parameters-profile' is depicted in Figure 7.

---rw vpn-services
   +--rw vpn-service* [vpn-id]

...
The description of the global parameters profile is as follows:

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

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

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

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

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

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

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

Figure 7: Global Parameters Profiles Subtree
IDs (CE-VLAN IDs) from ingress to egress, i.e., CE-VLAN tag of the egress frame are identical to those of the ingress frame that yielded this egress service frame. If all-to-one bundling within a site is enabled, then preservation applies to all ingress service frames. If all-to-one bundling is disabled, then preservation applies to tagged Ingress service frames having CE-VLAN ID 1 through 4094.

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

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

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

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

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

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

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

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

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

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

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

'protection-type': It defines the loop prevention type (e.g., shut).
‘multicast’: Controls whether multicast is allowed in the service.

7.5. VPN Nodes

The ‘vpn-node’ (Figure 8) is an abstraction that represents a set of policies/configurations applied to a network node and that belong to a single ‘vpn-service’. A ‘vpn-node’ contains ‘vpn-network-accesses’, which are the interfaces involved in the creation of the VPN. The customer sites are connected to the ‘vpn-network-accesses’.
The descriptions of VPN node data nodes are as follows:
'vpn-node-id': Used to uniquely identify a node that enables a VPN network access.

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

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

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

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

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

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

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

'signaling-option': See Section 7.5.2.

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

Note that, unlike the L2SM, the L2NM does not need to model the customer site -- only the points that receive traffic from the site are covered (i.e., the PE side of Provider Edge to Customer Edge (PE-CE) connections). Hence, the VPN network access contains the connectivity information between the provider's network and the customer premises. The VPN profiles ('vpn-profiles') have a set of routing policies that can be applied during the service creation.
See Section 7.6 for more details.

7.5.1. BGP Auto-Discovery

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

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

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

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

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

7.5.2. Signaling Options

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

The following signaling data nodes are supported:

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

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

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

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

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

'l2tp': The model supports the configuration of the parameters that are discussed in Section 4 of [RFC4667]. Refer to Section 7.5.2.3 for more details.
Note that LDP and L2TP choices are bundled ("ldp-or-l2tp") because they share a set of common parameters that are further detailed in Sections 7.5.2.2 and 7.5.2.3.

7.5.2.1. BGP

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

```
... +--rw (signaling-option)?
    ...
    +--:(bgp)
        +--rw (bgp-type)?
            +--:(l2vpn-bgp)
                +--rw ce-range?     uint16
                +--rw pw-encapsulation-type?
                    identityref
                +--rw vpls-instance
                    +--rw vpls-edge-id?     uint16
                    +--rw vpls-edge-id-range?   uint16
            +--:(evpn-bgp)
                +--rw evpn-type?     leafref
                +--rw service-interface-type?
                    identityref
                +--rw evpn-policies
                    +--rw mac-learning-mode?
                        identityref
                    +--rw ingress-replication?
                        boolean
                    +--rw p2mp-replication?
                        boolean
                    +--rw arp-proxy {vpn-common:ipv4}?
                        +--rw enable?     boolean
                    +--rw ip-mobility-threshold?
                        uint16
                    +--rw duplicate-ip-detection-interval?
                        uint16
                +--rw nd-proxy {vpn-common:ipv6}?
                    +--rw enable?     boolean
                    +--rw nd-suppression?
                        boolean
                    +--rw ip-mobility-threshold?
                        uint16
                    +--rw duplicate-ip-detection-interval?
                        uint16
                +--rw underlay-multicast?
```
Remote CEs that are entitled to connect to the same VPN should fit with the CE range ('ce-range') as discussed in Section 2.2.3 of [RFC6624]. 'pw-encapsulation-type' is used to control the pseudowire encapsulation type (Section 3 of [RFC6624]). The value of the 'pw-encapsulation-type' are taken from the IANA-maintained "iana-bgp-l2-encaps" module (Section 8.1).

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

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

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

```
---rw (signaling-option)?
  ...  
  +-rw (bgp)
  |  ...
  |  +-rw (ldp-or-l2tp)
  |     +-rw ldp-or-l2tp
  |     |     +-rw agi?
  |     |     |     rt-types:route-distinguisher
  |     |     +-rw saii?    uint32
  |     |     +-rw remote-targets* [taii]
  |     |     |     +-rw taii  uint32
  |     |     |     +-rw peer-addr  inet:ip-address
  |     |     +-rw (ldp-or-l2tp)?
  |     |     +-rw (ldp)
  |     |     |     +-rw t-ldp-pw-type?
  |     |     |     |     identityref
  |     |     |     +-rw pw-type?  identityref
  |     |     |     +-rw pw-description?  string
  |     |     |     +-rw mac-addr-withdraw?  boolean
  |     |     |     +-rw pw-peer-list*  [peer-addr vc-id]
  |     |     |     |     +-rw peer-addr  inet:ip-address
  |     |     |     |     +-rw vc-id  string
  |     |     |     |     +-rw pw-priority?  uint32
  |     |     |     +-rw qinq
  |     |     |     |     +-rw s-tag  dot1q-types:vlanid
  |     |     |     |     +-rw c-tag  dot1q-types:vlanid
  |     |     +-rw (l2tp)
  |     |     ...  
```

Figure 12: Signaling Option Subtree (LDP)
7.5.2.3. L2TP

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

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

Figure 13: Signaling Option Subtree (L2TP)

7.6. VPN Network Accesses

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

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

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

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

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

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

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

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

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

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

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

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

7.6.1. Connection

The 'connection' container (Figure 15) is used to configure the relevant properties of the interface to which the L2VPN instance is attached to (e.g., encapsulation type, Link Aggregation Group (LAG) interfaces, split-horizon). The L2NM supports tag manipulation operations (e.g., tag rewrite).
Note that the ‘connection’ container does not include the physical-specific configuration as this is assumed to be directly handled using device modules (e.g., interfaces module). Moreover, this design is also meant to avoid manipulated global parameters at the service level and lower the risk of impacting other services sharing the same physical interface.

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

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

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

```plaintext
... +--rw vpn-nodes
    +--rw vpn-node* [vpn-node-id]

... +--rw vpn-network-accesses
    +--rw vpn-network-access* [id]

... +--rw connection
    |  +--rw l2-termination-point? string
    |  +--rw local-bridge-reference? string
    |  +--rw bearer-reference? string
    |     {vpn-common:bearer-reference}?
    +--rw encapsulation
        +--rw encap-type? identityref
        +--rw dot1q
            +--rw tag-type? identityref
            +--rw cvlan-id? dot1q-types:vlanid
            +--rw tag-operations
                +--rw (op-choice)?
                    +--:(pop)
                        +--rw pop? empty
                    +--:(push)
                        +--rw push? empty
                    +--:(translate)
                        +--rw translate? empty
                +--rw tag-1?
```
dot1q-types:vlanid
+-rw tag-1-type?
   dot1q-types:dot1q-tag-type
+-rw tag-2?
   dot1q-types:vlanid
+-rw tag-2-type?
   dot1q-types:dot1q-tag-type
+-rw priority-tagged
   +-rw tag-type? identityref
+-rw qinq
   +-rw tag-type? identityref
   +-rw svlan-id
      dot1q-types:vlanid
   +-rw cvlan-id
      dot1q-types:vlanid
+-rw tag-operations
   +-rw (op-choice)?
      +-:(pop)
         +-rw pop? uint8
      +-:(push)
         +-rw push? empty
      +-:(translate)
         +-rw translate? empty
   +-rw tag-1?
      dot1q-types:vlanid
   +-rw tag-1-type?
      dot1q-types:dot1q-tag-type
   +-rw tag-2?
      dot1q-types:vlanid
   +-rw tag-2-type?
      dot1q-types:dot1q-tag-type
+-rw lag-interface
   (vpn-common:lag-interface)?
   +-rw lag-interface-id? string
+-rw lacp
   +-rw lacp-state? boolean
   +-rw mode? identityref
   +-rw speed? uint32
   +-rw mini-link-num? uint32
   +-rw system-id?
      yang:mac-address
   +-rw admin-key? uint16
   +-rw system-priority? uint16
   +-rw member-link-list
      +-rw member-link* [name]
         +-rw name string
         +-rw speed? uint32
         +-rw mode? identityref
7.6.2. EVPN-VPWS Service Instance

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

An example to illustrate the use of the L2NM to configure VPWS-EVPN instances is provided in Appendix A.4.
Figure 16: EVPN-VPWS Service Instance Subtree
7.6.3. Ethernet OAM

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

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

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

The ‘service’ container (Figure 18) provides a set of service-specific configuration such as Quality of Service (QoS).
The description of the service data nodes is as follows:

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

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

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

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

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth' can be represented using the Committed Information Rate (CIR), the Excess Information Rate (EIR), or the Peak Information Rate (PIR).
As shown in Figure 19, the structure of service bandwidth data nodes is inherited from the L2SM [RFC8466]. The following types, defined in [RFC9181], can be used to indicate the bandwidth type:

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

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

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

'bw-per-service': The bandwidth is per L2VPN service.
Figure 19: Service Bandwidth Subtree
'qos': Is used to define a set of QoS policies to apply on a given VPN network access (Figure 20). The QoS classification can be based on many criteria such as source MAC address, destination MAC address, etc. See also Section 5.10.2.1 of [RFC8466] for more discussion of QoS classification including the use of color types.

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

Figure 20: QoS Subtree

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

A data frame that matches an ACL can be dropped, flooded, or trigger an alarm. A rate-limit policy can be defined for handling frames that match an ACL entry with 'flood' action.
When 'mac-loop-prevention' or 'mac-addr-limit' data nodes are provided, they take precedence over the ones included in the 'global-parameters-profile' at the VPN service or VPN node levels.

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

Figure 21: MAC Policies Subtree

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

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

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

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

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

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

  organization "IANA";
  contact "Internet Assigned Numbers Authority

  Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA  90094-2536
      United States of America
  Tel:    +1 310 301 5000
            <mailto:iana@iana.org>
  description "This module contains a collection of IANA-maintained YANG
data types that are used for referring to BGP Layer 2
encapsulation types.

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

  Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Revised BSD License
set forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents

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

Internet-Draft                    L2NM                         June 2022

the RFC itself for full legal notices.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

identity t1 {
    base bgp-l2-encaps-type;
description
    "Structure-agnostic T1 (DS1) over packet.";
reference
"RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
over Packet (SAtoP)";
}

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

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

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

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

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

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

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

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

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

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

<CODE ENDS>
8.2. IANA-Maintained Module for Pseudowire Types

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

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

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

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

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

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

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

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

Boucadair, et al. Expires 4 December 2022 [Page 54]
identity iana-pw-types {
    description
    "Base Pseudowire Layer 2 encapsulation type.";
}

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

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

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

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

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

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

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

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

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

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

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

identity atm-aal5-vcc {
    base iana-pw-types;
    description
        "ATM AAL5 PDU VCC transport.";
    reference
        "RFC 4717: Encapsulation Methods for Transport of
        Asynchronous Transfer Mode (ATM) over MPLS
        Networks";
}
identity fr-port-mode {
    base iana-pw-types;
    description
        "Frame-Relay Port mode.";
    reference
        "RFC 4619: Encapsulation Methods for Transport of Frame Relay
        over Multiprotocol Label Switching (MPLS) Networks";
}

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

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

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

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

identity t3 {
identity ces-over-psn {
  base iana-pw-types;
  description
    "CESoPSN basic mode.";
  reference
    "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
     Circuit Emulation Service over Packet Switched
     Network (CESoPSN)";
}

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

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

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

identity dlc1 {
  base iana-pw-types;
  description
    "Frame Relay DLCI.";
}
reference
  "RFC 4619: Encapsulation Methods for Transport of Frame Relay over Multiprotocol Label Switching (MPLS) Networks";

)

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

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

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

identity crtp {
  base iana-pw-types;
  description
    "cRTP Transport Header-compressed Packets.";
  reference
    "RFC 2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial Links
    RFC 4901: Protocol Extensions for Header Compression over MPLS";
}
identity atm-vp-virtual-trunk {
    baseiana-pw-types;
    description
        "ATM VP Virtual Trunk.";
    reference
        "MFA Forum: The Use of Virtual Trunks for ATM/MPLS
         Control Plane Interworking Specification";
}

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

identity wildcard {
    baseiana-pw-types;
    description
        "Wildcard.";
    reference
        "RFC 4863: Wildcard Pseudowire Type";
}
}

8.3. Ethernet Segments

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

<CODE BEGINS>
file "ietf-ethernet-segment@2022-05-25.yang"
module ietf-ethernet-segment {
    yang-version 1.1;
    prefix l2vpn-es;

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

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

contact
"WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
WG List: <mailto:opsawg@ietf.org>
Editor:  Mohamed Boucadair
<mailto:mohamed.boucadair@orange.com>
Editor:  Samier Barguil
<mailto:samier.barguil@telefonica.com>
Author:  Oscar Gonzalez de Dios
<mailto:oscar.gonzalez@telefonica.com>");
description
"This YANG module defines a model for Ethernet Segments.

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

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Revised BSD License
set forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents

This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.");
revision 2022-05-25 {
  description
    "Initial version."
  reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs."
}
/* Typedefs */
typedef es-ref {
  type leafref {
    path "/l2vpn-es:ethernet-segments/l2vpn-es:ethernet-segment"
    + "/l2vpn-es:name";
  }
  description
    "Defines a type for referencing an Ethernet segment in
     other modules."
}
/* Identities */
identity esi-type {

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

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

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

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

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

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

identity esi-type-5-asn {

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

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

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

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

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

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

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

/* Main Ethernet Segment Container */

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

For ESI Type 1: The auto-mode must be used.
For ESI Type 2: The auto-mode must be used.
For ESI Type 3: The directly-assigned or auto-mode must be used.
For ESI Type 4: The directly-assigned or auto-mode must be used.
For ESI Type 5: The directly-assigned or auto-mode must be used.

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

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

leaf auto-ethernet-segment-identifier {
  type yang:hex-string {
    length "29"
  }
  config false;
  description
    "The value of the auto-assigned ESI."
}
}

leaf esi-redundancy-mode {
  type identityref {
    base es-redundancy-mode;
  }
  description
    "Indicates the ES redundancy mode."
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1"
}
container df-election {
  description
    "Top container for the DF election method properties."
  leaf df-election-method {
    type identityref {
      base df-election-methods;
    }
    default "default-7432";
    description
      "Specifies the DF election method."
    reference
      "RFC 8584: Framework for Ethernet VPN Designated Forwarder Election Extensibility"
  }
  leaf revertive {
    when "derived-from-or-self(../df-election-method, "
      + "'preference')" {
      description
        "The revertive value is only applicable to the preference method."
    }
    type boolean;
    default "true";
    description
      "The default behavior is that the DF election
procedure is triggered upon PE failures following configured preference values. Such a mode is called the revertive mode. This mode may not be suitable in some scenarios where, e.g., an operator may want to maintain the new DF even if the former DF recovers. Such a mode is called the 'non-revertive' mode.

The non-revertive mode can be configured by setting 'revertive' leaf to 'false'."

leaf split-horizon-filtering {
  type boolean;
  description
  "Controls split-horizon filtering. It is enabled when set to 'true'."

  In order to achieve split-horizon filtering, every Broadcast, unknown unicast, or multicast (BUM) packet originating from a non-DF PE is encapsulated with an MPLS label that identifies the origin ES.";
  reference
  "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3";
}

container pbb {
  description
  "Provider Backbone Bridging (PBB) parameters .";
  reference
  "IEEE 802.1ah: Provider Backbone Bridge";

  leaf backbone-src-mac {
    type yang:mac-address;
    description
    "The PEs connected to the same CE must share the same Provider Backbone (B-MAC) address in"
All-Active mode.

8.4. L2NM

The "ietf-l2vpn-ntw" YANG module uses types defined in [RFC6991], [RFC9181], [RFC8294], and [IEEE802.1Qcp-2018].
import ietf-vpn-common {
    prefix vpn-common;
    reference
    "RFC 9181: A Common YANG for Data Model for Layer 2 and Layer 3 VPNs";
}
import iana-bgp-l2-encaps {
    prefix iana-bgp-l2-encaps;
    reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
}
import iana-pseudowire-types {
    prefix iana-pw-types;
    reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
}
import ietf-ethernet-segment {
    prefix l2vpn-es;
    reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
}
import ietf-routing-types {
    prefix rt-types;
    reference
    "RFC 8294: Common YANG Data Types for the Routing Area";
}
import ieee802-dot1q-types {
    prefix dot1q-types;
    reference
    "IEEE Std 802.1Qcp-2018: Bridges and Bridged Networks – Amendment: YANG Data Model";
}

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

description
    "This YANG module defines a network model for Layer 2 VPN
services.

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

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

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

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

/*@ Features */

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

/*@ Identities */

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

identity vlan-based-service-interface {
  base evpn-service-interface-type;
  description
    "VLAN-Based Service Interface.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.1";
}
identity vlan-bundle-service-interface {
    base evpn-service-interface-type;
    description
        "VLAN Bundle Service Interface.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.2";
}

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

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

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

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

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

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

identity green {
    base color-type;
    description

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

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

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

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

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

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

identity hvpls {
  base t-ldp-pw-type;
  description
  "Identity for Hierarchical Virtual Private LAN Service (H-VPLS)
identity t-ldp-pw-type {
  description "t-ldp-pw-type.";
  reference
    "RFC 4762: Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling, Section 10";
}

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

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

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

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

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

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

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

identity delay {
  base pm-type;
  description "Delay measurement is the performance monitoring type.";
}
identity mac-learning-mode {
  description
    "Media Access Control (MAC) learning mode."
}

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

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

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

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

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

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

identity precedence-type {
  description
    "Redundancy type. The service can be created
    with primary and secondary signalization."
}
identity primary {
    base precedence-type;
    description
    "Identifies the main VPN network access.";
}

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

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

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

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

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

/* Groupings */
grouping cfm-802 {
description "Grouping for 802.1ag Connectivity Fault Management (CFM) attributes.";
reference "IEEE Std 802-1ag: Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management";

leaf maid {
  type string;
  description "Maintenance Association Identifier (MAID).";
}

leaf mep-id {
  type uint32;
  description "Local Maintenance Entity Group End Point (MEP) ID.";
}

leaf mep-level {
  type uint32;
  description "MEP level.";
}

leaf mep-up-down {
  type enumeration {
    enum up {
      description "MEP is up.";
    }
    enum down {
      description "MEP is down.";
    }
  }
  default "up";
  description "MEP up/down.";
}

leaf remote-mep-id {
  type uint32;
  description "Remote MEP ID.";
}

leaf cos-for-cfm-pdus {
  type uint32;
  description "Class of service for CFM PDUs.";
}

leaf ccm-interval {
  type uint32;
}
units "milliseconds";
default "10000";
description
   "Continuity Check Message (CCM) interval.";
}
leaf ccm-holdtime {
    type uint32;
    units "milliseconds";
    default "35000";
    description
       "CCM hold time.";
}
leaf ccm-p-bits-pri {
    type ccm-priority-type;
    description
       "The priority parameter for Continuity Check Messages (CCMs)
        transmitted by the MEP.";
}
}
grouping y-1731 {
    description
       "Grouping for Y-1731";
    reference
       "ITU-T Y-1731: Operations, administration and maintenance
        (OAM) functions and mechanisms for Ethernet-based networks";
    list y-1731 {
        key "maid";
        description
           "List of configured Y-1731 instances.";
        leaf maid {
            type string;
            description
               "MAID.";
        }
        leaf mep-id {
            type uint32;
            description
               "Local MEP ID.";
        }
        leaf pm-type {
            type identityref {
                base pm-type;
            }
            default "delay";
            description
               "Performance monitor types.";
        }
    }
}

leaf remote-mep-id {
    type uint32;
    description
    "Remote MEP ID."
}
leaf message-period {
    type uint32;
    units "milliseconds";
    default "10000";
    description
    "Defines the interval between OAM messages.";
}
leaf measurement-interval {
    type uint32;
    units "seconds";
    description
    "Specifies the measurement interval for statistics.";
}
leaf cos {
    type uint32;
    description
    "Identifies the Class of Service.";
}
leaf loss-measurement {
    type boolean;
    default "false";
    description
    "Controls whether loss measurement is ('true') or
    disabled ('false').";
}
leaf synthethic-loss-measurement {
    type boolean;
    default "false";
    description
    "Indicates whether synthetic loss measurement is enabled
    ('true') or disabled ('false').";
}
container delay-measurement {
    description
    "Container for delay measurement";
    leaf enable-dm {
        type boolean;
        default "false";
        description
        "Controls whether delay measurement is enabled ('true')
        or disabled ('false').";
    }
}
leaf two-way {
    type boolean;
    default "false";
    description
        "Whether delay measurement is two-way ('true') or one-
        way ('false').";
}
leaf frame-size {
    type uint32;
    units "bytes";
    description
        "Indicates the frame size.";
}
leaf session-type {
    type enumeration {
        enum proactive {
            description
                "Proactive mode.";
        }
        enum on-demand {
            description
                "On-demand mode.";
        }
    }
    default "on-demand";
    description
        "Specifies the session type.";
}

grouping parameters-profile {
    description
        "Container for per-service parameters.";
    leaf local-autonomous-system {
        type inet:as-number;
        description
            "Indicates a local AS Number (ASN).";
    }
    leaf svc-mtu {
        type uint32;
        units "bytes";
        description
            "Layer 2 service MTU.
            It is also known as the maximum transmission
            unit or maximum frame size.";
    }
}
leaf ce-vlan-preservation {
    type boolean;
    description
    "Preserve the CE-VLAN ID from ingress to egress, i.e.,
    CE-VLAN tag of the egress frame is identical to
    that of the ingress frame that yielded this egress
    service frame. If all-to-one bundling within a site
    is enabled, then preservation applies to all ingress
    service frames. If all-to-one bundling is disabled,
    then preservation applies to tagged ingress service
    frames having CE-VLAN ID 1 through 4094.";
}
leaf ce-vlan-cos-preservation {
    type boolean;
    description
    "CE VLAN CoS preservation. Priority Code Point (PCP) bits
    in the CE-VLAN tag of the egress frame are identical to
    those of the ingress frame that yielded this egress
    service frame.";
}
leaf control-word-negotiation {
    type boolean;
    description
    "Controls whether Control-word negotiation is enabled
    (if set to true) or not (if set to false).";
    reference
    "RFC 8077: Pseudowire Setup and Maintenance
    Using the Label Distribution Protocol (LDP),
    Section 7";
}
container mac-policies {
    description
    "Container of MAC policies.";
    container mac-addr-limit {
        description
        "Container of MAC address limit configuration.";
        leaf limit-number {
            type uint16;
            description
            "Maximum number of MAC addresses learned from
            the customer for a single service instance.
            The default value is '2' when this grouping
            is used at the service level.";
        }
        leaf time-interval {
            type uint32;
            units "milliseconds";
            description
            ""
"The aging time of the mac address.
The default value is '300' when this grouping
is used at the service level."
}
leaf action {
  type identityref {
    base mac-action;
  }
  description
  "Specifies the action when the upper limit is
  exceeded: drop the packet, flood the packet,
or log a warning message (without dropping
the packet).
The default value is 'warning' when this
  grouping is used at the service level."
}
}
container mac-loop-prevention {
  description
  "Container for MAC loop prevention."
  leaf window {
    type uint32;
    units "seconds";
    description
    "The time interval over which a MAC mobility event
is detected and checked.
The default value is '180' when this grouping
is used at the service level."
  }
  leaf frequency {
    type uint32;
    description
    "The number of times to detect MAC duplication, where
a 'duplicate MAC address' situation has occurred
within the 'window' time interval and the duplicate
MAC address has been added to a list of duplicate
MAC addresses.
The default value is '5' when this grouping is
called at the service level."
  }
  leaf retry-timer {
    type uint32;
    units "seconds";
    description
    "The retry timer. When the retry timer expires,
the duplicate MAC address will be flushed from
the MAC-VRF."
  }
}
leaf protection-type {
  type identityref {
    base loop-prevention-type;
  }
  description
    "Protection type. The default value is 'trap' when this grouping is used at the service level.";
}
}

container multicast {
  if-feature "vpn-common:multicast";
  description
    "Multicast container.";
  leaf enabled {
    type boolean;
    default "false";
    description
      "Enables multicast.";
  }
  container customer-tree-flavors {
    description
      "Type of trees used by the customer.";
    leaf-list tree-flavor {
      type identityref {
        base vpn-common:multicast-tree-type;
      }
      description
        "Type of multicast tree to be used.";
    }
  }
}

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

type uint64;
units "bytes";
description
"Committed Burst Size. CBS controls the
bursty nature of the traffic. Traffic
does not use the configured CIR
accumulates credits until the credits
reach the configured CBS."
}
leaf eir {
  type uint64;
  units "bps";
  description
  "Excess Information Rate, i.e., excess
  frame delivery allowed not subject to
  SLA. The traffic rate can be limited
  by EIR."
}
leaf ebs {
  type uint64;
  units "bytes";
  description
  "Excess Burst Size. The bandwidth
  available for burst traffic from the
  EBS is subject to the amount of
  bandwidth that is accumulated during
  periods when traffic allocated by the
  EIR policy is not used."
}
leaf pir {
  type uint64;
  units "bps";
  description
  "Peak Information Rate, i.e., maximum
  frame delivery allowed. It is equal
to or less than sum of CIR and EIR."
}
leaf pbs {
  type uint64;
  units "bytes";
  description
  "Peak Burst Size."
}

/* Main L2NM Container */

container l2vpn-ntw {
description
"Container for the L2NM."
container vpn-profiles {
  description
  "Container for VPN profiles."
  uses vpn-common:vpn-profile-cfg;
}
container vpn-services {
  description
  "Container for L2VPN services."
  list vpn-service {
    key "vpn-id"
    description
    "Container of a VPN service."
    uses vpn-common:vpn-description;
    leaf parent-service-id {
      type vpn-common:vpn-id;
      description
      "Pointer to the parent service that triggered the L2NM."
    }
    leaf vpn-type {
      type identityref {
        base vpn-common:service-type;
      } must "not(derived-from-or-self(current(), " + "'vpn-common:l3vpn'))" {
        error-message "L3VPN is only applicable in L3NM.";
      }
      description
      "Service type."
    }
    leaf vpn-service-topology {
      type identityref {
        base vpn-common:vpn-topology;
      } description
      "Defining service topology, such as any-to-any, hub-spoke, etc.";
    }
    leaf bgp-ad-enabled {
      type boolean;
      description
      "Indicates whether BGP auto-discovery is enabled or disabled."
    }
    leaf signaling-type {
      type identityref {

base vpn-common:vpn-signaling-type;
}

description
  "VPN signaling type."
}

container global-parameters-profiles {
  description
    "Container for a list of global parameters profiles.";
  list global-parameters-profile {
    key "profile-id";
    description
      "List of global parameters profiles.";
    leaf profile-id {
      type string;
      description
        "The identifier of the global parameters profile.";
    }
    uses vpn-common:route-distinguisher;
    uses vpn-common:vpn-route-targets;
    uses parameters-profile;
  }
}

container underlay-transport {
  description
    "Container for the underlay transport.";
  uses vpn-common:underlay-transport;
}

uses vpn-common:service-status;

container vpn-nodes {
  description
    "Set of VPN nodes that are involved in the L2NM.";
  list vpn-node {
    key "vpn-node-id";
    description
      "Container of the VPN nodes.";
    leaf vpn-node-id {
      type vpn-common:vpn-id;
      description
        "Sets the identifier of the VPN node.";
    }
    leaf description {
      type string;
      description
        "Textual description of a VPN node.";
    }
    leaf ne-id {
      type string;
      description
        "Network element identifier of the VPN node.";
    }
  }
}

uses vpn-common:parameters-profile;

description
"An identifier of the network element where
the VPN node is deployed. This identifier
uniquely identifies the network element within
an administrative domain.";
}
leaf role {
  type identityref {
    base vpn-common:role;
  }
  default "vpn-common:any-to-any-role";
  description
  "Role of the VPN node in the VPN.";
}
leaf router-id {
  type rt-types:router-id;
  description
  "A 32-bit number in the dotted-quad format that is
  used to uniquely identify a node within an
  autonomous system (AS).";
}
container active-global-parameters-profiles {
  description
  "Container for a list of global parameters
  profiles.";
  list global-parameters-profile {
    key "profile-id";
    description
    "List of active global parameters profiles.";
    leaf profile-id {
      type leafref {
        path "../../../../../global-parameters-profiles"/
        "global-parameters-profile/profile-id";
      }
      description
      "Points to a global profile defined at the
      service level.";
    }
    uses parameters-profile;
  }
}
uses vpn-common:service-status;
container bgp-auto-discovery {
  when ".//..//bgp-ad-enabled = 'true'" {
    description
    "Only applies when BGP auto-discovery is enabled.";
  }
  description
"BGP is used for auto-discovery."
choice bgp-type {
  description
  "Choice for the BGP type.";
  case l2vpn-bgp {
    description
    "Container for BGP L2VPN."
    leaf vpn-id {
      type vpn-common:vpn-id;
      description
      "VPN Identifier. This identifier serves to
      unify components of a given VPN for the
      sake of auto-discovery."
      reference
      "RFC 6624: Layer 2 Virtual Private Networks
      Using BGP for Auto-Discovery and
      Signaling";
    }
  }
  case evpn-bgp {
    description
    "EVPN case."
    leaf evpn-type {
      type leafref {
        path "./.../.../vpn-type";
      }
      description
      "EVPN type."
    }
    leaf auto-rt-enable {
      type boolean;
      default "false";
      description
      "Enables/disabled RT auto-derivation based on
      the ASN and Ethernet Tag ID."
      reference
      "RFC 7432: BGP MPLS-Based Ethernet VPN,
      Section 7.10.1";
    }
    leaf auto-route-target {
      when ". ./auto-rt-enable = 'true'" {
        description
        "Can only be used when auto-RD is enabled."
      }
      type rt-types:route-target;
      config false;
      description
      "The value of the auto-assigned RT.";
    }
  }
uses vpn-common:route-distinguisher;
uses vpn-common:vpn-route-targets;
}
container signaling-option {

description
"Container for the L2VPN signaling."
leaf advertise-mtu {

type boolean;
description
"Controls whether MTU is advertised."
reference
"RFC 4667: Layer 2 Virtual Private Network (L2VPN) Extensions for Layer 2 Tunneling Protocol (L2TP), Section 4.3"
}
leaf mtu-allow-mismatch {

type boolean;
description
"When set to true, it allows MTU mismatch."
reference
"RFC 4667: Layer 2 Virtual Private Network (L2VPN) Extensions for Layer 2 Tunneling Protocol (L2TP), Section 4.3"
}
leaf signaling-type {

type leafref {
  path "../../../../../signaling-type";
}
description
"VPN signaling type."
}
choice signaling-option {

description
"Choice for the signaling-option."
  case bgp {
    description
    "BGP is used as the signaling protocol."
    choice bgp-type {
      description
      "Choice for the BGP type."
      case l2vpn-bgp {
        description
        "Container for BGP L2VPN."
        leaf ce-range {
          type uint16;
        }
      }
    }
  }
}
description
"Determines the number of remote CEs with
which a given CE can communicate in the
context of a VPN.";
reference
"RFC 6624: Layer 2 Virtual Private Networks
Using BGP for Auto-Discovery and
Signaling";
}
leaf pw-encapsulation-type {
    type identityref {
        base iana-bgp-l2-encaps:bgp-l2-encaps-type;
    }
    description
    "PW encapsulation type.";
}
container vpls-instance {
    when "derived-from-or-self(../../../../
        vpn-type, 'vpn-common:vpls')" {
        description
        "Only applies for VPLS.";
    }
    description
    "VPLS instance.";
    leaf vpls-edge-id {
        type uint16;
        description
        "VPLS Edge Identifier (VE ID). This is
used when the same VE ID is configured
for the PE.";
        reference
        "RFC 4761: Virtual Private LAN Service
(VPLS) Using BGP for Auto-
Discovery and Signaling,
Section 3.5";
    }
    leaf vpls-edge-id-range {
        type uint16;
        description
        "Specifies the size of the range of
VE ID in a VPLS service. The range
controls the size of the label
block advertised in the context of
a VPLS instance.";
        reference
        "RFC 4761: Virtual Private LAN Service
(VPLS) Using BGP for Auto-
Discovery and Signaling";
}
case evpn-bgp {
  description
    "Used for EVPN.";
  leaf evpn-type {
    type leafref {
      path "../../../bgp-auto-discovery/evpn-type";
    }
    description
      "EVPN type.";
  }
  leaf service-interface-type {
    type identityref {
      base evpn-service-interface-type;
    }
    description
      "EVPN service interface type.";
  }
  container evpn-policies {
    description
      "Includes a set of EVPN policies such as those related to handling MAC addresses.";
    leaf mac-learning-mode {
      type identityref {
        base mac-learning-mode;
      }
      description
        "Indicates through which plane MAC addresses are advertised.";
    }
    leaf ingress-replication {
      type boolean;
      description
        "Controls whether ingress replication is enabled ('true') or disabled ('false').";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3.1.1";
    }
    leaf p2mp-replication {
      type boolean;
      description
        "Controls whether P2MP replication is enabled ('true') or disabled ('false')";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3.1.1";
    }
  }
}
"RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3.1.2";

} container arp-proxy {
    if-feature "vpn-common:ipv4";
    description "Top container for the ARP proxy.";
    leaf enable {
        type boolean;
        default "false";
        description "Enables (when set to 'true') or disables (when set to 'false') ARP proxy.";
        reference "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 10";
    }
    leaf arp-suppression {
        type boolean;
        default "false";
        description "Enables (when set to 'true') or disables (when set to 'false') ARP suppression.";
        reference "RFC 7432: BGP MPLS-Based Ethernet VPN";
    }
    leaf ip-mobility-threshold {
        type uint16;
        description "It is possible for a given host (as defined by its IP address) to move from one ES to another. IP mobility threshold specifies the number of IP mobility events that are detected for a given IP address within the detection-threshold before it is identified as a duplicate IP address. Once the detection threshold is reached, updates for the IP address are suppressed.";
    }
    leaf duplicate-ip-detection-interval {
        type uint16;
    }
}
units "seconds";
description
"The time interval used in detecting a
duplicate IP address. Duplicate IP address detection number of host moves
are allowed within this interval period.";
}
}
container nd-proxy {
  if-feature "vpn-common:ipv6";
description
"Top container for the ND proxy.";
leaf enable {
  type boolean;
default "false";
description
"Enables (when set to 'true') or
disables (when set to 'false') ND proxy.";
reference
"RFC 7432: BGP MPLS-Based Ethernet VPN,
Section 10";
}
leaf nd-suppression {
  type boolean;
default "false";
description
"Enables (when set to 'true') or
disables (when set to 'false') Neighbor Discovery (ND) message suppression.
ND suppression is a technique that is used to reduce the amount of ND packets flooding within individual segments, that is between hosts connected to the same logical switch.";
}
leaf ip-mobility-threshold {
  type uint16;
description
"It is possible for a given host (as defined by its IP address) to move from one ES to another.
IP mobility threshold specifies the number of IP mobility events that are detected for a given IP

address within the
detection-threshold before it
is identified as a duplicate IP
address.
Once the detection threshold is
reached, updates for the IP address
are suppressed."

} leaf duplicate-ip-detection-interval {
 type uint16;
 units "seconds";
 description
 "The time interval used in detecting a
duplicate IP address. Duplicate IP
address detection number of host moves
are allowed within this interval
period.";

}

leaf underlay-multicast {
 type boolean;
 default "false";
 description
 "Enables (when set to 'true') or disables
(when set to 'false') underlay
multicast.";

}

leaf flood-unknown-unicast-supression {
 type boolean;
 default "false";
 description
 "Enables (when set to 'true') or disables
(when set to 'false') unknown flood
unicast suppression.";

}

leaf vpws-vlan-aware {
 type boolean;
 default "false";
 description
 "Enables (when set to 'true') or disables
(when set to 'false') VPWS VLAN-aware.";

} container bum-management {
 description
 "Broadcast-unknown-unicast-multicast
management.";
 leaf discard-broadcast {
 type boolean;
default "false";
description
  "Discards broadcast, when enabled.";
}
leaf discard-unknown-multicast {
  type boolean;
  default "false";
  description
    "Discards unknown multicast, when enabled.";
}
leaf discard-unknown-unicast {
  type boolean;
  default "false";
  description
    "Discards unknown unicast, when enabled.";
}
}
}
}
}
}
}
}
}
}
}
}
}
}
container pbb {
  when "derived-from-or-self(" + ".../../evpn-type, 'pbb-evpn')" { 
    description
      "Only applies for PBB EVPN.";
  }
  description
    "PBB parameters container.";
  reference
    "IEEE 802.1ah: Provider Backbone Bridge";
}
leaf backbone-src-mac {
  type yang:mac-address;
  description
    "Includes provider backbone MAC (B-MAC) address.";
  reference
    "RFC 7623: Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN), Section 8.1";
}
}
}
}
container ldp-or-l2tp {
  description
    "Container for LDP or L2TP-signaled PWs choice.";
leaf agi {
    type rt-types:route-distinguisher;
    description
        "Attachment Group Identifier. Also, called
        VPLS-Id.";
    reference
        "RFC 4667: Layer 2 Virtual Private Network
        (L2VPN) Extensions for Layer 2
        Tunneling Protocol (L2TP),
        Section 4.3
        RFC 4762: Virtual Private LAN Service (VPLS)
        Using Label Distribution Protocol
        (LDP) Signaling, Section 6.1.1";
}

leaf saii {
    type uint32;
    description
        "Source Attachment Individual Identifier
        (SAII).";
    reference
        "RFC 4667: Layer 2 Virtual Private Network
        (L2VPN) Extensions for Layer 2
        Tunneling Protocol (L2TP),
        Section 3";
}

list remote-targets {
    key "taii";
    description
        "List of allowed target Attachment Individual
        Identifier (AI1) and peers.";
    reference
        "RFC 4667: Layer 2 Virtual Private Network
        (L2VPN) Extensions for Layer 2
        Tunneling Protocol (L2TP),
        Section 5";

    leaf taii {
        type uint32;
        description
            "Target Attachment Individual Identifier.";
        reference
            "RFC 4667: Layer 2 Virtual Private Network
            (L2VPN) Extensions for Layer 2
            Tunneling Protocol (L2TP),
            Section 3";
    }

    leaf peer-addr {
        type inet:ip-address;
        description

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

"RFC 4762: Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling, Section 6.2";

```yaml
list pw-peer-list {
  key "peer-addr vc-id";
  description "List of AC and PW bindings.";
  leaf peer-addr {
    type inet:ip-address;
    description "Indicates the peer’s IP address.";
  }
  leaf vc-id {
    type string;
    description "VC label used to identify a PW.";
  }
  leaf pw-priority {
    type uint32;
    description "Defines the priority for the PW. The higher the pw-priority value, the higher the preference of the PW will be.";
  }
}

container qinq {
  when "derived-from-or-self(" + "/.t-ldp-pw-type, 'hvpls')" {
    description "Only applies when t-ldp pw type is h-vpls.";
  }
  description "Container for QinQ."
  leaf s-tag {
    type dot1q-types:vlanid;
    mandatory true;
    description "S-TAG."
  }
  leaf c-tag {
    type dot1q-types:vlanid;
    mandatory true;
    description "C-TAG."
  }
```
case l2tp {
    description "Container for L2TP PWs."
    leaf router-id {
        type rt-types:router-id;
        description "A 32-bit number in the dotted-quad format that is used to uniquely identify a node within a service provider network."
    }
    leaf pseudowire-type {
        type identityref {
            base iana-pw-types:iana-pw-types;
        }
        description "Encapsulation type."
    }
}

container vpn-network-accesses {
    description "Main container for VPN network accesses."
    list vpn-network-access {
        key "id";
        description "List of VPN network accesses."
        leaf id {
            type vpn-common:vpn-id;
            description "Identifier of the network access"
        }
        leaf description {
            type text;
            description "Description of the network access";
        }
    }
}

leaf description {
    type text;
    description "Description of the network access";
}
type string;
description
"A textual description of the VPN network access.";
}

leaf interface-id {
  type string;
description
"Refers to a physical or logical interface.";
}

leaf active-vpn-node-profile {
  type leafref {
    path "../../.." + "/active-global-parameters-profiles" + "/global-parameters-profile/profile-id";
  }
  description
  "An identifier of an active VPN instance profile.";
}

uses vpn-common:service-status;

container connection {
  description
  "Container for the bearer and AC.";
  leaf l2-termination-point {
    type string;
description
    "Specifies a reference to a local Layer 2 termination point such as a Layer 2 sub-interface.";
  }
  leaf local-bridge-reference {
    type string;
description
    "Specifies a local bridge reference to accommodate, for example, implementations that require internal bridging. A reference may be a local bridge domain.";
  }
  leaf bearer-reference {
    if-feature "vpn-common:bearer-reference";
    type string;
description
    "This is an internal reference for the service provider to identify the bearer associated with this VPN.";
  }
}

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

"Selects the tag rewriting policy for a
VPN network access."

leaf pop {
  type empty;
  description
  "Pop the outer tag.";
}

leaf push {
  type empty;
  description
  "Push one or two tags defined by the
tag-1 and tag-2 leaves. It is
assumed that, absent any policy, the
default value of 0 will be used for
PCP setting.";
}

leaf translate {
  type empty;
  description
  "Translate the outer tag to one or two
tags. PCP bits are preserved.";
}

leaf tag-1 {
  when 'not(../pop)';
  type dot1q-types:vlanid;
  description
  "A first tag to be used for push or
translate operations. This tag will be
used as the outermost tag as a result
of the tag operation.";
}

leaf tag-1-type {
  type dot1q-types:dot1q-tag-type;
  default "dot1q-types:s-vlan";
  description
  "Specifies a specific 802.1Q tag type
of tag-1.";
}

leaf tag-2 {
  when '(.*/translate)';
  type dot1q-types:vlanid;
  description
  "A second tag to be used for
translation.";
}

leaf tag-2-type {
  type dot1q-types:dot1q-tag-type;
default "dot1q-types:c-vlan";

description
   " Specifies a specific 802.1Q tag type of tag-2."

}
}

container priority-tagged {
   when "derived-from-or-self(../encap-type, 
       "+ ",
       "+'vpn-common:priority-tagged')" {
      description
        " Only applies when the type of the tagged interface is 'priority-tagged'."
   }

description
   " Priority tagged container."

leaf tag-type {
   type identityref {
      base vpn-common:tag-type;
   }
   default "vpn-common:c-vlan";
   description
      " Tag type. By default, the tag type is 'c-vlan'."
}

}

container qinq {
   when "derived-from-or-self(../encap-type, 
       "+ ",
       "+'vpn-common:qinq')" {
      description
        " Only applies when the type of the tagged interface is QinQ."
   }

description
   " Includes QinQ parameters."

leaf tag-type {
   type identityref {
      base vpn-common:tag-type;
   }
   default "vpn-common:s-c-vlan";
   description
      " Tag type. By default, the tag type is 's-c-vlan'."
}

leaf svlan-id {
   type dot1q-types:vlanid;
   mandatory true;
   description

"S-VLAN identifier.");

leaf cvlan-id {
  type dot1q-types:vlanid;
  mandatory true;
  description
    "C-VLAN identifier.";
}

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

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

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

- translate 2 with tag-1 leaf is provided: the outer tag is popped while the inner tag is translated to the value in tag-1.

```
leaf tag-1 {
    when 'not(../pop)';
    type dot1q-types:vlanid;
    description
        "A first tag to be used for push or translate operations. This tag will be used as the outermost tag as a result of the tag operation."
}

leaf tag-1-type {
    type dot1q-types:dot1q-tag-type;
    default "dot1q-types:s-vlan";
    description
        "Specifies a specific 802.1Q tag type of tag-1."
}

leaf tag-2 {
    when 'not(../pop)';
    type dot1q-types:vlanid;
    description
        "A second tag to be used for push or translate operations."
}

leaf tag-2-type {
    type dot1q-types:dot1q-tag-type;
    default "dot1q-types:c-vlan";
    description
        "Specifies a specific 802.1Q tag type of tag-2."
}
```
container lag-interface {
    if-feature "vpn-common:lag-interface";
    description
        "Container of LAG interface attributes configuration.";
    leaf lag-interface-id {
        type string;
        description
        "LAG interface identifier.";
    }
}

container lacp {
    description
        "Container for LACP.";
    leaf lacp-state {
        type boolean;
        default "false";
        description
        "Controls whether LACP is enabled.";
    }
    leaf mode {
        type identityref {
            base lacp-mode;
        }
        description
        "Indicates the LACP mode.";
    }
    leaf speed {
        type uint32;
        units "mbps";
        default "10";
        description
        "LACP speed. This low default value is inherited from the L2SM.";
    }
    leaf mini-link-num {
        type uint32;
        description
        "Defines the minimum number of links that must be active before the aggregating link is put into service.";
    }
    leaf system-id {
        type yang:mac-address;
        description
        "LAG interface identifier.";
    }
}
"Indicates the System ID used by LACP."
}
leaf admin-key {
  type uint16;
  description
      "Indicates the value of the key used for
       the aggregate interface."
}
leaf system-priority {
  type uint16 {
    range "0..65535";
  }
  default "32768";
  description
      "Indicates the LACP priority for the
       system."
}
container member-link-list {
  description
      "Container of Member link list.";
  list member-link {
    key "name";
    description
      "Member link."
    leaf name {
      type string;
      description
        "Member link name."
    }
    leaf speed {
      type uint32;
      units "mbps";
      default "10";
      description
        "Port speed."
    }
    leaf mode {
      type identityref {
        base vpn-common:neg-mode;
      }
      description
        "Negotiation mode."
    }
    leaf link-mtu {
      type uint32;
      units "bytes";
      description
        "Link MTU size."
    }
container oam-802.3ah-link {
    if-feature "oam-3ah"
    description "Container for oam 802.3ah link."
    leaf enable {
        type boolean;
        default "false"
        description "Indicates support of OAM 802.3ah link."
    }
}

leaf flow-control {
    type boolean;
    default "false"
    description "Indicates whether flow control is supported."
}

leaf llpd {
    type boolean;
    default "false"
    description "Indicates whether Link Layer Discovery Protocol (LLDP) is supported."
}

container split-horizon {
    description "Configuration with split horizon enabled."
    leaf group-name {
        type string
        description "Group name of the Split Horizon."
    }
}

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

case evpn-bgp {
  description "Used for EVPN."
  leaf df-preference {
    type uint16;
    default "32767"
    description
"Defines a 2-octet value that indicates the PE preference to become the DF in the ES.

The preference value is only applicable to the preference based method.";

reference
"RFC 8584: Framework for Ethernet VPN Designated Forwarder Election Extensibility";

} container vpws-service-instance {
  when "derived-from-or-self(../../../
+ "vpn-type, 'vpn-common:vpws-evpn')" {
    description
    "Only applies for EVPN-VPWS.";
  }
  description
  "Local and remote VPWS Service Instance (VSI)";
  reference
  "RFC 8214: Virtual Private Wire Service Support in Ethernet VPN";
  choice local-vsi-choice {
    description
    "Choices for assigning local VSI.";
    case directly-assigned {
      description
      "Explicitly assign a local VSI.";
      leaf local-vpws-service-instance {
        type uint32 {
          range "1..16777215";
        }
        description
        "Indicates the assigned local VSI.";
      }  
    }
    case auto-assigned {
      description
      "The local VSI is auto-assigned.";
      container local-vsi-auto {
        description
        "The local VSI is auto-assigned.";
        choice auto-mode {
          description
          "Indicates the auto-assignment mode of local VSI. VSI can be
automatically assigned either with or without indicating a pool from which the VSI should be taken.

For both cases, the server will auto-assign a local VSI value and use that value."

case from-pool {
  leaf vsi-pool-name {
    type string;
    description
    "The auto-assignment will be made from this pool.";
  }
}

case full-auto {
  leaf auto {
    type empty;
    description
    "Indicates that a local VSI is fully auto-assigned.";
  }
}

leaf auto-local-vsi {
  type uint32 {
    range "1..16777215";
  }
  config false;
  description
  "The value of the auto-assigned local VSI.";
}

}

choice remote-vsi-choice {
  description
  "Choice for assigning the remote VSI.";
  case directly-assigned {
    description
    "Explicitly assign a remote VSI.";
    leaf remote-vpws-service-instance {
      type uint32 {
        range "1..16777215";
      }
      description
      "";
    }
  }
}
"Indicates the value of the remote VSI.";
}
}

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

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

leaf auto-remote-vsi {
  type uint32 {
    range "1..16777215";
  }
  config false;
  description
  "The value of the auto-assigned remote VSI.";
}
list group {
    key "group-id";
    description
        "List of group-ids."
    leaf group-id {
        type string;
        description
            "Indicates the group-id to which the network
             access belongs to."
    }
    leaf precedence {
        type identityref {
            base precedence-type;
        }
        description
            "Defining service redundancy in transport
             network."
    }
    leaf ethernet-segment-identifier {
        type l2vpn-es:es-ref;
        description
            "Reference to the ESI associated with the VPN
             network access."
    }
}

container ethernet-service-oam {
    description
        "Container for Ethernet service OAM."
    leaf md-name {
        type string;
        description
            "Maintenance domain name."
    }
    leaf md-level {
        type uint8;
        description
            "Maintenance domain level."
    }
    container cfm-802.1-ag {
        description
            ""
"Container of 802.1ag CFM configurations.

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

"Choice based upon bandwidth type.";

case per-cos {
  description
  "Bandwidth per CoS.";
  list cos {
    key "cos-id";
    description
    "List of class of services.";
    leaf cos-id {
      type uint8;
      description
      "Identifier of the CoS, indicated by DSCP or a CE-CLAN CoS (802.1p) value in the service frame.";
      reference
      "IEEE Std 802.1Q: Bridges and Bridged Networks";
    }
    uses bandwidth-parameters;
  }
}

case other {
  description
  "Other bandwidth types.";
  uses bandwidth-parameters;
}

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

container qos {
  if-feature "vpn-common:qos";
  description
  "QoS configuration.";
  container qos-classification-policy {
    description
    "Configuration of the traffic classification
    policy.";
    list rule {
      key "id";
      ordered-by user;
      description
      "List of classification rules.";
      leaf id {
        type string;
        description
  ...}
"A description identifying the QoS classification policy rule."
}

choice match-type {
  default "match-flow";
  description
    "Choice for classification."
  case match-flow {
    container match-flow {
      description
        "Describes flow-matching criteria.";
      leaf dscp {
        type inet:dscp;
        description
          "DSCP value.";
      }
      leaf dot1q {
        type uint16;
        description
          "802.1Q matching. It is a VLAN tag added into a frame.";
        reference
          "IEEE Std 802.1Q: Bridges and Bridged Networks";
      }
      leaf pcp {
        type uint8 {
          range "0..7";
        }
        description
          "Priority Code Point (PCP) value.";
      }
      leaf src-mac-address {
        type yang:mac-address;
        description
          "Source MAC address.";
      }
      leaf dst-mac-address {
        type yang:mac-address;
        description
          "Destination MAC address.";
      }
      leaf color-type {
        type identityref {
          base color-type;
        }
        description
      }
    }
  }
}
"Color type."
}
leaf any {
    type empty;
    description
    "Allows all.";
}
}
case match-application {
    leaf match-application {
        type identityref {
            base vpn-common:customer-application;
        }
        description
        "Defines the application to match.";
    }
}
leaf target-class-id {
    type string;
    description
    "Identification of the CoS. This identifier is internal to the administration.";
}
}
}
container qos-profile {
    description
    "QoS profile configuration.";
list qos-profile {
    key "profile";
    description
    "QoS profile. Can be standard profile or customized profile.";
    leaf profile {
        type leafref {
            path "/l2vpn-ntw/vpn-profiles" + "/valid-provider-identifiers" + "/qos-profile-identifier/id";
        }
        description
        "QoS profile to be used.";
    }
    leaf direction {
        type identityref {

base vpn-common:qos-profile-direction;
} default "vpn-common:both";

description "The direction to which the QoS profile is applied."
};
}
}
}
}
}
}
container mac-policies {

description "Container for MAC-related policies.";

list access-control-list {
    key "name";
    description "Container for access control List.";
    leaf name {
        type string;
        description "Specifies the name of the ACL.";
    }
    leaf-list src-mac-address {
        type yang:mac-address;
        description "Specifies the source MAC address.";
    }
    leaf-list src-mac-address-mask {
        type yang:mac-address;
        description "Specifies the source MAC address mask.";
    }
    leaf-list dst-mac-address {
        type yang:mac-address;
        description "Specifies the destination MAC address.";
    }
    leaf-list dst-mac-address-mask {
        type yang:mac-address;
        description "Specifies the destination MAC address mask.";
    }
    leaf action {
        type identityref {
            base mac-action;
        }
        default "drop";
    }
}
description
  "Specifies the filtering action."
}
leaf rate-limit {
  when "derived-from-or-self(../action, 
    + "'/flood")"
    description
      "Rate-limit is valid only when the action
        is to accept the matching frame."
  }
  type decimal64 {
    fraction-digits 2;
  }
  units "bytes per second";
  description
    "Specifies how to rate-limit the traffic."
}
}
container mac-loop-prevention {
  description
    "Container of MAC loop prevention."
  leaf window {
    type uint32;
    units "seconds";
    default "180";
    description
      "The timer when a MAC mobility event is 
        detected."
  }
  leaf frequency {
    type uint32;
    default "5";
    description
      "The number of times to detect MAC 
        duplication, where a 'duplicate MAC 
        address' situation has occurred and 
        the duplicate MAC address has been 
        added to a list of duplicate MAC 
        addresses."
  }
  leaf retry-timer {
    type uint32;
    units "seconds";
    description
      "The retry timer. When the retry timer 
        expires, the duplicate MAC address will 
        be flushed from the MAC-VRF."
  }
}
leaf protection-type {
  type identityref {
    base loop-prevention-type;
  }
  default "trap";
  description
    "Protection type";
}

container mac-addr-limit {
  description
    "Container of MAC-Addr limit configurations";
  leaf limit-number {
    type uint16;
    default "2";
    description
      "Maximum number of MAC addresses learned
       from the subscriber for a single service
       instance.";
  }
  leaf time-interval {
    type uint32;
    units "milliseconds";
    default "300";
    description
      "The aging time of the mac address.";
  }
  leaf action {
    type identityref {
      base mac-action;
    }
    default "warning";
    description
      "Specifies the action when the upper limit
       is exceeded: drop the packet, flood the
       packet, or log a warning message (without
       dropping the packet).";
  }
}

container broadcast-unknown-unicast-multicast {
  description
    "Container of broadcast, unknown unicast, and
    multicast configurations";
  leaf multicast-site-type {
    type enumeration {
      enum receiver-only {
        description
      }
    }
  }
}
"The site only has receivers."
}
enum source-only {

description
"The site only has sources."
}
enum source-receiver {

description
"The site has both sources and receivers."
}
}
default "source-receiver";
description
"Type of the multicast site."
}
list multicast-gp-address-mapping {

description
"List of Port to group mappings."
leaf id {

type uint16;
description
"Unique identifier for the mapping."
}
leaf vlan-id {

type uint32;
mandatory true;
description
"The VLAN ID of the multicast group."
}
leaf mac-gp-address {

type yang:mac-address;
mandatory true;
description
"The MAC address of the multicast group."
}
leaf port-lag-number {

type uint32;
description
"The port/LAG belonging to the multicast group."
}
leaf bum-overall-rate {

type uint64;
units "bps";
description
9. Security Considerations

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

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

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

* 'vpn-profiles': This container includes a set of sensitive data that influence how the L3VPN service is delivered. For example, an attacker who has access to these data nodes may be able to manipulate routing policies, QoS policies, or encryption properties. These data nodes are defined with "nacm:default-deny-write" tagging [RFC9181].
* ‘ethernet-segments’ and ‘vpn-services’: An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L2VPN service, interrupting all the traffic of a client. In addition, an attacker may modify the attributes of a running service (e.g., QoS, bandwidth) or an ES, leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create an L2VPN service, add a new network access, or intercept/redirect the traffic to a non-authorized node. In addition to using NACM to prevent authorized access, such activity can be detected by adequately monitoring and tracking network configuration changes.

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

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

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

10. IANA Considerations

10.1. Registering YANG Modules

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

Boucadair, et al. Expires 4 December 2022
This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry:

```yaml
named: iana-bgp-l2-encaps
maintained by IANA: Y
prefix: iana-bgp-l2-encaps
reference: RFC XXXX
```

```yaml
name: iana-pseudowire-types
maintained by IANA: Y
prefix: iana-pw-types
reference: RFC XXXX
```

```yaml
name: ietf-ethernet-segment
maintained by IANA: N
prefix: l2vpn-es
reference: RFC XXXX
```

```yaml
name: ietf-l2vpn-ntw
maintained by IANA: N
prefix: l2vpn-ntw
reference: RFC XXXX
```
10.2. BGP Layer 2 Encapsulation Types

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

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

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

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

"description": Replicates the description from the registry.

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

Unassigned or reserved values are not present in the module.

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

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

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

10.3. Pseudowire Types

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

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

When a pseudowire type is added to the "iana-pseudowire-types" registry, a new "identity" statement must be added to the "iana-pseudowire-types" YANG module. The name of the "identity" is a
lower-case version of the encapsulation name provided in the
description. The "identity" statement should have the following sub-
statements defined:

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

"description": Replicates the description from the registry.

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

Unassigned or reserved values are not present in the module.

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

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

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

11. References

11.1. Normative References

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

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

[IEEE-802-1ag]
IEEE, "802.1ag - 2007 - IEEE Standard for Local and
Metropolitan Area Networks - Virtual Bridged Local Area
Networks Amendment 5: Connectivity Fault Management",

[IEEE802.1Qcp-2018]
IEEE, "IEEE Standard for Local and metropolitan area
networks--Bridges and Bridged Networks--Amendment 30: YANG
Data Model", September 2018,


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

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

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

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

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

[I-D.ietf-teas-ietf-network-slices]
Farrel, A., Drake, J., Rokui, R., Homma, S., Makhijani,
K., Contreras, L. M., and J. Tantsura, "Framework for IETF
Network Slices", Work in Progress, Internet-Draft, draft-
ietf-teas-ietf-network-slices-10, 27 March 2022,
<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-
network-slices-10.txt>.
[I-D.ietf-teas-te-service-mapping-yang]
Lee, Y., Dhody, D., Fioccola, G., Wu, Q., Ceccarelli, D.,
and J. Tantsura, "Traffic Engineering (TE) and Service
Mapping YANG Model", Work in Progress, Internet-Draft,
draft-ietf-teas-te-service-mapping-yang-10, 7 March 2022,
<https://www.ietf.org/archive/id/draft-ietf-teas-te-
service-mapping-yang-10.txt>.

[IEEE-802-1ah]
IEEE, "IEEE Standard for Local and metropolitan area
networks -- Virtual Bridged Local Area Networks Amendment
7: Provider Backbone Bridges", IEEE Std 801.3AH-2008,

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

[IEEE802.1AX]

[IEEE802.1Q]
"Bridges and Bridged Networks", IEEE Std 802.1Q-2018, 6

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

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

[RFC2507]
Degermark, M., Nordgren, B., and S. Pink, "IP Header
Compression", RFC 2507, DOI 10.17487/RFC2507, February

[RFC2508]
Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP
Headers for Low-Speed Serial Links", RFC 2508,
DOI 10.17487/RFC2508, February 1999,


Appendix A.  Examples

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

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

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

A.1.  BGP-based VPLS

This section provides an example to illustrate how the L2NM can be used to manage BGP-based VPLS. We consider the sample VPLS service delivered using the architecture depicted in Figure 23. In accordance with [RFC4761], we assume that a full mesh is established between all PEs. The details about such full mesh are not detailed here.
Figure 24 show an example of a message body used to configure a VPLS instance using the L2NM. In this example, BGP is used for both auto-discovery and signaling. The 'signaling-type' data node is set to 'vpn-common:bgp-signaling'.

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

```json
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpls7714825356",
          "vpn-description": "Sample BGP-based VPLS",
          "customer-name": "customer-7714825356",
          "vpn-type": "ietf-vpn-common:vpls",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:bgp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65535,
                "svc-mtu": 1518,
                "rd-suffix": 1,
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
              }
            ]
          }
        }
      ]
    }
  }
}
```
"vpn-nodes": {
  "vpn-node": [
  {
    "vpn-node-id": "pe1",
    "ne-id": "198.51.100.1",
    "active-global-parameters-profiles": {
      "global-parameters-profile": [
        {
          "profile-id": "simple-profile"
        }
      ]
    },
    "bgp-auto-discovery": {
      "vpn-id": "1"
    },
    "signaling-option": {
      "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\-tagged-mode",
      "vpls-instance": {
        "vpls-edge-id": 1,
        "vpls-edge-id-range": 100
      }
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "1/1/1.1",
          "interface-id": "1/1/1",
          "description": "Interface to CE1",
          "active-vpn-node-profile": "simple-profile",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          },
          "connection": {
            "encapsulation": {
              "encap-type": "ietf-vpn-common:dot1q",
              "dot1q": {
                "cvlan-id": 1
              }
            }
          }
        }
      ]
    }
  }
]}

{
  "vpn-node-id": "pe2",
  "ne-id": "198.51.100.2",
  "active-global-parameters-profiles": {
    "global-parameters-profile": [
      {
        "profile-id": "simple-profile"
      }
    ]
  },
  "bgp-auto-discovery": {
    "vpn-id": "1"
  },
  "signaling-option": {
    "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\ntagged-mode",
    "vpls-instance": {
      "vpls-edge-id": 2,
      "vpls-edge-id-range": 100
    }
  },
  "vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1.1",
        "description": "Interface to CE2",
        "active-vpn-node-profile": "simple-profile",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        },
        "connection": {
          "encapsulation": {
            "encap-type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        }
      }
    ]
  }
}
"vpn-node-id": "pe3",
"ne-id": "198.51.100.3",
"active-global-parameters-profiles": {
  "global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ]
},
"bgp-auto-discovery": {
  "vpn-id": "1"
},
"signaling-option": {
  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\ntagged-mode",
  "vpls-instance": {
    "vpls-edge-id": 3,
    "vpls-edge-id-range": 100
  }
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE3",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 1
          }
        }
      }
    }
  ]
},
"vpn-node-id": "pe4",
"ne-id": "198.51.100.4",
"active-global-parameters-profiles": {
"global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ],
"bgp-auto-discovery": {
  "vpn-id": "1"
},
"signaling-option": {
  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\ntagged-mode",
  "vpls-instance": {
    "vpls-edge-id": 4,
    "vpls-edge-id-range": 100
  }
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE4",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 1
          }
        }
      }
    }
  ]
}
]
A.2. BGP-based VPWS with LDP Signaling

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

```
+-----+   +--------------+   +-----+
| CE1+--------+     |   |     Core     |   |     +--------+ CE2|
|       +-----+   +--------------+   +-----+       +----+
site1                                      site2
```

Figure 25: An Example of VPLS

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpws12345",
          "vpn-description": "Sample VPWS",
          "customer-name": "customer-12345",
          "vpn-type": "ietf-vpn-common:vpws",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65550,
                "rd-auto": {
                  "auto": [null]
                }
              }
            ],
            "vpn-target": {
              "id": 1,
              "route-targets": [{
                "route-target": "0:65535:1"
              }],
              "route-target-type": "both"
            }
          }
        }
      ]
    }
  }
}
```
"vpn-nodes": {
  "vpn-node": [
    {
      "vpn-node-id": "pe1",
      "ne-id": "2001:db8:100::1",
      "active-global-parameters-profiles": {
        "global-parameters-profile": [
          {
            "profile-id": "simple-profile"
          }
        ]
      }
    },
    "bgp-auto-discovery": {
      "vpn-id": "587"
    },
    "signaling-option": {
      "advertise-mtu": true,
      "ldp-or-l2tp": {
        "saii": 1,
        "remote-targets": [
          {
            "taii": 2
          }
        ],
        "t-ldp-pw-type": "ethernet"
      }
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "1/1/1.1",
          "interface-id": "1/1/1",
          "description": "Interface to CE1",
          "active-vpn-node-profile": "simple-profile",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          }"
{  
  "vpn-node-id": "pe2",
  "ne-id": "2001:db8:200::1",
  "active-global-parameters-profiles": {  
    "global-parameters-profile": [  
      {  
        "profile-id": "simple-profile"
      }
    ]
  },  
  "bgp-auto-discovery": {  
    "vpn-id": "587"
  },  
  "signaling-option": {  
    "advertise-mtu": true,
    "ldp-or-l2tp": {  
      "saii": 2,
      "remote-targets": [  
        {  
          "taii": 1
        }
      ],
      "t-ldp-pw-type": "ethernet"
    }
  },  
  "vpn-network-accesses": {  
    "vpn-network-access": [  
      {  
        "id": "5/1/1.1",
        "interface-id": "5/1/1",
        "description": "Interface to CE2",
        "active-vpn-node-profile": "simple-profile",
        "status": {  
          "admin-status": {  
            "status": "ietf-vpn-common:admin-up"
          }
        }
      }
    ]
  }
}
A.3. LDP-based VPLS

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

Figure 27: An Example of VPLS topology

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

------------------ NOTE: '\n' line wrapping per RFC 8792 ------------------

```json
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "450",
          "vpn-name": "CORPO-EXAMPLE",
          "vpn-description": "SEDE_CENTRO_450",
          "customer-name": "EXAMPLE",
          "vpn-type": "ietf-vpn-common:vpls",
          "vpn-service-topology": "ietf-vpn-common:hub-spoke",
          "bgp-ad-enabled": false,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": []
          }
        }
      ]
    }
  }
}
```
"profile-id": "simple-profile",
"ce-vlan-preservation": true,
"ce-vlan-cos-preservation": true
]}
},
"vpn-nodes": {
"vpn-node": [
{
"vpn-node-id": "450",
"description": "SEDE_CENTRO_450",
"ne-id": "2001:db8:5::1",
"role": "ietf-vpn-common:hub-role",
"status": {
"admin-status": {
"status": "ietf-vpn-common:admin-up"
}
},
"active-global-parameters-profiles": {
"global-parameters-profile": [
{
"profile-id": "simple-profile"
}
]
},
"signaling-option": {
"ldp-or-l2tp": {
"t-ldp-pw-type": "vpls-type",
"pw-peer-list": [
{
"peer-addr": "2001:db8:50::1",
"vc-id": "1543"
}
]
}
},
"vpn-network-accesses": {
"vpn-network-access": [
{
"id": "4508671287",
"description": "VPN_450_SNA",
"interface-id": "gigabithethernet0/0/1",
"status": {
"admin-status": {
"status": "ietf-vpn-common:admin-up"
}
},
"connection": {
"l2-termination-point": "550",
"encapsulation": {
    "encap-type": "ietf-vpn-common:dot1q",
    "dot1q": {
        "tag-type": "ietf-vpn-common:c-vlan",
        "cvlan-id": 550
    }
},
"service": {
    "mtu": 1550,
    "svc-pe-to-ce-bandwidth": {
        "pe-to-ce-bandwidth": [
            {
                "bw-type": "ietf-vpn-common:bw-per-port",
                "cir": "20480000"
            }
        ],
    }
},
"svc-ce-to-pe-bandwidth": {
    "ce-to-pe-bandwidth": [
        {
            "bw-type": "ietf-vpn-common:bw-per-port",
            "cir": "20480000"
        }
    ],
"qos": {
    "qos-profile": {
        "qos-profile": [
            {
                "profile": "QoS_Profile_A",
                "direction": "ietf-vpn-common:both"
            }
        ],
    }
}
},

"vpn-node-id": "451",
"description": "SEDE_CHAPINERO_451",
"ne-id": "2001:db8:50::1",
"role": "ietf-vpn-common:spoke-role",
"status": {
"admin-status": {
  "status": "ietf-vpn-common:admin-up"
}
},
"active-global-parameters-profiles": {
  "global-parameters-profile": [
  {
    "profile-id": "simple-profile"
  }
  ]
},
"signaling-option": {
  "ldp-or-l2tp": {
    "t-ldp-pw-type": "vpls-type",
    "pw-peer-list": [
    {
      "peer-addr": "2001:db8:5::1",
      "vc-id": "1543"
    }
    ]
  }
},
"vpn-network-accesses": {
  "vpn-network-access": [
  {
    "id": "4508671288",
    "description": "VPN_450_SNA",
    "interface-id": "gigabithethernet0/0/1",
    "status": {
      "admin-status": {
        "status": "ietf-vpn-common:admin-up"
      }
    },
    "connection": {
      "12-termination-point": "550",
      "encapsulation": {
        "encap-type": "ietf-vpn-common:dot1q",
        "dot1q": {
          "tag-type": "ietf-vpn-common:c-vlan",
          "cvlan-id": 550
        }
      }
    },
    "service": {
      "mtu": 1550,
      "svc-pe-to-ce-bandwidth": {
        "pe-to-ce-bandwidth": [
        
      ]
    }
  }
}
"bw-type": "ietf-vpn-common:bw-per-port",
"cir": "20480000"
],
"svc-ce-to-pe-bandwidth": {
"ce-to-pe-bandwidth": [
{
"bw-type": "ietf-vpn-common:bw-per-port",
"cir": "20480000"
}
],
"qos": {
"qos-profile": {
"qos-profile": [
{
"profile": "QoS_Profile_A",
"direction": "ietf-vpn-common:both"
}
]
}
}
}
}
}
}

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

A.4. VPWS-EVPN Service Instance

Figure 29 depicts a sample architecture to offer VPWS-EVPN service between CE1 and CE2. Both CEs are multi-homed. BGP sessions are maintained between these PEs as per [RFC8214]. In this EVPN instance, an All-Active redundancy mode is used.
Let's first suppose that the following ES was created (Figure 30).

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

Figure 29 shows a simplified configuration to illustrate the use of the L2NM to configured VPWS-EVPN instance.
{"ietf-l2vpn-ntw:l2vpn-ntw": {
  "vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "vpws15432855",
        "vpn-description": "Sample VPWS-EVPN",
        "customer-name": "customer_15432855",
        "vpn-type": "ietf-vpn-common:vpws-evpn",
        "bgp-ad-enabled": true,
        "signaling-type": "ietf-vpn-common:bgp-signaling",
        "global-parameters-profiles": {
          "global-parameters-profile": [
            {
              "profile-id": "simple-profile",
              "local-autonomous-system": 65535,
              "rd-suffix": 1,
              "vpn-target": [
                {
                  "id": 1,
                  "route-targets": [
                    {
                      "route-target": "0:65535:1"
                    }
                  ],
                  "route-target-type": "both"
                }
              ]
            }
          ]
        },
        "vpn-nodes": {
          "vpn-node": [
            {
              "vpn-node-id": "pe1",
              "ne-id": "198.51.100.1",
              "active-global-parameters-profiles": {
                "global-parameters-profile": [
                  {
                    "profile-id": "simple-profile"
                  }
                ]
              },
              "vpn-network-accesses": {
                "vpn-network-access": [
                  {
                    "id": "1/1/1.1",
                    "interface-id": "1/1/1",
                    "interface-type": "ethernet",
                    "vlan": 100,
                    "ip-address": "192.168.1.1/24"
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
},
"vpn-nodes": {
  "vpn-node": [
    {
      "vpn-node-id": "pel",
      "ne-id": "198.51.100.1",
      "active-global-parameters-profiles": {
        "global-parameters-profile": [
          {
            "profile-id": "simple-profile"
          }
        ]
      },
      "vpn-network-accesses": {
        "vpn-network-access": [
          {
            "id": "1/1/1.1",
            "interface-id": "1/1/1",
            "interface-type": "ethernet",
            "vlan": 200,
            "ip-address": "192.168.2.1/24"
          }
        ]
      }
    }
  ]
},
"vpn-ntws": {
  "vpn-ntw": [
    {
      "vpn-ntw-id": "sample-ntw",
      "customer-name": "customer_15432855",
      "vpn-type": "ietf-vpn-common:vpws-evpn",
      "bgp-ad-enabled": true,
      "signaling-type": "ietf-vpn-common:bgp-signaling",
      "global-parameters-profiles": {
        "global-parameters-profile": [
          {
            "profile-id": "simple-profile",
            "local-autonomous-system": 65535,
            "rd-suffix": 1,
            "vpn-target": [
              {
                "id": 1,
                "route-targets": [
                  {
                    "route-target": "0:65535:1"
                  }
                ],
                "route-target-type": "both"
              }
            ]
          }
        ]
      },
      "vpn-nodes": {
        "vpn-node": [
          {
            "vpn-node-id": "pe1",
            "ne-id": "198.51.100.1",
            "active-global-parameters-profiles": {
              "global-parameters-profile": [
                {
                  "profile-id": "simple-profile"
                }
              ]
            },
            "vpn-network-accesses": {
              "vpn-network-access": [
                {
                  "id": "1/1/1.1",
                  "interface-id": "1/1/1",
                  "interface-type": "ethernet",
                  "vlan": 100,
                  "ip-address": "192.168.1.1/24"
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
}
"description": "Interface to CE1",
"active-vpn-node-profile": "simple-profile",
"status": {
    "admin-status": {
        "status": "ietf-vpn-common:admin-up"
    }
},
"connection": {
    "encapsulation": {
        "encap-type": "ietf-vpn-common:dot1q",
        "dot1q": {
            "cvlan-id": 1
        }
    }
},
"vpws-service-instance": {
    "local-vpws-service-instance": 1111,
    "remote-vpws-service-instance": 1112
},
"group": [
    {
        "group-id": "gr1",
        "ethernet-segment-identifier": "esi1"
    }
]
},
"vpn-node-id": "pe2",
"ne-id": "198.51.100.2",
"active-global-parameters-profiles": {
    "global-parameters-profile": [ {
        "profile-id": "simple-profile"
    }
]
},
"vpn-network-accesses": {
    "vpn-network-access": [
        {
            "id": "1/1/1.1",
            "interface-id": "1/1/1",
            "description": "Interface to CE1",
            "active-vpn-node-profile": "simple-profile",
            "status": {
                "admin-status": {
                     }}
        }
]
"status": "ietf-vpn-common:admin-up"
},
"connection": {
  "encapsulation": {
    "encap-type": "ietf-vpn-common:dot1q",
    "dot1q": {
      "cvlan-id": 1
    }
  }
},
"vpws-service-instance": {
  "local-vpws-service-instance": 1111,
  "remote-vpws-service-instance": 1112
},
"group": [
  {
    "group-id": "gr1",
    "ethernet-segment-identifier": "esi1"
  }
]
},
"vpn-node-id": "pe3",
"ne-id": "198.51.100.3",
"active-global-parameters-profiles": {
  "global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ]
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE2",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {

"encapsulation": {
  "encap-type": "ietf-vpn-common:dot1q",
  "dot1q": {
    "cvlan-id": 1
  }
},
"vpws-service-instance": {
  "local-vpws-service-instance": 1112,
  "remote-vpws-service-instance": 1111
},
"group": [
  {
    "group-id": "gr1",
    "ethernet-segment-identifier": "esi2"
  }
]
},
"vpn-node-id": "pe4",
"ne-id": "198.51.100.4",
"active-global-parameters-profiles": {
  "global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ]
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE2",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 1
          }
        }
      }
    }
  ]
}
A.5. Automatic ESI Assignment

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

Figure 32: Example Architecture for EVPN Service Delivery

ES
+-----+  +-----+  +-----+  +-----+
| PE1 |  |------+  |------+  | PE3 |
+-----+  +-----+  +-----+  +-----+  +-----+
| CE1  |  | CE2  |  | Core  |  | CE3  |
+-----+  +-----+  +-----+  +-----+  +-----+
| LACP |  |------+  |------+  | PE4 |
+-----+  +-----+  +-----+  +-----+  +-----+
Figure 32: An Example of Automatic ESI Assignment

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

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

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

```json
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "ietf-l2vpn-ntw:vpn-services": {
      "vpn-service": {
        "vpn-id": "auto-esi-lacp",
        "vpn-description": "Sample to illustrate auto-ESI",
        "vpn-type": "ietf-vpn-common:vpws-evpn",
        "vpn-nodes": {
          "vpn-node": ["pe1",
            {"ne-id": "198.51.100.1", "status": {"admin-status": {"status": "ietf-vpn-common:admin-up"}}}
          ]
        }
      }
    }
  }
}
```
"connection": {
    "lag-interface": {
        "lag-interface-id": "1",
        "lacp": {
            "lacp-state": true,
            "system-id": "11:00:11:00:11:11",
            "admin-key": 154
        }
    }
},
"group": [
    {
        "group-id": "gr1",
        "ethernet-segment-identifier": "esi1"
    }
]
},
"vpn-node-id": "pe2",
"ne-id": "198.51.100.2",
"vpn-network-accesses": {
    "vpn-network-access": [
        {
            "id": "2/2/2.5",
            "interface-id": "2/2/2",
            "description": "Interface to CE1",
            "status": {
                "admin-status": {
                    "status": "ietf-vpn-common:admin-up"
                }
            }
        },
        "connection": {
            "lag-interface": {
                "lag-interface-id": "1",
                "lacp": {
                    "lacp-state": true,
                    "system-id": "11:00:11:00:11:11",
                    "admin-key": 154
                }
            }
        },
        "group": [
            {
                "group-id": "gr1",
                "ethernet-segment-identifier": "esi1"
            }
        ]
    ]
}
The auto-assigned ESI can be retrieved using, e.g., a GET RESTCONF method. The assigned value will be then returned as shown in the 'esi-auto' data node in Figure 35.

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

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

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

A.6. VPN Network Access Precedence

In reference to the example depicted in Figure 36, an L2VPN service involves two VPN network accesses to sites that belong to the same customer.
In order to tag one of these VPN network accesses as "primary" and the other one as "secondary", Figure 37 shows an excerpt of the corresponding L2NM configuration. In such a configuration, both accesses are bound to the same "group-id" and the "precedence" data node set as function of the intended role of each access (primary or secondary).

Figure 36: Example of Multiple VPN Network Accesses
{"ietf-l2vpn-ntw:l2vpn-ntw": {
  "vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "Sample-Service",
        "vpn-nodes": {
          "vpn-node": [
            {
              "vpn-node-id": "VPN-NODE",
              "vpn-network-accesses": {
                "vpn-network-access": [
                  {
                    "id": "NET-ACC-1",
                    "connection": {
                      "bearer-reference": "br1"
                    },
                    "group": [
                      {
                        "group-id": "1",
                        "precedence": "primary"
                      }
                    ]
                  },
                  {
                    "id": "NET-ACC-2",
                    "connection": {
                      "bearer-reference": "br2"
                    },
                    "group": [
                      {
                        "group-id": "1",
                        "precedence": "secondary"
                      }
                    ]
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
Acknowledgements

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

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

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

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

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

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

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

Contributors

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

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

Raul Arco
Nokia
Email: raul.arco@nokia.com
Authors’ Addresses

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

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

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

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

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

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 April 2022.

Copyright Notice

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

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

Table of Contents

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

[RFC8299] defines a Layer 3 Virtual Private Network Service YANG data model (L3SM) that can be used for communication between customers and service providers. Such a model focuses on describing the customer view of the Virtual Private Network (VPN) services and provides an abstracted view of the customer’s requested services. That approach limits the usage of the L3SM to the role of a customer service model (as per [RFC8309]).
This document defines a YANG module called L3VPN Network Model (L3NM). The L3NM is aimed at providing a network-centric view of Layer 3 (L3) VPN services. This data model can be used to facilitate communication between the service orchestrator and the network controller/orchestrator by allowing for more network-centric information to be included. It enables further capabilities such as resource management or serves as a multi-domain orchestration interface, where logical resources (such as route targets or route distinguishers) must be coordinated.

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

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

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

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

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

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

The L3NM focuses on BGP Provider Edge (PE) based Layer 3 VPNs as described in [RFC4026][RFC4110][RFC4364] and Multicast VPNs as described in [RFC6037][RFC6513].
The YANG data model in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

2. Terminology

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

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

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

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

This document makes use of the following terms:

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

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

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

Network orchestrator: A functional entity that is hierarchically
intermediate between a service orchestrator and network controllers. A network orchestrator can manage one or several network controllers.

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

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

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

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

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

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

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

3. Acronyms

The following acronyms are used in the document:

- ACL: Access Control List
- AS: Autonomous System
- ASM: Any-Source Multicast
- ASN: AS Number
- BSR: Bootstrap Router
- BFD: Bidirectional Forwarding Detection
- BGP: Border Gateway Protocol
- CE: Customer Edge
4. L3NM Reference Architecture

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

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

The intelligence for translating customer-facing information into network-centric one (and vice versa) is implementation specific.
The terminology from [RFC8309] is introduced to show the distinction between the customer service model, the service delivery model, the network configuration model, and the device configuration model. In that context, the "Domain Orchestration" and "Config Manager" roles may be performed by "Controllers".

![Diagram of L3NM Reference Architecture](image)

Figure 1: L3NM Reference Architecture

The customer may use a variety of means to request a service that may trigger the instantiation of an L3NM. The customer may use the L3SM or more abstract models to request a service that relies upon an L3VPN service. For example, the customer may supply an IP Connectivity Provisioning Profile (CPP) that characterizes the
requested service [RFC7297], an enhanced VPN (VPN+) service [I-D.ietf-teas-enhanced-vpn], or an IETF network slice service [I-D.ietf-teas-ietf-network-slices].

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

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

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

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

L3SM: L3NM is not a customer service model.

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

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

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

Network Topology Modules: An L3VPN involves nodes that are part of a
topology managed by the service provider network. The topology can be represented using the network topology YANG module defined in [RFC8345] or its extension such as a User-Network Interface (UNI) topology module (e.g., [I-D.ogondio-opsawg-uni-topology]).

Device Modules: L3NM is not a device model.

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

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

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

6. Sample Uses of the L3NM Data Model

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

6.1. Enterprise Layer 3 VPN Services

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

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

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

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

6.3. Management of Multicast Services

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

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

7. Description of the L3NM YANG Module

The L3NM ('ietf-l3vpn-ntw') is defined to manage L3VPNs in a service provider network. In particular, the 'ietf-l3vpn-ntw' module can be used to create, modify, and retrieve L3VPN services of a network.
The full tree diagram of the module can be generated using the "pyang" tool [PYANG]. That tree is not included here because it is too long (Section 3.3 of [RFC8340]). Instead, subtrees are provided for the reader's convenience.

7.1. Overall Structure of the Module

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

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

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

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

Figure 3: Overall L3NM Tree Structure

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

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

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

Figure 4: VPN Profiles Subtree Structure
```

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

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

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

‘qos-profile-identifier’: A Quality of Service (QoS) profile refers to a set of policies such as classification, marking, and actions (e.g., [RFC3644]).
'bfd-profile-identifier': A Bidirectional Forwarding Detection (BFD) profile refers to a set of BFD [RFC5880] policies that can be invoked when building a VPN service.

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

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

7.3. VPN Services

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

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

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

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

'vpn-description': Includes a textual description of the service.
The internal structure of a VPN description is local to each VPN service provider.

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

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

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

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

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

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

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

More details are provided in Section 7.4.

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

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

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

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

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

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

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

More details are provided in Section 7.5.
7.4. VPN Instance Profiles

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

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

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

... +--rw vpn-instance-profile* [profile-id]
      +--rw profile-id               string
      +--rw role?                    identityref
      +--rw local-as?                 inet:as-number
         |   {vpn-common:rtg-bgp}?      
      +--rw (rd-choice)?             
         |   +(directly-assigned)
         |      +--rw rd?               rt-types:route-distinguisher
         |         +(directly-assigned-suffix)
         |            +--rw rd-suffix?    uint16
         |   +(auto-assigned)
         |      +--rw rd-auto
         |         |   +(auto-mode)?            
         |         |      +(from-pool)
         |         |         +--rw rd-pool-name?   string
         |         |         +(full-auto)
         |         |            +--rw auto?     empty
         |         +--ro auto-assigned-rd?
         |            rt-types:route-distinguisher
         |   +(auto-assigned-suffix)
         |      +--rw rd-auto-suffix
         |         |   +(auto-mode)?            
         |         |      +(from-pool)
         |         |         +--rw rd-pool-name?   string
         |         |         +(full-auto)
         |         |            +--rw auto?     empty
```
Figure 6: Subtree Structure of VPN Instance Profiles

The description of the listed data nodes is as follows:

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

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

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

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

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

'full-auto': The network controller auto-assigns RDs. This can apply for the deployment of new services.
'no-rd': The VPN service provider (service orchestrator) explicitly wants no RD to be assigned. This case can be used for CE testing within the network or for troubleshooting proposes.

Also, the module accommodates deployments where only the Assigned Number subfield of RDs (Section 4.2 of [RFC4364]) is assigned from a pool while the Administrator subfield is set to, e.g., the Router ID that is assigned to a VPN node. The module supports these modes for managing the Assigned Number subfield: explicit assignment, auto-assignment from a pool, and full auto-assignment.

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

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

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

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

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

7.5. VPN Nodes

The 'vpn-node' is an abstraction that represents a set of common policies applied on a given network node (typically, a PE) and belong to one L3VPN service. The 'vpn-node' includes a parameter to indicate the network node on which it is applied. In the case that the 'ne-id' points to a specific PE, the 'vpn-node' will likely be mapped into a VRF in the node. However, the model also allows pointing to an abstract node. In this case, the network controller will decide how to split the 'vpn-node' into VRFs.
++-rw vpn-node-id vpn-common:vpn-id
++-rw description? string
++-rw ne-id? string
++-rw local-as? inet:as-number
|       {vpn-common:rtg-bgp}?
+++rw router-id? rt-types:router-id
++-rw active-vpn-instance-profiles
    +--rw vpn-instance-profile* [profile-id]
      ++-rw profile-id leafref
      ++-rw router-id* [address-family]
      |      ++-rw address-family identityref
      |      ++-rw router-id? inet:ip-address
      ++-rw local-as? inet:as-number
      |       {vpn-common:rtg-bgp}?  
      ++-rw (rd-choice)?  
      |      ...  
      ++-rw address-family* [address-family]
      |      ++-rw address-family identityref  
      |      |      ...  
      |      ++-rw vpn-targets  
      |      |      ...  
      ++-rw maximum-routes* [protocol]  
      |      ...  
      ++-rw multicast {vpn-common:multicast}?  
      ...  
++-rw msdp {msdp}?
    ++-rw peer? inet:ipv4-address
    ++-rw local-address? inet:ipv4-address
++-rw status
    ++-rw admin-status
    |    ++-rw status? identityref
    |    ++-rw last-change? yang:date-and-time
    ++-rw oper-status
    |    ++-rw status? identityref
    |    ++-rw last-change? yang:date-and-time
++-rw groups
    ++-rw group* [group-id]
    |    ++-rw group-id string
++-rw status
    ++-rw admin-status
    |    ++-rw status? identityref
    |    ++-rw last-change? yang:date-and-time
    ++-ro oper-status
    |    ++-ro status? identityref
    |    ++-ro last-change? yang:date-and-time
++-rw vpn-network-accesses
...
In reference to the subtree shown in Figure 7, the description of VPN node data nodes is as follows:

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

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

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

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

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

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

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

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

'groups': Lists the groups to which a VPN node belongs to
The 'group-id' is used to associate, e.g., redundancy or protection constraints with VPN nodes.

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

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

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

See Section 7.6 for more details.

7.6. VPN Network Accesses

The 'vpn-network-access' includes a set of data nodes that describe the access information for the traffic that belongs to a particular L3VPN (Figure 8).
...  
++–rw vpn-nodes  
  ++–rw vpn-node* [vpn-node-id]  
...  
++–rw vpn-network-accesses  
++–rw vpn-network-access* [id]  
  ++–rw id  
    vpn-common:vpn-id  
  ++–rw interface-id?  
    string  
  ++–rw description?  
    string  
  ++–rw vpn-network-access-type?  
    identityref  
  ++–rw vpn-instance-profile?  
    leafref  
  ++–rw status  
    ++–rw admin-status  
      ++–rw status?  
        identityref  
      ++–rw last-change?  
        yang:date-and-time  
    ++–ro oper-status  
      ++–ro status?  
        identityref  
      ++–ro last-change?  
        yang:date-and-time  
  ++–rw connection  
  | ...  
  ++–rw ip-connection  
  | ...  
  ++–rw routing-protocols  
  | ...  
  ++–rw oam  
  | ...  
  ++–rw security  
  | ...  
  ++–rw service  
...  

Figure 8: VPN Network Access Subtree Structure

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

‘id’: Is an identifier of the VPN network access.

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

‘description’: Includes a textual description of the VPN network access.

‘vpn-network-access-type’: Is used to select the type of network interface to be deployed in the devices. The available defined values are:
'point-to-point': Represents a direct connection between the endpoints. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.

'multipoint': Represents a multipoint connection between the customer site and the PEs. The controller must keep the association between a logical or physical interface on the device with the 'id' of the 'vpn-network-access'.

'irb': Represents a connection coming from an L2VPN service. An identifier of such service ('l2vpn-id') may be included in the 'connection' container as depicted in Figure 9. The controller must keep the relationship between the logical tunnels or bridges on the devices with the 'id' of the 'vpn-network-access'.

'loopback': Represents the creation of a logical interface on a device. An example to illustrate how a loopback interface can be used in the L3NM is provided in Appendix A.2.

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

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

'connection': Represents and groups the set of Layer 2 connectivity from where the traffic of the L3VPN in a particular VPN Network access is coming. See Section 7.6.1.

'ip-connection': Contains Layer 3 connectivity information of a VPN network access (e.g., IP addressing). See Section 7.6.2.

'routing-protocols': Includes the CE-PE routing configuration information. See Section 7.6.3.

'oam': Specifies the Operations, Administration, and Maintenance (OAM) mechanisms used for a VPN network access. See Section 7.6.4.

'security': Specifies the authentication and the encryption to be applied for a given VPN network access. See Section 7.6.5.
'service': Specifies the service parameters (e.g., QoS, multicast) to apply for a given VPN network access. See Section 7.6.6.

7.6.1. Connection

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

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

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

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

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

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

A site, as per [RFC4176] represents a VPN customer’s location that is connected to the service provider network via a CE-PE link, which can access at least one VPN. The connection from the site to the service provider network is the bearer. Every site is associated with a list of bearers. A bearer is the layer two connection with the site. In the L3NM, it is assumed that the bearer has been allocated by the service provider at the service orchestration stage. The bearer is associated to a network element and a port. Hence, a bearer is just a 'bearer-reference' to allow the association between a service request (e.g., L3SM) and L3NM.
The L3NM can be used to create a LAG interface for a given L3VPN service ('lag-interface') [IEEE802.1AX]. Such a LAG interface can be referenced under 'interface-id' (Section 7.6).

```yang
+--rw connection
    |   +--rw encapsulation
    |       |   +--rw type?                identityref
    |       |       +--rw dot1q
    |       |       |   +--rw tag-type?          identityref
    |       |       |   +--rw cvlan-id?         uint16
    |       |       +--rw priority-tagged
    |       |       |   +--rw tag-type?          identityref
    |       |       +--rw qinq
    |       |       |   +--rw tag-type?          identityref
    |       |       +--rw svlan-id      uint16
    |       |       +--rw cvlan-id      uint16
    |   +--rw (l2-service)?
    |     |   +--:(l2-tunnel-service)
    |     |     +--rw 12-tunnel-service
    |     |     |   +--rw type?                identityref
    |     |     |     +--rw pseudowire
    |     |     |     |   +--rw vcid?     uint32
    |     |     |     |   +--rw far-end?  union
    |     |     |     +--rw vpls
    |     |     |     |   +--rw vcid?     uint32
    |     |     |     |   +--rw far-end* union
    |     |     +--rw vxlan
    |     |     |   +--rw vni-id             uint32
    |     |     |   +--rw peer-mode?         identityref
    |     |     |   +--rw peer-ip-address* inet:ip-address
    |     |   +--:(l2vpn)
    |     |     +--rw 12vpn-id?         vpn-common:vpn-id
    |     |     +--rw 12-termination-point?  string
    |     |     +--rw local-bridge-reference? string
    |     |     |   |   (vpn-common:bearer-reference)?
    |     |     +--rw lag-interface {vpn-common:lag-interface}?
    |     |     |   +--rw lag-interface-id? string
    |     |     +--rw member-link-list
    |     |     |   +--rw member-link* [name]
    |     |     |       +--rw name    string

Figure 9: Connection Subtree Structure
```
7.6.2. IP Connection

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

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

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

Figure 10: IP Connection Subtree Structure

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

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

Figure 11 shows the structure of the dynamic IPv4 address assignment (i.e., by means of DHCP).
Figure 11: IP Connection Subtree Structure (IPv4)

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

Barguil, et al. Expires 11 April 2022 [Page 31]
In the case of the static addressing (Figure 13), the model supports the assignment of several IP addresses in the same 'vpn-network-access'. To identify which of the addresses is the primary address of a connection, the 'primary-address' reference MUST be set with the corresponding 'address-id'.

--- (provider-dhcp)
  +--rw provider-dhcp
    +--rw dhcp-service-type?
      |   enumeration
    +--rw (service-type)?
      +--:(relay)
        +--rw server-ip-address?
          |   inet:ipv6-address
        +--:(server)
        +--rw (address-assign)?
          +--:(number)
            +--rw number-of-dynamic-address?
              |   uint16
          +--:(explicit)
            +--rw customer-addresses
              +--rw address-pool* [pool-id]
                +--rw pool-id
                +--rw start-address
                  |   inet:ipv6-address
                +--rw end-address?
                  |   inet:ipv6-address
          +--:(dhcp-relay)
            +--rw customer-dhcp-servers
              +--rw server-ip-address*
                |   inet:ipv6-address
            +--:(static-addresses)

Figure 12: IP Connection Subtree Structure (IPv6)
7.6.3.  CE-PE Routing Protocols

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

The subtree of the 'routing-protocols' is shown in Figure 14.
Multiple routing instances can be defined; each uniquely identified by an ‘id’. The type of routing instance is indicated in ‘type’. The values of these attributes are those defined in [I-D.ietf-opsawg-vpn-common] (‘routing-protocol-type’ identity).

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

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

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

7.6.3.1. Static Routing

The L3NM supports the configuration of one or more IPv4/IPv6 static routes. Since the same structure is used for both IPv4 and IPv6, it was considered to have one single container to group both static entries independently of their address family, but that design was abandoned to ease the mapping with the structure in [RFC8299].
As depicted in Figure 15, the following data nodes can be defined for a given IP prefix:

'lan-tag': Indicates a local tag (e.g., "myfavourite-lan") that is used to enforce local policies.
'next-hop': Indicates the next-hop to be used for the static route. It can be identified by an IP address, an interface, etc.

'bfd-enable': Indicates whether BFD is enabled or disabled for this static route entry.

'metric': Indicates the metric associated with the static route entry.

'preference': Indicates the preference associated with the static route entry. This preference is used to selecting a preferred route among routes to the same destination prefix.

'status': Used to convey the status of a static route entry. This data node can also be used to control the (de)activation of individual static route entries.

7.6.3.2. BGP

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

```yang
---rw routing-protocols
  +--rw routing-protocol* [id]
  ...
  +--rw bgp
    ---rw description? string
    +--rw local-as? inet:as-number
    +--rw peer-as inet:as-number
    +--rw address-family? identityref
    +--rw local-address? union
    +--rw neighbor* inet:ip-address
    +--rw multihop? uint8
    +--rw as-override? boolean
    +--rw allow-own-as? uint8
    +--rw prepend-global-as? boolean
    +--rw send-default-route? boolean
    +--rw site-of-origin? rt-types:route-origin
    +--rw ipv6-site-of-origin? rt-types:ipv6-route-origin
    +--rw redistribute-connected* [address-family]
      +--rw address-family identityref
      +--rw enable? boolean
    +--rw bgp-max-prefix
```
The following data nodes are captured in Figure 16. It is up to the implementation (e.g., network orchestrator) to derive the corresponding BGP device configuration:

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

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

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

'address-family': Indicates the address-family of the peer. It can be set to IPv4, IPv6, or dual-stack.
This address family will be used together with the 'vpn-type' to derive the appropriate Address Family Identifiers (AFIs)/Subsequent Address Family Identifiers (SAFIs) that will be part of the derived device configurations (e.g., Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128) defined in Section 4.3.4 of [RFC4364]).

'local-address': Specifies an address or a reference to an interface to use when establishing the BGP transport session.

'neighbor': Can indicate two neighbors (each for a given address-family) or one neighbor (if 'address-family' attribute is set to dual-stack). A list of IP address(es) of the BGP neighbors can be then conveyed in this data node.

'multihop': Indicates the number of allowed IP hops between a PE and its BGP peer.

'as-override': If set, this parameter indicates whether ASN override is enabled, i.e., replace the ASN of the customer specified in the AS_PATH BGP attribute with the ASN identified in the 'local-as' attribute.

'allow-own-as': Is used in some topologies (e.g., hub-and-spoke) to allow the provider's ASN to be included in the AS_PATH BGP attribute received from a CE. Loops are prevented by setting 'allow-own-as' to a maximum number of provider's ASN occurrences. This parameter is set by default to '0' (that is, reject any AS_PATH attribute that includes the provider's ASN).

'prepend-global-as': When distinct ASNs are configured in the VPN node and network access levels, this parameter controls whether the ASN provided at the VPN node level is prepended to the AS_PATH attribute.

'send-default-route': Controls whether default routes can be advertised to the peer.

'site-of-origin': Is meant to uniquely identify the set of routes learned from a site via a particular CE/PE connection and is used to prevent routing loops (Section 7 of [RFC4364]). The Site of Origin attribute is encoded as a Route Origin Extended Community.

'ipv6-site-of-origin': Carries an IPv6 Address Specific BGP Extended Community that is used to indicate the Site of Origin for VRF information [RFC5701]. It is used to prevent routing loops.

'redistribute-connected': Controls whether the PE-CE link is advertised to other PEs.
'bgp-max-prefix': Controls the behavior when a prefix maximum is reached.

'max-prefix': Indicates the maximum number of BGP prefixes allowed in the BGP session. If the limit is reached, the action indicated in 'violate-action' will be followed.

'warning-threshold': A warning notification is triggered when this limit is reached.

'violate-action': Indicates which action to execute when the maximum number of BGP prefixes is reached. Examples of such actions are: send a warning message, discard extra paths from the peer, or restart the session.

'restart-timer': Indicates, in seconds, the time interval after which the BGP session will be reestablished.

'bgp-timers': Two timers can be captured in this container: (1) 'hold-time' which is the time interval that will be used for the HoldTimer (Section 4.2 of [RFC4271]) when establishing a BGP session. (2) 'keepalive' which is the time interval for the KeepAlive timer between a PE and a BGP peer (Section 4.4 of [RFC4271]). Both timers are expressed in seconds.

'authentication': The module adheres to the recommendations in Section 13.2 of [RFC4364] as it allows enabling TCP-AO [RFC5925] and accommodates the installed base that makes use of MD5. In addition, the module includes a provision for the use of IPsec.

This version of the L3NM assumes that TCP-AO specific parameters are preconfigured as part of the key-chain that is referenced in the L3NM. No assumption is made about how such a key-chain is pre-configured. However, the structure of the key-chain should cover data nodes beyond those in [RFC8177], mainly SendID andRecvID (Section 3.1 of [RFC5925]).

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

7.6.3.3. OSPF

OSPF can be configured to run as a routing protocol on the 'vpn-network-access'.
Figure 17: OSPF Routing Subtree Structure

The following data nodes are captured in Figure 17:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.
When the IPv4 or dual-stack address-family is requested, it is up to the implementation (e.g., network orchestrator) to decide whether OSPFv2 [RFC4577] or OSPFv3 [RFC6565] is used to announce IPv4 routes. Such decision will be typically reflected in the device configurations that will be derived to implement the L3VPN.

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

'metric': Associates a metric with OSPF routes.

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

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

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

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

7.6.3.4. IS-IS

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

The following IS-IS data nodes are supported:

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

'area-address': Indicates the IS-IS area address.

'level': Indicates the IS-IS level: Level 1, Level 2, or both.

'metric': Associates a metric with IS-IS routes.

'mode': Indicates the IS-IS interface mode type. It can be set to 'active' (that is, send or receive IS-IS protocol control packets) or 'passive' (that is, suppress the sending of IS-IS updates through the interface).
'authentication': Controls the authentication schemes to be enabled for the IS-IS instance. Both the specification of a key-chain [RFC8177] and the direct specification of key and authentication algorithm are supported.

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

7.6.3.5. RIP

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

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

Figure 19: RIP Subtree Structure

As shown in Figure 19, the following RIP data nodes are supported:
'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. This parameter is used to determine whether RIPv2 [RFC2453] and/or RIPng are to be enabled [RFC2080].

'timers': Indicates the following timers:

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

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

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

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

These timers are expressed in seconds.

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

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

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

7.6.3.6. VRRP

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

The following data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. Note that VRRP version 3 [RFC5798] supports both IPv4 and IPv6.

'vrrp-group': Is used to identify the VRRP group.

'backup-peer': Carries the IP address of the peer.

'virtual-ip-address': Includes virtual IP addresses for a single VRRP group.

'priority': Assigns the VRRP election priority for the backup virtual router.

'ping-reply': Controls whether ping requests can be replied to.

'status': Indicates the status of the VRRP instance.

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

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

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

Figure 21: IP Connection Subtree Structure (OAM)

The following OAM data nodes can be specified:

'session-type': Indicates which BFD flavor is used to set up the session (e.g., classic BFD [RFC5880], Seamless BFD [RFC7880]). By default, the BFD session is assumed to follow the behavior specified in [RFC5880].

'desired-min-tx-interval': Is the minimum interval, in microseconds, that a PE would like to use when transmitting BFD Control packets less any jitter applied.

'required-min-rx-interval': Is the minimum interval, in microseconds, between received BFD Control packets that a PE is capable of supporting, less any jitter applied by the sender.

'local-multiplier': The negotiated transmit interval, multiplied by this value, provides the detection time for the peer.

'holdtime': Is used to indicate the expected BFD holddown time, in
milliseconds. This value may be inherited from the service request (see Section 6.3.2.2.2 of [RFC8299]).

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

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

'status': Indicates the status of BFD.

7.6.5. Security

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

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

Figure 22: Security Subtree Structure
7.6.6. Services

7.6.6.1. Overview

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

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

Figure 23: Services Subtree Structure

The following data nodes are defined:

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

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

'mtu': Indicates the MTU at the service level.
'qos': Is used to define a set of QoS policies to apply on a given connection (refer to Section 7.6.6.2 for more details).

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

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

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

7.6.6.2. QoS

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

Layer 3: As shown in Figure 25, classification can be based on any IP header field or a combination thereof. Both IPv4 and IPv6 are supported.
### QoS Subtree Structure (L3)

Layer 4: As discussed in [I-D.ietf-opsawg-vpn-common], any layer 4
protocol can be indicated in the ‘protocol’ data node under ‘l3’ (Figure 25), but only TCP and UDP specific match criteria are elaborated in this version as these protocols are widely used in the context of VPN services. Augmentations can be considered in the future to add other Layer 4 specific data nodes, if needed.

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

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

```yang
++--rw qos {vpn-common:qos}? 
    ++--rw qos-classification-policy 
        ++--rw rule* [id] 
            ++--rw id string 
            ++--rw (match-type)? 
                ++--:(match-flow) 
                |        ++--rw (l3)? 
                |        |        ++-- ... 
                |        ++--rw (l4)? 
                |            ++--:(tcp) 
                |                ++--rw 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 
```
Figure 26: QoS Subtree Structure (L4)
Application match: Relies upon application-specific classification.

7.7. Multicast

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

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

Figure 27: Overall Multicast Subtree Structure

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

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

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

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

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

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

The same structure as the one depicted in Figure 30 is used when configuring multicast-related parameters at the VPN node level. When defined at the VPN node level (Figure 29), Internet Group Management Protocol (IGMP) [RFC1112][RFC2236][RFC3376], Multicast Listener Discovery (MLD) [RFC2710][RFC3810], and Protocol Independent Multicast (PIM) [RFC7761] parameters are applicable to all VPN network accesses of that VPN node unless corresponding nodes are overridden at the VPN network access level.
Multicast-related data nodes at the VPN network access level are shown in Figure 30. The values configured at the VPN network access level override the values configured for the corresponding data nodes in other levels.
8. L3NM YANG Module

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

<CODE BEGINS> file "ietf-l3vpn-ntw@2021-09-28.yang"
module ietf-l3vpn-ntw {
  yang-version 1.1;
  prefix l3nm;

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

  # L3NM YANG Model
  # Figure 30: Multicast Subtree Structure (VPN Network Access Level)
This YANG module defines a generic network-oriented model for the configuration of Layer 3 Virtual Private Networks.
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)";
}

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

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

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

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

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

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

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

identity local-link {
  base local-defined-next-hop;
  description "Treat traffic towards addresses within the specified next-hop prefix as though they are connected to a local link.";
}
identity l2-tunnel-type {
    description
        "Base identity for layer-2 tunnel selection under the VPN
         network access.";
}

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

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

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

/* Typedefs */

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

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

/* Groupings */
grouping vpn-instance-profile {
  description
      "Grouping for data nodes that may be factorized
      among many levels of the model. The grouping can
      be used to define generic profiles at the VPN service
      level and then referenced at the VPN node and VPN
      network access levels.";
  leaf local-as {
    if-feature "vpn-common:rtg-bgp";
    type inet:as-number;
    description
        "Provider’s Autonomous System (AS) number. Used if the
        customer requests BGP routing.";
  }
  uses vpn-common:route-distinguisher;
}
list address-family {
  key "address-family";
  description
      "Set of per-address family parameters.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
        "Indicates the address family (IPv4 and/or IPv6).";
  }
  container vpn-targets {
    description
        "Set of route targets to match for import and export routes
        to/from VRF.";
    uses vpn-common:vpn-route-targets;
  }
  list maximum-routes {
    key "protocol";
    description
        "Defines the maximum number of routes for the VRF.";
    leaf protocol {
      type identityref {
        base vpn-common:routing-protocol-type;
      }
      description
        "Indicates the routing protocol. ‘any’ value can
        be used to identify a limit that will apply for
        each active routing protocol.";
    }
    leaf maximum-routes {
      type uint32;
      description
"Indicates the maximum number of prefixes that the
VRF can accept for this address family and protocol.";
}
}
}
container multicast {
  if-feature "vpn-common:multicast";
  description
    "Global multicast parameters.";
  leaf tree-flavor {
    type identityref {
      base vpn-common:multicast-tree-type;
    }
    description
      "Type of the multicast tree to be used.";
  }
  container rp {
    description
      "Rendezvous Point (RP) parameters.";
    container rp-group-mappings {
      description
        "RP-to-group mappings parameters.";
      list rp-group-mapping {
        key "id";
        description
          "List of RP-to-group mappings.";
        leaf id {
          type uint16;
          description
            "Unique identifier for the mapping.";
        }
        container provider-managed {
          description
            "Parameters for a provider-managed RP.";
          leaf enabled {
            type boolean;
            default "false";
            description
              "Set to true if the Rendezvous Point (RP)
               must be a provider-managed node. Set to
               false if it is a customer-managed node.";
          }
          leaf rp-redundancy {
            type boolean;
            default "false";
            description
              "If set to true, it indicates that a redundancy
               mechanism for the RP is required.";
        }
      }
    }
  }
}
leaf optimal-traffic-delivery {
  type boolean;
  default "false";
  description "If set to true, the service provider (SP) must ensure that the traffic uses an optimal path. An SP may use Anycast RP or RP-tree-to-SPT switchover architectures.";
}

container anycast {
  when "../rp-redundancy = 'true' and ../optimal-traffic-delivery = 'true'" {
    description "Only applicable if both RP redundancy and delivery through optimal path are activated.";
  }
  description "PIM Anycast-RP parameters.";
  leaf local-address {
    type inet:ip-address;
    description "IP local address for PIM RP. Usually, it corresponds to the Router ID or the primary address.";
  }
  leaf-list rp-set-address {
    type inet:ip-address;
    description "Specifies the IP address of other RP routers that share the same RP IP address.";
  }
}

leaf rp-address {
  when "../provider-managed/enabled = 'false'" {
    description "Relevant when the RP is not provider-managed.";
  }
  type inet:ip-address;
  mandatory true;
  description "Defines the address of the RP. Used if the RP is customer-managed.";
}

container groups {

Barguil, et al. Expires 11 April 2022 [Page 66]
description
  "Multicast groups associated with the RP."
list group {
  key "id";
  description
    "List of multicast groups."
  leaf id {
    type uint16;
    description
      "Identifier for the group.";
  }
  choice group-format {
    mandatory true;
    description
      "Choice for multicast group format."
    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.";
      }
    }
  }
}
default "vpn-common:static-rp";
  description
    "Type of RP discovery used.";
}
container bsr-candidates {
  when "derived-from-or-self(../rp-discovery-type, "
    + "'vpn-common:bsr-rp')"
  description
    "Only applicable if discovery type is BSR-RP.";
}
  description
    "Container for the customer Bootstrap Router (BSR) 
    candidate’s addresses.";
  leaf-list bsr-candidate-address {
    type inet:ip-address;
    description
    " Specifies the address of candidate BSR.";
  }
}
}

container igmp {
  if-feature "vpn-common:igmp and vpn-common:ipv4"
  description
    "Includes IGMP-related parameters.";
  list static-group {
    key "group-addr";
    description
    "Multicast static source/group associated to the 
    IGMP session.";
    leaf group-addr {
      type rt-types:ipv4-multicast-group-address;
      description
      "Multicast group IPv4 address.";
    }
    leaf source-addr {
      type rt-types:ipv4-multicast-source-address;
      description
      "Multicast source IPv4 address.";
    }
  }
  leaf max-groups {
    type uint32;
    description
    "Indicates the maximum number of groups.";
  }
  leaf max-entries {
    type uint32;
  }

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

base vpn-common:mld-version;
}
default "vpn-common:mldv2";

description
"Indicates the MLD protocol version."

reference
"RFC 2710: Multicast Listener Discovery (MLD) for IPv6
RFC 3810: Multicast Listener Discovery Version 2 (MLDv2)
for IPv6"

} container pim {
if-feature "vpn-common:pim";

description
"Only applies when protocol type is PIM."

leaf hello-interval {

type rt-types:timer-value-seconds16;
default "30";

description
"PIM hello-messages interval. If set to
'infinity' or 'not-set', no periodic
Hello messages are sent."

reference
"RFC 7761: Protocol Independent Multicast - Sparse
Mode (PIM-SM): Protocol Specification (Revised),
Section 4.11"

} leaf dr-priority {

type uint32;
default "1";

description
"Indicates the preference in the Designated Router (DR)
election process. A larger value has a higher
priority over a smaller value."

reference
"RFC 7761: Protocol Independent Multicast - Sparse
Mode (PIM-SM): Protocol Specification (Revised),
Section 4.3.2"

} */ Main Blocks */
/* Main 13vpn-ntw */

container 13vpn-ntw {

description
"Main container for L3VPN services management.");
container vpn-profiles {
  description
    "Contains a set of valid VPN profiles to reference in the VPN service.";
  uses vpn-common:vpn-profile-cfg;
}
container vpn-services {
  description
    "Container for the VPN services.");
list vpn-service {
  key "vpn-id";
  description
    "List of VPN services.");
  uses vpn-common:vpn-description;
  leaf parent-service-id {
    type vpn-common:vpn-id;
    description
      "Pointer to the parent service, if any.
      A parent service can be an L3SM, a slice request, a VPN+ service, etc.";
  }
  leaf vpn-type {
    type identityref {
      base vpn-common:service-type;
    }
    description
      "Indicates the service type.";
  }
  leaf vpn-service-topology {
    type identityref {
      base vpn-common:vpn-topology;
    }
    default "vpn-common:any-to-any";
    description
      "VPN service topology.";
  }
  uses vpn-common:service-status;
  container vpn-instance-profiles {
    description
      "Container for a list of VPN instance profiles.");
  list vpn-instance-profile {
    key "profile-id";
    description
      "List of VPN instance profiles.";
    leaf profile-id {
      type string;
      description
    }
  }
"
"VPN instance profile identifier.";
)
leaf role {
  type identityref {
    base vpn-common:role;
  }
  default "vpn-common:any-to-any-role";
  description
    "Role of the VPN node in the VPN.";
}
uses vpn-instance-profile;
)
}
container underlay-transport { 
  description
    "Container for underlay transport.";
  uses vpn-common:underlay-transport;
}
container external-connectivity {
  if-feature "vpn-common:external-connectivity";
  description
    "Container for external connectivity.";
  choice profile {
    description
      "Choice for the external connectivity profile.";
    case profile {
      leaf profile-name {
        type leafref {
          path "/l3vpn-ntw/vpn-profiles"
          + "/valid-provider-identifiers"
          + "/external-connectivity-identifier/id";
        }
        description
          "Name of the service provider’s profile to be applied at the VPN service level.";
    }
  }
}
}
container vpn-nodes {
  description
    "Container for VPN nodes.";
list vpn-node {
  key "vpn-node-id";
  description
    "Includes a list of VPN nodes.";
  leaf vpn-node-id {
    type vpn-common:vpn-id;
description
    "An identifier of the VPN node.";
}
leaf description {
    type string;
    description
    "Textual description of the VPN node.";
}
leaf ne-id {
    type string;
    description
    "Unique identifier of the network element where the VPN node is deployed.";
}
leaf local-as {
    if-feature "vpn-common:rtg-bgp";
    type inet:as-number;
    description
    "Provider’s AS number in case the customer requests BGP routing.";
}
leaf router-id {
    type rt-types:router-id;
    description
    "A 32-bit number in the dotted-quad format that is used to uniquely identify a node within an autonomous system. This identifier is used for both IPv4 and IPv6.";
}
container active-vpn-instance-profiles {
    description
    "Container for active VPN instance profiles.";
    list vpn-instance-profile {
        key "profile-id";
        description
        "Includes a list of active VPN instance profiles.";
        leaf profile-id {
            type leafref {
                path "/l3vpn-ntw/vpn-services/vpn-service" + "/vpn-instance-profiles/vpn-instance-profile" + "/profile-id";
            }
            description
            "Node’s active VPN instance profile.";
            }
        list router-id {
            key "address-family";
            description
leaf address-family {
  type identityref {
    base vpn-common:address-family;
  }
  description
  "Indicates the address family for which the
  Router-ID applies.";
}

leaf router-id {
  type inet:ip-address;
  description
  "The router-id information can be an IPv4 or IPv6
  address. This can be used, for example, to
  configure an IPv6 address as a router-id
  when such capability is supported by underlay
  routers. In such case, the configured value
  overrides the generic one defined at the VPN
  node level.";
}
}

uses vpn-instance-profile;
}

container msdp {
  if-feature "msdp";
  description
  "Includes MSDP-related parameters.";
  leaf peer {
    type inet:ipv4-address;
    description
    "Indicates the IPv4 address of the MSDP peer.";
  }
  leaf local-address {
    type inet:ipv4-address;
    description
    "Indicates the IPv4 address of the local end.
    This local address must be configured on
    the node.";
  }
  uses vpn-common:service-status;
}

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

"Container for layer 2 encapsulation."

leaf type {
  type identityref {
    base vpn-common:encapsulation-type;
  }
  default "vpn-common:priority-tagged";
  description
    "Encapsulation type. By default, the type of
    the tagged interface is 'priority-tagged'.";
}

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

container priority-tagged {
  when "derived-from-or-self(../type, " + "'vpn-common:priority-tagged')" {
    description
      "Only applies when the type of the
      tagged interface is 'priority-tagged'.";
  }
  description
    "Priority tagged."
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
  }
}
default "vpn-common:c-vlan";

description
  "Tag type. By default, the tag type is 'c-vlan'.";
}

container qinq {
  when "derived-from-or-self(../type, "
    + "'vpn-common:qinq')" {
    description
      "Only applies when the type of the tagged interface is QinQ.";
  }
  description
    "Includes QinQ parameters.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:s-c-vlan";
    description
      "Tag type. By default, the tag type is 'c-s-vlan'.";
  }
  leaf svlan-id {
    type uint16;
    mandatory true;
    description
      "S-VLAN identifier.";
  }
  leaf cvlan-id {
    type uint16;
    mandatory true;
    description
      "C-VLAN identifier.";
  }
}

}  

}  

choice l2-service {
  description
    "The layer 2 connectivity service can be provided by indicating a pointer to an L2VPN or by specifying a layer 2 tunnel service."
  container l2-tunnel-service {
    description
      "Defines a layer 2 tunnel termination. It is only applicable when a tunnel is
required. The supported values are: pseudowire, VPLS, and VXLAN. Other values may be defined, if needed.

leaf type {
  type identityref {
    base l2-tunnel-type;
  }
  description
    "Selects the tunnel termination option for each vpn-network-access."
}

container pseudowire {
  when "derived-from-or-self(../type, "pseudowire")" {
    description
      "Only applies when the type of the layer 2 service type is pseudowire."
  }
  description
    "Includes pseudowire termination parameters."
  leaf vcid {
    type uint32;
    description
      "Indicates a PW or VC identifier."
  }
  leaf far-end {
    type union {
      type uint32;
      type inet:ip-address;
    }
    description
      "Neighbor reference."
    reference
      "RFC 8077: Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP), Section 6.1"
  }
}

container vpls {
  when "derived-from-or-self(../type, "vpls")" {
    description
      "Only applies when the type of the layer 2 service type is VPLS."
  }
  description
    "VPLS termination parameters."
  leaf vcid {

type uint32;
  description
    "VC Identifier."
};

leaf-list far-end {
  type union {
    type uint32;
    type inet:ip-address;
  }
  description
    "Neighbor reference."
}

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

case l2vpn {
  leaf l2vpn-id {
    type vpn-common:vpn-id;
  }
}
description
"Indicates the L2VPN service associated with an Integrated Routing and Bridging (IRB) interface."
}
}
leaf l2-termination-point {
    type string;
    description
    "Specifies a reference to a local layer 2 termination point such as a layer 2 sub-interface.";
}

leaf local-bridge-reference {
    type string;
    description
    "Specifies a local bridge reference to accommodate, for example, implementations that require internal bridging. A reference may be a local bridge domain.";
}

leaf bearer-reference {
    if-feature "vpn-common:bearer-reference";
    type string;
    description
    "This is an internal reference for the service provider to identify the bearer associated with this VPN.";
}

container lag-interface {
    if-feature "vpn-common:lag-interface";
    description
    "Container of LAG interface attributes configuration.";
    leaf lag-interface-id {
        type string;
        description
        "LAG interface identifier.";
    }
}

container member-link-list {
    description
    "Container of Member link list.";
    list member-link {
        key "name";
        description
        "Member link.";
        leaf name {

type string;
  description
      "Member link name."
};
)
}
)
}
)
)
)
)
}
)
}
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)

Barguil, et al. Expires 11 April 2022 [Page 81]
"Defines how addresses are allocated to the peer site.

If there is no value for the address allocation type, then IPv4 addressing is not enabled.";

choice allocation-type {
  description
  "Choice of the IPv4 address allocation.";
  case provider-dhcp {
    description
    "DHCP allocated addresses related parameters. IP addresses are allocated by DHCP that is operated by the provider";
    leaf dhcp-service-type {
      type enumeration {
        enum server {
          description
          "Local DHCP server.";
        }
        enum relay {
          description
          "Local DHCP relay. DHCP requests are relayed to a provider’s server.";
        }
      }
    }
    description
    "Indicates the type of DHCP service to be enabled on this access.";
  }
  choice service-type {
    description
    "Choice based on the DHCP service type.";
    case relay {
      description
      "Container for list of provider’s DHCP servers (i.e., dhcp-service-type is set to relay).";
      leaf-list server-ip-address {
        type inet:ipv4-address;
        description
        "IPv4 addresses of the provider’s DHCP server to use by the local DHCP relay.";
      }
    }
    case server {

    }

Barguil, et al. Expires 11 April 2022 [Page 82]
description
  "A choice about how addresses are assigned when a local DHCP server is enabled.";
choice address-assign {
  default "number";
  description
  "Choice for how IPv4 addresses are assigned.";
  case number {
    leaf number-of-dynamic-address {
      type uint16;
      default "1";
      description
      "Specifies the number of IP addresses to be assigned to the customer on this access.";
    }
  }
  case explicit {
    container customer-addresses {
      description
      "Container for customer addresses to be allocated using DHCP.";
      list address-pool {
        key "pool-id";
        description
        "Describes IP addresses to be allocated by DHCP.

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

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

Barguil, et al. Expires 11 April 2022 [Page 83]
leaf end-address {
  type inet:ipv4-address;
  description "Indicates the last address in the pool.";
}

case dhcp-relay {
  description "DHCP relay is provided by the operator.";
  container customer-dhcp-servers {
    description "Container for a list of customer’s DHCP servers.";
    leaf-list server-ip-address {
      type inet:ipv4-address;
      description "IPv4 addresses of the customer’s DHCP server.";
    }
  }
}

case static-addresses {
  description "Lists the IPv4 addresses that are used.";
  leaf primary-address {
    type leafref {
      path "../address/address-id";
    }
    description "Primary address of the connection.";
  }
  list address {
    key "address-id";
    description "Lists the IPv4 addresses that are used.";
    leaf address-id {
      type string;
    }
  }
}
description
"An identifier of the static IPv4 address."
}
leaf customer-address {
  type inet:ipv4-address;
  description
  "IPv4 address at the customer side.";
}

container ipv6 {
  if-feature "vpn-common:ipv6";
  description
  "IPv6-specific parameters.";
  leaf local-address {
    type inet:ipv6-address;
    description
    "IPv6 address of the provider side.";
  }
  leaf prefix-length {
    type uint8 {
      range "0..128";
    }
    description
    "Subnet prefix length expressed in bits.
    It is applied to both local and customer addresses.";
  }
  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 addressing is disabled.";
  }
  choice allocation-type {
    description
    "A choice based on the IPv6 allocation type.";
    container provider-dhcp {
      when "derived-from-or-self(../address-allocation-type, 'provider-dhcp')"
      + "cation-type, 'provider-dhcp'") "
      + "or derived-from-or-self(../address-allocation-type, 'provider-dhcp')"
    }
  }
}
description "DHCPv6 allocated addresses related parameters.";
leaf dhcp-service-type {
type enumeration {
enum server {
description "Local DHCPv6 server.";
}
enum relay {
description "DHCPv6 relay.";
}
}
description "Indicates the type of the DHCPv6 service to be enabled on this access.";
}
choice service-type {
description "Choice based on the DHCPv6 service type.";
case relay {
leaf-list server-ip-address {
type inet:ipv6-address;
description "IPv6 addresses of the provider’s DHCPv6 server.";
}
}
case server {
choice address-assign {
default "number";
description "Choice about how IPv6 prefixes are assigned by the DHCPv6 server.";
case number {
leaf number-of-dynamic-address {
type uint16;
default "1";
description "Describes the number of IPv6 prefixes that are allocated to
the customer on this access.
}
}
case explicit {
    container customer-addresses {
        description "Container for customer IPv6 addresses allocated by DHCPv6.";
        list address-pool {
            key "pool-id";
            description "Describes IPv6 addresses allocated by DHCPv6.

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

            When both start-address and end-address are specified, it implies a range inclusive of both addresses.";
            leaf pool-id {
                type string;
                description "Pool identifier for the address range from identified by start-address and end-address.";
            }
            leaf start-address {
                type inet:ipv6-address;
                mandatory true;
                description "Indicates the first address.";
            }
            leaf end-address {
                type inet:ipv6-address;
                description "Indicates the last address.";
            }
        }
    }
}
}
}
}
case dhcp-relay {
description
"DHCPv6 relay provided by the operator.";
container customer-dhcp-servers {
    description
    "Container for a list of customer DHCP servers.";
    leaf-list server-ip-address {
        type inet:ipv6-address;
        description
        "Contains the IP addresses of the customer DHCPv6 server.";
    }
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
key "id";
description
"List of routing protocols used on
the CE/PE link. This list can be augmented.";
leaf id {
  type string;
description
  "Unique identifier for routing protocol.";
}
leaf type {
  type identityref {
    base vpn-common:routing-protocol-type;
  }
description
  "Type of routing protocol.";
}
list routing-profiles {
  key "id";
description
  "Routing profiles.";
  leaf id {
    type leafref {
      path "/l3vpn-ntw/vpn-profiles" + "/valid-provider-identifiers" + "/routing-profile-identifier/id";
    }
description
    "Routing profile to be used.";
  }
  leaf type {
    type identityref {
      base vpn-common:ie-type;
    }
description
    "Import, export, or both.";
  }
}
container static {
  when "derived-from-or-self(../type, " + "/vpn-common:static-routing")" {
    description
    "Only applies when protocol is static.";
  }
description
  "Configuration specific to static routing.";
  container cascaded-lan-prefixes {
    description
    "LAN prefixes from the customer.";
  }
}

Barguil, et al. Expires 11 April 2022 [Page 89]
list ipv4-lan-prefixes {
  if-feature "vpn-common:ipv4";
  key "lan next-hop";
  description
      "List of LAN prefixes for the site.";
  leaf lan {
    type inet:ipv4-prefix;
    description
      "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
      "Internal tag to be used in VPN policies.";
  }
  leaf next-hop {
    type union {
      type inet:ip-address;
      type predefined-next-hop;
    }
    description
      "The next-hop that is to be used for the static route. This may be specified as an IP address or a pre-defined next-hop type (e.g., discard or local-link).";
  }
  leaf bfd-enable {
    if-feature "vpn-common:bfd";
    type boolean;
    description
      "Enables BFD.";
  }
  leaf metric {
    type uint32;
    description
      "Indicates the metric associated with the static route.";
  }
  leaf preference {
    type uint32;
    description
      "Indicates the preference of the static routes.";
  }
  uses vpn-common:service-status;
}
list ipv6-lan-prefixes {
    if-feature "vpn-common:ipv6";
    key "lan next-hop";
    description
        "List of LAN prefixes for the site.";
    leaf lan {
        type inet:ipv6-prefix;
        description
            "LAN prefixes.";
    }
    leaf lan-tag {
        type string;
        description
            "Internal tag to be used in VPN policies.";
    }
    leaf next-hop {
        type union {
            type inet:ip-address;
            type predefined-next-hop;
        }
        description
            "The next-hop that is to be used for the static route. This may be specified as an IP address or a pre-defined next-hop type (e.g., discard or local-link).";
    }
    leaf bfd-enable {
        if-feature "vpn-common:bfd";
        type boolean;
        description
            "Enables BFD.";
    }
    leaf metric {
        type uint32;
        description
            "Indicates the metric associated with the static route.";
    }
    leaf preference {
        type uint32;
        description
            "Indicates the preference associated with the static route.";
    }
    uses vpn-common:service-status;
}

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

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

leaf-list neighbor {
  type inet:ip-address;
  description "IP address(es) of the BGP neighbor. IPv4 and IPv6 neighbors may be indicated if two sessions will be used for IPv4 and IPv6.";
}

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

leaf as-override {
  type boolean;
  default "false";
  description "Defines whether ASN override is enabled, i.e., replace the ASN of the customer specified in the AS_Path attribute with the local ASN.";
}

leaf allow-own-as {
  type uint8;
  default "0";
  description "Specifies the number of occurrences of the provider’s ASN that can occur within the AS_PATH before it is rejected.";
}

leaf prepend-global-as {
  type boolean;
  default "false";
  description "In some situations, the ASN that is provided at the VPN node level may be distinct from the one configured at the VPN network access level. When such ASNs are provided, they are both prepended to the BGP route updates for this access. To disable that
behavior, the prepend-global-as
must be set to 'false'. In such a case,
the ASN that is provided at
the VPN node level is not prepended to
the BGP route updates for this access.
}

leaf send-default-route {
  type boolean;
  default "false";
  description
    "Defines whether default routes can be
    advertised to its peer. If set, the
default routes are advertised to its
peer.";
}

leaf site-of-origin {
  when "/.address-family = 'vpn-common:ipv4' or "
    + "'vpn-common:dual-stack'" {
    description
      "Only applies if IPv4 is activated.";
  }
  type rt-types:route-origin;
  description
    "The Site of Origin attribute is encoded as
a Route Origin Extended Community. It is
meant to uniquely identify the set of routes
learned from a site via a particular CE/PE
connection and is used to prevent routing
loops.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private
Networks (VPNs), Section 7";
}

leaf ipv6-site-of-origin {
  when "/.address-family = 'vpn-common:ipv6' or "
    + "'vpn-common:dual-stack'" {
    description
      "Only applies if IPv6 is activated.";
  }
  type rt-types:ipv6-route-origin;
  description
    "IPv6 Route Origins are IPv6 Address Specific
BGP Extended that are meant to the Site of
Origin for VRF information.";
  reference
    "RFC 5701: IPv6 Address Specific BGP Extended
Community Attribute";
}
list redistribute-connected {
  key "address-family";
  description "Indicates the per-AF policy to follow for connected routes.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description "Indicates the address family.";
  }
  leaf enable {
    type boolean;
    description "Enables to redistribute connected routes.";
  }
}

container bgp-max-prefix {
  description "Controls the behavior when a prefix maximum is reached.";
  leaf max-prefix {
    type uint32;
    default "5000";
    description "Indicates the maximum number of BGP prefixes allowed in the BGP session.
    It allows control of how many prefixes can be received from a neighbor.
    If the limit is exceeded, the action indicated in violate-action will be followed.";
    reference "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 8.2.2";
  }
  leaf warning-threshold {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    units "percent";
    default "75";
    description
  }
}
"When this value is reached, a warning notification will be triggered."
}
leaf violate-action {
  type enumeration {
    enum warning {
      description "Only a warning message is sent to the peer when the limit is exceeded.";
    }
    enum discard-extra-paths {
      description "Discards extra paths when the limit is exceeded.";
    }
    enum restart {
      description "The BGP session restarts after a time interval.";
    }
  }
  description "BGP neighbor max-prefix violate action.";
}
leaf restart-timer {
  type uint32;
  units "seconds";
  description "Time interval after which the BGP session will be reestablished.";
}
}
container bgp-timers {
  description "Includes two BGP timers that can be customized when building a VPN service with BGP used as CE-PE routing protocol.";
  leaf keepalive {
    type uint16 {
      range "0..21845";
    }
    units "seconds";
    default "30";
    description "This timer indicates the KEEPALIVE
messages’ frequency between a PE and a BGP peer.

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

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

reference
"RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.4";

leaf hold-time {
  type uint16 {
    range "0 | 3..65535";
  } units "seconds";
  default "90";
  description
"It indicates the maximum number of seconds that may elapse between the receipt of successive KEEPALIVE and/or UPDATE messages from the peer. The Hold Time must be either zero or at least three seconds.";
reference
"RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.2";
}

container authentication {
  description
"Container for BGP authentication parameters between a PE and a CE."
leaf enable {
  type boolean;
  default "false";
  description
"Enables or disables authentication."
}
container keying-material {
  when "./.enable = ’true’";
  description
"Container for describing how a BGP routing session is to be secured between a PE and a CE.";
choice option {
  description
  "Choice of authentication options.";
  case ao {
    description
    "Uses TCP-Authentication Option (TCP-AO).";
    reference
    "RFC 5925: The TCP Authentication Option.";
    leaf enable-ao {
      type boolean;
      description
      "Enables TCP-AO.";
    }
    leaf ao-keychain {
      type key-chain:key-chain-ref;
      description
      "Reference to the TCP-AO key chain.";
      reference
      "RFC 8177: YANG Key Chain";
    }
  }
  case md5 {
    description
    "Uses MD5 to secure the session.";
    reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 13.2";
    leaf md5-keychain {
      type key-chain:key-chain-ref;
      description
      "Reference to the MD5 key chain.";
      reference
      "RFC 8177: YANG Key Chain";
    }
  }
  case explicit {
    leaf key-id {
      type uint32;
      description
      "Key Identifier.";
    }
    leaf key {
      type string;
      description
      "BGP authentication key.";
    }
  }
}
This model only supports the subset of keys that are representable as ASCII strings.

leaf crypto-algorithm {
  type identityref {
    base key-chain:crypto-algorithm;
  }
  description
    "Indicates the cryptographic algorithm associated with the key."
}

case ipsec {
  description
    "Specifies a reference to an IKE Security Association (SA)."
  leaf sa {
    type string;
    description
      "Indicates the administrator-assigned name of the SA."
  }
}

uses vpn-common:service-status;

container ospf {
  when "derived-from-or-self(../type, " + "/vpn-common:ospf-routing")"
    description
      "Only applies when protocol is OSPF."
  description
    "OSPF-specific configuration."
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both are to be activated."
  }
  leaf area-id {
    type yang:dotted-quad;
    mandatory true;
description
  "Area ID."
reference
  "RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs), Section 4.2.3
RFC 6565: OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol, Section 4.2";
}
leaf metric {
type uint16;
default "1";
description
  "Metric of the PE-CE link. It is used in the routing state calculation and path selection."
}
container sham-links {
  if-feature "vpn-common:rtg-ospf-sham-link";
description
  "List of sham links."
reference
  "RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs), Section 4.2.7
RFC 6565: OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol, Section 5";
list sham-link {
  key "target-site";
description
  "Creates a sham link with another site."
leaf target-site {
type string;
description
  "Target site for the sham link connection. The site is referred to by its identifier."
}
leaf metric {
type uint16;
default "1";
description
  "Metric of the sham link. It is used in the routing state calculation and path
selection. The default value is set to 1."
reference
"RFC 4577: OSPF as the Provider/Customer
Edge Protocol for BGP/MPLS IP
Virtual Private Networks
(VPNs), Section 4.2.7.3
RFC 6565: OSPFv3 as a Provider Edge to
Customer Edge (PE-CE) Routing
Protocol, Section 5.2"

leaf max-lsa {
  type uint32 {
    range "1..4294967294"
  }
  description
  "Maximum number of allowed LSAs OSPF."
}

container authentication {
  description
  "Authentication configuration."
  leaf enable {
    type boolean;
    default "false"
    description
    "Enables or disables authentication."
  }
  container keying-material {
    when ".//enable = 'true'"
    description
    "Container for describing how an OSPF
    session is to be secured between a CE
    and a PE."
    choice option {
      description
      "Options for OSPF authentication."
      case auth-key-chain {
        leaf key-chain {
          type key-chain:key-chain-ref
          description
          "key-chain name."
        }
      }
      case auth-key-explicit {
        leaf key-id {
          type uint32
        }
      }
    }
  }
}
description
  "Key identifier.";
}
leaf key {
    type string;
    description
      "OSPF authentication key. This model only supports the subset
      of keys that are representable as ASCII strings.";
}
leaf crypto-algorithm {
    type identityref {
      base key-chain:crypto-algorithm;
    }
    description
      "Indicates the cryptographic algorithm associated with the key.";
}
}
}
case ipsec {
    leaf sa {
      type string;
      description
        "Indicates the administrator-assigned name of the SA.";
      reference
        "RFC 4552: Authentication/
         Confidentiality for OSPFv3";
    }
}
}
}
uses vpn-common:service-status;
}
container isis {
  when "derived-from-or-self(../type, "
     + ")/type, "
     + "/type, "
     + "/type, "
     + "/type, "
     + "" vpn-common:isis-routing')" "
    description
      "Only applies when protocol is IS-IS.";
  }
  description
    "IS-IS specific configuration.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
leaf area-address {
  type area-address;
  mandatory true;
  description
    "Area address.";
}
leaf level {
  type identityref {
    base vpn-common:isis-level;
  }
  description
    "Can be level-1, level-2, or level-1-2.";
}
leaf metric {
  type uint16;
  default "1";
  description
    "Metric of the PE-CE link. It is used
    in the routing state calculation and
    path selection.";
}
leaf mode {
  type enumeration {
    enum active {
      description
        "Interface sends or receives IS-IS
        protocol control packets.";
    }
    enum passive {
      description
        "Suppresses the sending of IS-IS
        updates through the specified
        interface.";
    }
  }
  default "active";
  description
    "IS-IS interface mode type.";
}
container authentication {
  description
    "Authentication configuration.";
  leaf enable {
    }
type boolean;
default "false";
description
  "Enables or disables authentication.";
}
container keying-material {
  when ".../enable = 'true'";
  description
  "Container for describing how an IS-IS 
    session is to be secured between a CE
    and a PE.";
  choice option {
    description
      "Options for IS-IS authentication.";
    case auth-key-chain {
      leaf key-chain {
        type key-chain:key-chain-ref;
        description
          "key-chain name.";
      }
    }
    case auth-key-explicit {
      leaf key-id {
        type uint32;
        description
          "Key Identifier.";
      }
      leaf key {
        type string;
        description
          "IS-IS authentication key. 
            This model only supports the subset 
            of keys that are representable as 
            ASCII strings.";
      }
      leaf crypto-algorithm {
        type identityref {
          base key-chain:crypto-algorithm;
        }
        description
          "Indicates the cryptographic algorithm 
            associated with the key.";
      }
    }
  }
}
uses vpn-common:service-status;
container rip {
  when "derived-from-or-self(../type, "
    + "vpn-common:rip-routing")"
    description
    "Only applies when the protocol is RIP. For IPv4, the model assumes that RIP version 2 is used.";
}
description
"Configuration specific to RIP routing."
leaf address-family {
  type identityref {
    base vpn-common:address-family;
  }
description
  "Indicates whether IPv4, IPv6, or both address families are to be activated.";
}
container timers {
description
  "Indicates the RIP timers."
reference
  "RFC 2453: RIP Version 2"
leaf update-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "30";
  description
    "Indicates the RIP update time. That is, the amount of time for which RIP updates are sent.";
}
leaf invalid-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "180";
  description
    "Is the interval before a route is declared invalid after no updates are received. This value is at least three times the value for the update-interval argument.";
}
leaf holddown-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "180";
  description
    "Specifies the interval before better routes are released.";
}
leaf flush-interval {
  type uint16 {
    range "1..32767";
  }
  units "seconds";
  default "240";
  description
    "Indicates the RIP flush timer. That is, the amount of time that must elapse before a route is removed from the routing table.";
}
leaf default-metric {
  type uint8 {
    range "0..16";
  }
  default "1";
  description
    "Sets the default metric.";
}
container authentication {
  description
    "Authentication configuration.";
  leaf enable {
    type boolean;
    default "false";
    description
      "Enables or disables authentication.";
  }
  container keying-material {
    when "/../enable = 'true'";
    description
      "Container for describing how a RIP session is to be secured between a CE and a PE.";
    choice option {
      description

"Specifies the authentication scheme."

case auth-key-chain {
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
    "key-chain name.";
  }
}
case auth-key-explicit {
  leaf key {
    type string;
    description
    "RIP authentication key.
    This model only supports the subset
    of keys that are representable as
    ASCII strings.";
  }
  leaf crypto-algorithm {
    type identityref {
      base key-chain:crypto-algorithm;
    }
    description
    "Indicates the cryptographic algorithm
    associated with the key.";
  }
}
}

uses vpn-common:service-status;

container vrrp {
  when "derived-from-or-self(../type, "
  + "vpn-common:vrrp-routing")" {
    description
    "Only applies when protocol is VRRP.";
  }
  description
  "Configuration specific to VRRP.";
  reference
  "RFC 5798: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
    "Indicates whether IPv4, IPv6, or both
address families are to be enabled."
}
leaf vrrp-group {
    type uint8 {
        range "1..255";
    }
    description
        "Includes the VRRP group identifier.";
}
leaf backup-peer {
    type inet:ip-address;
    description
        "Indicates the IP address of the peer.";
}
leaf-list virtual-ip-address {
    type inet:ip-address;
    description
        "Virtual IP addresses for a single VRRP group.";
    reference
        "RFC 5798: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6, Sections 1.2 and 1.3";
}
leaf priority {
    type uint8 {
        range "1..254";
    }
    default "100";
    description
        "Sets the local priority of the VRRP speaker.";
}
leaf ping-reply {
    type boolean;
    default "false";
    description
        "Controls whether the VRRP speaker should answer to ping requests.";
}
uses vpn-common:service-status;
}
}
container oam {
    description
        "Defines the Operations, Administration, and Maintenance (OAM) mechanisms used."
BFD is set as a fault detection mechanism, but other mechanisms can be defined in the future.

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

    The detection interval for the receiving
BFD peer is calculated by multiplying the value of the negotiated transmission interval by the received detection multiplier value.

reference
"RFC 5880: Bidirectional Forwarding Detection (BFD), Section 6.8.7";

} leaf holdtime {
    type uint32;
    units "milliseconds";
    description
    "Expected BFD holdtime.

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

    If the provider doesn’t allow the customer to use this function, the fixed-value will not be set."

reference
"RFC 5880: Bidirectional Forwarding Detection (BFD), Section 6.8.18";

} leaf profile {
    type leafref {
        path "/l3vpn-ntw/vpn-profiles" + "/valid-provider-identifiers" + "/bfd-profile-identifier/id";
    }
    description
    "Well-known service provider profile name.

    The provider can propose some profiles to the customer, depending on the service level the customer wants to achieve."

} container authentication {
    presence "Enables BFD authentication";
    description
    "Parameters for BFD authentication."
    leaf key-chain {
        type key-chain:key-chain-ref;
        description
        "Name of the key-chain.";
    }
leaf meticulous {
    type boolean;
    description
        "Enables meticulous mode."
    reference
        "RFC 5880: Bidirectional Forwarding
         Detection (BFD), Section 6.7";
}

uses vpn-common:service-status;

container security {
    description
        "Site-specific security parameters.";
    container encryption {
        if-feature "vpn-common:encryption";
        description
            "Container for CE-PE security encryption.";
    }
    leaf enabled {
        type boolean;
        default "false";
        description
            "If true, traffic encryption on the
             connection is required. Otherwise, it
             is disabled.";
    }
    leaf layer {
        when "/enabled = 'true'" {
            description
                "It is included only when encryption
                 is enabled.";
        }
    }
    type enumeration {
        enum layer2 {
            description
                "Encryption occurs at Layer 2.";
        }
        enum layer3 {
            description
                "Encryption occurs at Layer 3.
                 For example, IPsec may be used when
                 a customer requests Layer 3
                 encryption.";
        }
    }
    description
        "Indicates the layer on which encryption
is applied.
}
}

container encryption-profile {
    when "../encryption/enabled = 'true'" {
        description
            "Indicates the layer on which encryption is enabled."
    }
    description
        "Container for encryption profile."
    choice profile {
        description
            "Choice for the encryption profile."
        case provider-profile {
            leaf profile-name {
                type leafref {
                    path "/l3vpn-ntw/vpn-profiles" + "/valid-provider-identifiers" + "/encryption-profile-identifier/id";
                }
                description
                    "Name of the service provider’s profile to be applied."
        }
        case customer-profile {
            leaf customer-key-chain {
                type key-chain:key-chain-ref;
                description
                    "Customer-supplied key chain."
        }
    }
    }
}

container service {
    description
        "Service parameters of the attachment."
    leaf inbound-bandwidth {
        if-feature "vpn-common:inbound-bw";
        type uint64;
        units "bps";
        description
            "From the customer site’s perspective, the service inbound bandwidth of the connection or download bandwidth from the SP to the site. Note that the L3SM uses 'input-
leaf outbound-bandwidth {
  if-feature "vpn-common:outbound-bw";
  type uint64;
  units "bps";
  description
    "From the customer site’s perspective, the service outbound bandwidth of the connection or upload bandwidth from the site to the SP. Note that the L3SM uses ‘output-bandwidth’ to refer to the same concept."
}

leaf mtu {
  type uint32;
  units "bytes";
  description
    "MTU at service level. If the service is IP, it refers to the IP MTU. If Carriers’ Carriers (CsC) is enabled, the requested MTU will refer to the MPLS maximum labeled packet size and not to the IP MTU."
}

container qos {
  if-feature "vpn-common:qos";
  description
    "QoS configuration."
  container qos-classification-policy {
    description
      "Configuration of the traffic classification policy."
    uses vpn-common:qos-classification-policy;
  }
  container qos-action {
    description
      "List of QoS action policies."
    list rule {
      key "id";
      description
        "List of QoS actions."
      leaf id {
        type string;
        description
          "An identifier of the QoS action rule."
      }
      leaf target-class-id {
        type string;
      }
    }
  }
}


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

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

"Time synchronization may be needed in some VPNs such as infrastructure and Management VPNs. This container includes parameters to enable NTP service."

reference

leaf broadcast {
type enumeration {
  enum client {
    description
    "The VPN node will listen to NTP broadcast messages on this VPN network access.";
  }
  enum server {
    description
    "The VPN node will behave as a broadcast server.";
  }
}
description
"Indicates NTP broadcast mode to use for the VPN network access.";
}

container auth-profile {
description
"Pointer to a local profile.";
leaf profile-id {
type string;
description
"A pointer to a local authentication profile on the VPN node is provided.";
}
}
uses vpn-common:service-status;
}

container multicast {
if-feature "vpn-common:multicast";
description
"Multicast parameters for the network access.";
leaf access-type {
type enumeration {
  enum receiver-only {
    description
    "The peer site only has receivers.";
  }
}
}
enum source-only {
    description
    "The peer site only has sources.";
}
enum source-receiver {
    description
    "The peer site has both sources and receivers.";
}
default "source-receiver";
description
    "Type of multicast site.";
}
leaf address-family {
    type identityref {
        base vpn-common:address-family;
    }
description
    "Indicates the address family.";
}
leaf protocol-type {
    type enumeration {
        enum host {
            description
            "Hosts are directly connected to the provider network.
            Host protocols such as IGMP or MLD are required.";
        }
        enum router {
            description
            "Hosts are behind a customer router. PIM will be implemented.";
        }
        enum both {
            description
            "Some hosts are behind a customer router, and some others are directly connected to the provider network. Both host and routing protocols must be used. Typically, IGMP and PIM will be implemented.";
        }
    }
    default "both";
description
  "Multicast protocol type to be used with the customer site."
}

leaf remote-source {
  type boolean;
  default "false";
  description
  "A remote multicast source is a source that is not on the same subnet as the
  vpn-network-access. When set to 'true', the multicast traffic from a remote source is accepted.";
}

container igmp {
  when "../protocol-type = 'host' and "
    + "../address-family = 'vpn-common:ipv4' or "
    + "'vpn-common:dual-stack'";
  if-feature "vpn-common:igmp";
  description
  "Includes IGMP-related parameters.";
  list static-group {
    key "group-addr";
    description
    "Multicast static source/group associated to IGMP session";
    leaf group-addr {
      type rt-types:ipv4-multicast-group-address;
      description
      "Multicast group IPv4 address.";
    }
    leaf source-addr {
      type rt-types:ipv4-multicast-source-address;
      description
      "Multicast source IPv4 address.";
    }
  }
  leaf max-groups {
    type uint32;
    description
    "Indicates the maximum number of groups.";
  }
  leaf max-entries {
    type uint32;
    description
    "Indicates the maximum number of IGMP entries.";
  }
}
leaf max-group-sources {
    type uint32;
    description
    "The maximum number of group sources."
}

leaf version {
    type identityref {
        base vpn-common:igmp-version;
    }
    default "vpn-common:igmpv2";
    description
    "Version of the IGMP."
}

uses vpn-common:service-status;

container mld {
    when ".//protocol-type = 'host' and 
    + "./address-family = 'vpn-common:ipv6' or 
    + "'vpn-common:dual-stack'";
    if-feature "vpn-common:mld";
    description
    "Includes MLD-related parameters."
    list static-group {
        key "group-addr";
        description
        "Multicast static source/group associated to
        the MLD session";
        leaf group-addr {
            type rt-types:ipv6-multicast-group-address;
            description
            "Multicast group IPv6 address."
        }
        leaf source-addr {
            type rt-types:ipv6-multicast-source-address;
            description
            "Multicast source IPv6 address."
        }
    }
    leaf max-groups {
        type uint32;
        description
        "Indicates the maximum number of groups."
    }
    leaf max-entries {
        type uint32;
        description
        "Indicates the maximum number of MLD
        entries."
    }
}
leaf max-group-sources {
  type uint32;
  description
    "The maximum number of group sources."
}
leaf version {
  type identityref {
    base vpn-common:mld-version;
  }
  default "vpn-common:mldv2";
  description
    "Version of the MLD protocol."
}
uses vpn-common:service-status;

container pim {
  when "../protocol-type = 'router';"
  if-feature "vpn-common:pim";
  description
    "Only applies when protocol type is PIM.";
  leaf hello-interval {
    type rt-types:timer-value-seconds16;
    default "30";
    description
      "PIM hello-messages interval. If set to
       'infinity' or 'not-set', no periodic
       Hello messages are sent.";
    reference
      "RFC 7761: Protocol Independent Multicast -
       Sparse Mode (PIM-SM): Protocol
       Specification (Revised),
       Section 4.11";
  }
  leaf dr-priority {
    type uint32;
    default "1";
    description
      "Indicates the preference in the DR election
       process. A larger value has a higher
       priority over a smaller value.";
    reference
      "RFC 7761: Protocol Independent Multicast -
       Sparse Mode (PIM-SM): Protocol
       Specification (Revised),
       Section 4.3.2";
  }
}
uses vpn-common:service-status;
9. 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 [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) 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-profiles': This container includes a set of sensitive data that influence how the L3VPN service is delivered. For example, an attacker who has access to these data nodes may be able to manipulate routing policies, QoS policies, or encryption properties. These data nodes are defined with "nacm:default-deny-write" tagging [I-D.ietf-opsawg-vpn-common].

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

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

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

* 'keying-material': An attacker can retrieve the cryptographic keys protecting the underlying VPN service (CE-PE routing, in particular). These keys could be used to inject spoofed routing advertisements.

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

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

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

10. IANA Considerations

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

Barguil, et al. Expires 11 April 2022 [Page 122]
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
maintained by IANA: N
prefix: l3nm
reference: RFC XXXX

11. References

11.1. Normative References


11.2. Informative References

[I-D.evenwu-opsawg-yang-composed-vpn]

[I-D.ietf-bess-evpn-prefix-advertisement]

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", Work in

[I-D.ietf-pim-yang]

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

[I-D.ietf-teas-enhanced-vpn]

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

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

[IEEE802.1AX]


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 a service that meets the SLA requirements.

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

![Diagram](image)

Figure 31: Mobile Backhaul Example

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

First: Create the 4G VPN service (Figure 32).
POST: /restconf/data/ietf-l3vpn-ntw:l3vpn-ntw/vpn-services
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "4G",
        "customer-name": "mycustomer",
        "vpn-service-topology": "custom",
        "vpn-description": "VPN to deploy 4G services",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "simple-profile",
              "local-as": 65550,
              "rd": "0:65550:1",
              "address-family": [
                {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "vpn-target": [
                    {
                      "id": 1,
                      "route-targets": [
                        { "route-target": "0:65550:1" }
                      ],
                      "route-target-type": "both"
                    }
                  ]
                }
              ]
            }
          ]
        }
      }
    ]
  }
}

Figure 32: Create VPN Service

Second: Create a VPN node as depicted in Figure 33. In this type of service, the VPN node is equivalent to the VRF configured in the physical device (‘ne-id’=198.51.100.1).
Figure 33: Create VPN Node

Finally, two VPN network accesses are created using the same physical port (‘interface-id’=1/1/1). Each ‘vpn-network-access’ has a particular VLAN (1,2) to differentiate the traffic between: Sync and data (Figure 34).

Figure 34: Create VPN Network Accesses

-------------- NOTE: ‘\’ line wrapping per RFC 8792 --------------
"admin-status": {
    "status": "ietf-vpn-common:admin-up"
},
"connection": {
    "encapsulation": {
        "type": "ietf-vpn-common:dot1q",
        "dot1q": {
            "cvlan-id": 1
        }
    }
},
"ip-connection": {
    "ipv4": {
        "local-address": "192.0.2.1",
        "prefix-length": 30,
        "address-allocation-type": "static-address",
        "static-addresses": {
            "primary-address": "1",
            "address": [
                {
                    "address-id": "1",
                    "customer-address": "192.0.2.2"
                }
            ]
        }
    },
    "ipv6": {
        "local-address": "2001:db8::1",
        "prefix-length": 64,
        "address-allocation-type": "static-address",
        "primary-address": "1",
        "address": [
            {
                "address-id": "1",
                "customer-address": "2001:db8::2"
            }
        ]
    }
},
"routing-protocols": {
    "routing-protocol": [
        {
            "id": "1",
            "type": "ietf-vpn-common:direct"
        }
    ]
}

{
  "id": "1/1/1.2",
  "interface-id": "1/1/1",
  "description": "Interface DATA to eNODE-B",
  "vpn-network-access-type": "ietf-vpn-common:point-to-point",
  "vpn-instance-profile": "simple-profile",
  "status": {
    "admin-status": {
      "status": "ietf-vpn-common:admin-up"
    }
  },
  "connection": {
    "encapsulation": {
      "type": "ietf-vpn-common:dot1q",
      "dot1q": {
        "cvlan-id": 2
      }
    }
  },
  "ip-connection": {
    "ipv4": {
      "local-address": "192.0.2.1",
      "prefix-length": 30,
      "address-allocation-type": "static-address",
      "static-addresses": {
        "primary-address": "1",
        "address": [
          {
            "address-id": "1",
            "customer-address": "192.0.2.2"
          }
        ]
      }
    },
    "ipv6": {
      "local-address": "2001:db8::1",
      "prefix-length": 64,
      "address-allocation-type": "static-address",
      "primary-address": "1",
      "address": [
        {
          "address-id": "1",
          "customer-address": "2001:db8::2"
        }
      ]
    }
  }
}
"routing-protocols": {
  "routing-protocol": [
    {
      "id": "1",
      "type": "ietf-vpn-common:direct"
    }
  ]
}

Figure 34: Create VPN Network Access

A.2. Loopback Interface

An example of loopback interface is depicted in Figure 35.

{ "ietf-l3vpn-ntw:vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "vpn-access-loopback",
      "interface-id": "Loopback1",
      "description": "An example of loopback interface.",
      "vpn-network-access-type": "ietf-vpn-common:loopback",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "ip-connection": {
        "ipv6": {
          "local-address": "2001:db8::4",
          "prefix-length": 128
        }
      }
    }
  ]
}

Figure 35: VPN Network Access with a Loopback Interface (Message Body)
Figure 36 shows a simplified example to illustrate how some information that is provided at the VPN service level (particularly as part of the 'vpn-instance-profiles') can be overridden by the one configured at the VPN node level. In this example, PE3 and PE4 inherit the 'vpn-instance-profiles' parameters that are specified at the VPN service level, but PE1 and PE2 are provided with "maximum-routes" values at the VPN node level that override the ones that are specified at the VPN service level.

```json
{
   "ietf-l3vpn-ntw:vpn-services": {
      "vpn-service": [ {
         "vpn-id": "override-example",
         "vpn-service-topology": "ietf-vpn-common:hub-spoke",
         "vpn-instance-profiles": {
            "vpn-instance-profile": [ {
               "profile-id": "HUB",
               "role": "ietf-vpn-common:hub-role",
               "local-as": 64510,
               "rd-suffix": 1001,
               "address-family": [ {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "maximum-routes": [ {
                     "protocol": "ietf-vpn-common:any",
                     "maximum-routes": 100
                  } ]
               } ]
            }, {
               "profile-id": "SPOKE",
               "role": "ietf-vpn-common:spoke-role",
               "local-as": 64510,
               "address-family": [ {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "maximum-routes": [ {
                     "protocol": "ietf-vpn-common:any",
                     "maximum-routes": 1000
                  } ]
               } ]
            } ]
         } ]
      } ]
   }
}```
"vpn-nodes": {
  "vpn-node": [
    {
      "vpn-node-id": "PE1",
      "ne-id": "pe1",
      "router-id": "198.51.100.1",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "HUB",
            "rd": "1:198.51.100.1:1001",
            "address-family": {
              "address-family": "ietf-vpn-common:dual-stack",
              "maximum-routes": [
                {
                  "protocol": "ietf-vpn-common:any",
                  "maximum-routes": 10
                }
              ]
            }
          }
        ]
      }
    },
    {
      "vpn-node-id": "PE2",
      "ne-id": "pe2",
      "router-id": "198.51.100.2",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "SPOKE",
            "address-family": {
              "address-family": "ietf-vpn-common:dual-stack",
              "maximum-routes": [
                {
                  "protocol": "ietf-vpn-common:any",
                  "maximum-routes": 100
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
Figure 36: VPN Instance Profile Example (Message Body)
A.4. Multicast VPN Provisioning Example

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

```
+-----------+   +------+     +------+    +-----------+
| Multicast |---|  CE  |--|--|  PE  |----|  Backbone |
| source    |   +------+     +------+    |   IP/MPLS |
+-----------+                            +-----------+
```

Figure 37: Multicast L3VPN Service Example

An example is provided below to illustrate how to configure a multicast L3VPN service using the L3NM.

First, the multicast service is created together with a generic VPN instance profile (see the excerpt of the request message body shown in Figure 38)
Figure 38: Create Multicast VPN Service (Excerpt of the Message Request Body)
Then, the VPN nodes are created (see the excerpt of the request message body shown in Figure 39). In this example, the VPN node will represent VRF configured in the physical device.

```
{
  "ietf-l3vpn-ntw:vpn-node": [
    {
      "vpn-node-id": "500003105",
      "description": "VRF-IPTV-MULTICAST",
      "ne-id": "198.51.100.10",
      "router-id": "198.51.100.10",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "multicast",
            "rd": "65536:31050202"
          }
        ]
      }
    }
  ]
}
```

Figure 39: Create Multicast VPN Node (Excerpt of the Message Request Body)

Finally, create the VPN network access with multicast enabled (see the excerpt of the request message body shown in Figure 40).

```
{
  "ietf-l3vpn-ntw:vpn-network-access": {
    "id": "1/1/1",
    "description": "Connected-to-source",
    "vpn-network-access-type": "ietf-vpn-common:point-to-point",
    "vpn-instance-profile": "multicast",
    "status": {
      "admin-status": {
        "status": "vpn-common:admin-up"
      },
      "ip-connection": {
        "ipv4": {
          "local-address": "203.0.113.1",
          "prefix-length": 30,
          "address-allocation-type": "static-address",
          "static-addresses": {
            "primary-address": "1",
            "address": [
```
"address-id": "1",
  "customer-address": "203.0.113.2"
}]
},
"routing-protocols": {
  "routing-protocol": [
    {
      "id": "1",
      "type": "ietf-vpn-common:bgp-routing",
      "bgp": {
        "description": "Connected to CE",
        "peer-as": "65537",
        "address-family": "ietf-vpn-common:ipv4",
        "neighbor": "203.0.113.2"
      }
    }
  ],
},
"service": {
  "inbound-bandwidth": "100000000",
  "outbound-bandwidth": "100000000",
  "mtu": 1500,
  "multicast": {
    "access-type": "source-only",
    "address-family": "ietf-vpn-common:ipv4",
    "protocol-type": "router",
    "pim": {
      "hello-interval": 30,
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      }
    }
  }
}
}

Figure 40: Create VPN Network Access (Excerpt of the Message Request Body)
Appendix B. Implementation Status

This section records the status of known implementations of the YANG module defined by this specification at the time of posting of this document and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note to the RFC Editor: As per [RFC7942] guidelines, please remove this Implementation Status appendix prior publication.

B.1. Nokia Implementation

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

B.2. Huawei Implementation

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

B.3. Infinera Implementation

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

B.4. Ribbon-ECI Implementation

Details can be found at: https://github.com/IETF-OPSAWG-WG/l3nm/blob/master/Implementattion/Ribbon-ECI.txt
B.5. Juniper Implementation

https://github.com/IETF-OPSAWG-WG/lxnm/blob/master/Implementation/
Juniper

Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from (listed alphabetically): Raul Arco, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Greg Mirsky, and Tom Petch. Many thanks to them. Thanks to Philip Eardly for the review of an early version of the document.

Daniel King, Daniel Voyer, Luay Jalil, and Stephane Litkowski contributed to early version of the individual submission. Many thanks to Robert Wilton for the AD review. Thanks to Andrew Malis for the routing directorate review, Rifaat Shekh-Yusef for the security directorate review, Qin Wu for the opsdir review, and Pete Resnick for the genart directorate review. Thanks to Michael Scharf for the discussion on TCP-AO. Thanks to Martin Duke, Lars Eagert, Zaheduzzaman Sarker, Roman Danyliw, Erik Kline, Benjamin Kaduk, Francesca Palombini, and Eric Vyncke for the IESG review.

This work was supported in part by the European Commission funded H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727) and Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (G.A. 101015857).

Contributors
Authors’ Addresses

Samier Barguil
Telefonica
Madrid
Spain

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
Spain

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com
Abstract

This memo extends the Manufacturer Usage Description (MUD) specification to incorporate (D)TLS profile parameters. This allows a network security service to identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software or malware on an endpoint.
1. Introduction

Encryption is necessary to enhance the privacy of end users using IoT devices. TLS [RFC8446] and DTLS [I-D.ietf-tls-dtls13] are the dominant protocols (counting all (D)TLS versions) providing encryption for IoT device traffic. Unfortunately, in conjunction with IoT applications’ rise of encryption, malware authors are also using encryption which thwarts network-based analysis such as deep packet inspection (DPI). Other mechanisms are thus needed to help
detecting malware running on an IoT device.

Malware frequently uses proprietary libraries for its activities, and those libraries are reused much like any other software engineering project. [malware] indicates that there are observable differences in how malware uses encryption compared with how non-malware uses encryption. There are several interesting findings specific to (D)TLS which were found common to malware:

* Older and weaker cryptographic parameters (e.g., TLS_RSA_WITH_RC4_128_SHA).

* TLS server name indication (SNI) extension [RFC6066] and server certificates are composed of subjects with characteristics of a domain generation algorithm (DGA) (e.g., 'www.33mhwt2j.net').

* Higher use of self-signed certificates compared with typical legitimate software.

* Discrepancies in the SNI TLS extension and the DNS names in the SubjectAltName (SAN) X.509 extension in the server certificate message.

* Discrepancies in the key exchange algorithm and the client public key length in comparison with legitimate flows. As a reminder, the Client Key Exchange message has been removed from TLS 1.3.

* Lower diversity in TLS client advertised extensions compared to legitimate clients.

* Using privacy enhancing technologies like Tor, Psiphon, Ultrasurf (see [malware-tls]), and evasion techniques such as ClientHello randomization.

* Using DNS-over-HTTPS (DoH) [RFC8484] to avoid detection by malware DNS filtering services [malware-doh]. Specifically, malware may not use the DoH server provided by the local network.

If observable (D)TLS profile parameters are used, the following functions are possible which have a positive impact on the local network security:

* Permit intended DTLS or TLS use and block malicious DTLS or TLS use. This is superior to the layers 3 and 4 ACLs of Manufacturer Usage Description Specification (MUD) [RFC8520] which are not suitable for broad communication patterns.
* Ensure TLS certificates are valid. Several TLS deployments have been vulnerable to active Man-In-The-Middle (MITM) attacks because of the lack of certificate validation or vulnerability in the certificate validation function (see [crypto-vulnerability]). By observing (D)TLS profile parameters, a network element can detect when the TLS SNI mismatches the SubjectAltName and when the server’s certificate is invalid. In TLS 1.2, the ClientHello, ServerHello and Certificate messages are all sent in clear-text. This check is not possible with TLS 1.3, which encrypts the Certificate message thereby hiding the server identity from any intermediary. In TLS 1.3, the server certificate validation functions should be executed within an on-path TLS proxy, if such a proxy exists.

* Support new communication patterns. An IoT device can learn a new capability, and the new capability can change the way the IoT device communicates with other devices located in the local network and Internet. There would be an inaccurate policy if an IoT device rapidly changes the IP addresses and domain names it communicates with while the MUD ACLs were slower to update (see [clear-as-mud]). In such a case, observable (D)TLS profile parameters can be used to permit intended use and to block malicious behavior from the IoT device.

The YANG module specified in Section 5 of this document is an extension of YANG Data Model for Network Access Control Lists (ACLs) [RFC8519] to enhance MUD [RFC8520] to model observable (D)TLS profile parameters. Using these (D)TLS profile parameters, an active MUD-enforcing network security service (e.g., firewall) can identify MUD non-compliant (D)TLS behavior indicating outdated cryptography or malware. This detection can prevent malware downloads, block access to malicious domains, enforce use of strong ciphers, stop data exfiltration, etc. In addition, organizations may have policies around acceptable ciphers and certificates for the websites the IoT devices connect to. Examples include no use of old and less secure versions of TLS, no use of self-signed certificates, deny-list or accept-list of Certificate Authorities, valid certificate expiration time, etc. These policies can be enforced by observing the (D)TLS profile parameters. Network security services can use the IoT device’s (D)TLS profile parameters to identify legitimate flows by observing (D)TLS sessions, and can make inferences to permit legitimate flows and to block malicious or insecure flows. The proposed technique is also suitable in deployments where decryption techniques are not ideal due to privacy concerns, non-cooperating end-points, and expense.
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.

"(D)TLS" is used for statements that apply to both Transport Layer Security [RFC8446] and Datagram Transport Layer Security [RFC6347]. Specific terms are used for any statement that applies to either protocol alone.

'DoH/DoT' refers to DNS-over-HTTPS and/or DNS-over-TLS.

3. Overview of MUD (D)TLS profiles for IoT devices

In Enterprise networks, protection and detection are typically done both on end hosts and in the network. Host security agents have deep visibility on the devices where they are installed, whereas the network has broader visibility. Installing host security agents may not be a viable option on IoT devices, and network-based security is an efficient means to protect such IoT devices. If the IoT device supports a MUD (D)TLS profile, the (D)TLS profile parameters of the IoT device can be used by a middlebox to detect and block malware communication, while at the same time preserving the privacy of legitimate uses of encryption. The middlebox need not proxy (D)TLS but can passively observe the parameters of (D)TLS handshakes from IoT devices and gain visibility into TLS 1.2 parameters and partial visibility into TLS 1.3 parameters.
Malicious agents can try to use the (D)TLS profile parameters of legitimate agents to evade detection, but it becomes a challenge to mimic the behavior of various IoT device types and IoT device models from several manufacturers. In other words, malware developers will have to develop malicious agents per IoT device type, manufacturer and model, infect the device with the tailored malware agent and will have keep up with updates to the device’s (D)TLS profile parameters over time. Furthermore, the malware’s command and control server certificates need to be signed by the same certifying authorities trusted by the IoT devices. Typically, IoT devices have an infrastructure that supports a rapid deployment of updates, and malware agents will have a near-impossible task of similarly deploying updates and continuing to mimic the TLS behavior of the IoT device it has infected. However, if the IoT device has reached end-of-life and the IoT manufacturer will not issue a firmware or software update to the Thing or will not update the MUD file, the "is-supported" attribute defined in Section 3.6 of [RFC8520] can be used by the MUD manager to identify the IoT manufacturer no longer supports the device.

The end-of-life of a device does not necessarily mean that it is defective; rather, it denotes a need to replace and upgrade the network to next-generation devices for additional functionality. The network security service will have to rely on other techniques discussed in Section 8 to identify malicious connections until the device is replaced.

Compromised IoT devices are typically used for launching DDoS attacks (Section 3 of [RFC8576]). For example, DDoS attacks like Slowloris and Transport Layer Security (TLS) re-negotiation can be blocked if the victim’s server certificate is not be signed by the same certifying authorities trusted by the IoT device.

4. (D)TLS 1.3 Handshake

In (D)TLS 1.3, full (D)TLS handshake inspection is not possible since all (D)TLS handshake messages excluding the ClientHello message are encrypted. (D)TLS 1.3 has introduced new extensions in the handshake record layers called Encrypted Extensions. Using these extensions handshake messages will be encrypted and network security services (such as a firewall) are incapable to decipher the handshake, and thus cannot view the server certificate. However, the ClientHello and ServerHello still have some fields visible, such as the list of supported versions, named groups, cipher suites, signature algorithms and extensions in ClientHello, and chosen cipher in the ServerHello. For instance, if the malware uses evasion techniques like ClientHello randomization, the observable list of cipher suites and extensions offered by the malware agent in the ClientHello message will not
match the list of cipher suites and extensions offered by the legitimate client in the ClientHello message, and the middlebox can block malicious flows without acting as a (D)TLS 1.3 proxy.

4.1. Full (D)TLS 1.3 Handshake Inspection

To obtain more visibility into negotiated TLS 1.3 parameters, a middlebox can act as a (D)TLS 1.3 proxy. A middlebox can act as a (D)TLS proxy for the IoT devices owned and managed by the IT team in the Enterprise network and the (D)TLS proxy must meet the security and privacy requirements of the organization. In other words, the scope of middlebox acting as a (D)TLS proxy is restricted to Enterprise network owning and managing the IoT devices. The middlebox would have to follow the behaviour detailed in Section 9.3 of [RFC8446] to act as a compliant (D)TLS 1.3 proxy.

To further increase privacy, Encrypted Client Hello (ECH) extension [I-D.ietf-tls-esni] prevents passive observation of the TLS Server Name Indication extension and other potentially sensitive fields, such as the ALPN [RFC7301]. To effectively provide that privacy protection, ECH extension needs to be used in conjunction with DNSSEC or DNS encryption (e.g., DoH). A middlebox (e.g., firewall) passively inspecting ECH extension cannot observe the encrypted SNI nor observe the encrypted DNS traffic. The middlebox would have to follow the behaviour detailed in [I-D.ietf-tls-esni] to disable ECH or fake ECH records in the DNS response so that the client encrypts data to the middlebox or strips the ECH record from the DNS response. However, if the client performs full DNSSEC validation locally [RFC6698], it can detect forged DNS responses.

4.2. Encrypted DNS

A common usage pattern for certain type of IoT devices (e.g., light bulb) is for it to "call home" to a service that resides on the public Internet, where that service is referenced through a domain name (A or AAAA record). As discussed in Manufacturer Usage Description Specification [RFC8520], because these devices tend to require access to very few sites, all other access should be considered suspect. If an IoT device is pre-configured to use a DNS resolver not signaled by the network, the MUD policy enforcement point is moved to that resolver, which cannot enforce the MUD policy based on domain names (Section 8 of [RFC8520]). If the DNS query is not accessible for inspection, it becomes quite difficult for the infrastructure to suspect anything. Thus the use of a DNS resolver not signaled by the network is incompatible with MUD in general. A network-designated DoH/DoT server is necessary to allow MUD policy enforcement on the local network (Section 6.5 of [I-D.ietf-opsawg-mud-iot-dns-considerations]).
5. (D)TLS Profile of a IoT device

This document specifies a YANG module for representing (D)TLS profile. The (D)TLS profile YANG module provides a method for network security services to observe the (D)TLS profile parameters in the (D)TLS handshake to permit intended use and to block malicious behavior. This module uses the cryptographic types defined in [I-D.ietf-netconf-crypto-types]. See [RFC7925] for (D)TLS 1.2 and [I-D.ietf-uta-tls13-iot-profile] for DTLS 1.3 recommendations related to IoT devices, and [RFC7525] for additional (D)TLS 1.2 recommendations.

A companion YANG module is defined to include a collection of (D)TLS parameters and (D)TLS versions maintained by IANA: "iana-tls-profile" (Section 5.3).

The (D)TLS parameters in each (D)TLS profile include the following:

* Profile name
* (D)TLS versions supported by the IoT device.
* List of supported cipher suites. For (D)TLS1.2, [RFC7925] recommends AEAD ciphers for IoT devices.
* List of supported extension types
* List of trust anchor certificates used by the IoT device. If the server certificate is signed by one of the trust anchors, the middlebox continues with the connection as normal. Otherwise, the middlebox will react as if the server certificate validation has failed and takes appropriate action (e.g., block the (D)TLS session). An IoT device can use a private trust anchor to validate a server's certificate (e.g., the private trust anchor can be preloaded at manufacturing time on the IoT device and the IoT device fetches the firmware image from the Firmware server whose certificate is signed by the private CA). This empowers the middlebox to reject TLS sessions to servers that the IoT device does not trust.
* List of SPKI pin set pre-configured on the client to validate self-signed server certificates or raw public keys. A SPKI pin set is a cryptographic digest to "pin" public key information in a manner similar to HTTP Public Key Pinning (HPKP) [RFC7469]. If SPKI pin set is present in the (D)TLS profile of a IoT device and the server certificate does not pass the PKIX certification path validation, the middlebox computes the SPKI Fingerprint for the public key found in the server’s certificate (or in the raw public
key, if the server provides that instead). If a computed fingerprint exactly matches one of the SPKI pin sets in the (D)TLS profile, the middlebox continues with the connection as normal. Otherwise, the middlebox will act on the SPKI validation failure and takes appropriate action.

* Cryptographic hash algorithm used to generate the SPKI pinsets

* List of pre-shared key exchange modes

* List of named groups (DHE or ECDHE) supported by the client

* List of signature algorithms the client can validate in X.509 server certificates

* List of signature algorithms the client is willing to accept for CertificateVerify message (Section 4.2.3 of [RFC8446]). For example, a TLS client implementation can support different sets of algorithms for certificates and in TLS to signal the capabilities in "signature_algorithms_cert" and "signature_algorithms" extensions.

* List of supported application protocols (e.g., h3, h2, http/1.1 etc.)

* List of certificate compression algorithms (defined in [I-D.ietf-tls-certificate-compression])

* List of the distinguished names [X501] of acceptable certificate authorities, represented in DER-encoded format [X690] (defined in Section 4.2.4 of [RFC8446])

GREASE [RFC8701] sends random values on TLS parameters to ensure future extensibility of TLS extensions. Similar random values might be extended to other TLS parameters. Thus, the (D)TLS profile parameters defined in the YANG module by this document MUST NOT include the GREASE values for extension types, named groups, signature algorithms, (D)TLS versions, pre-shared key exchange modes, cipher suites and for any other TLS parameters defined in future RFCs.

The (D)TLS profile does not include parameters like compression methods for data compression, [RFC7525] recommends disabling TLS-level compression to prevent compression-related attacks. In TLS 1.3, only the "null" compression method is allowed (Section 4.1.2 of [RFC8446]).
5.1. Tree Structure of the (D)TLS profile Extension to the ACL YANG Model

This document augments the "ietf-acl" ACL YANG module defined in [RFC8519] for signaling the IoT device (D)TLS profile. This document defines the YANG module "ietf-acl-tls", which has the following tree structure:

```
module: ietf-acl-tls
   augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
     +--rw client-profile (match-on-tls-dtls)?
       +--rw tls-dtls-profiles* (profile-name)
         +--rw profile-name string
         +--rw supported-tls-versions* ianatp:tls-version
         +--rw supported-dtls-versions* ianatp:dtls-version
         +--rw cipher-suites* (cipher hash)
           |   +--rw cipher ianatp:cipher-algorithm
           |   +--rw hash ianatp:hash-algorithm
           +--rw extension-types* ianatp:extension-type
           +--rw acceptlist-ta-certs* ct:trust-anchor-cert-cms
           +--rw spki
             |   +--rw spki-pin-sets* ianatp:spki-pin-set
             |   +--rw spki-hash-algorithm? iha:hash-algorithm-type
             +--rw psk-key-exchange-modes* ianatp:psk-key-exchange-mode
               |   |   (tls-1-3 or dtls-1-3)?
             +--rw supported-groups* ianatp:supported-group
             +--rw signature-algorithms-cert* ianatp:signature-algorithm
               |   (tls-1-3 or dtls-1-3)?
             +--rw signature-algorithms* ianatp:signature-algorithm
             +--rw application-protocols* ianatp:application-protocol
             +--rw cert-compression-algorithms* ianatp:cert-compression-algorithm
               |   (tls-1-3 or dtls-1-3)?
             +--rw certificate-authorities* ianatp:certificate-authority
               |   (tls-1-3 or dtls-1-3)?
```

5.2. The (D)TLS profile Extension to the ACL YANG Model
<CODE BEGINS> file "ietf-acl-tls@2020-10-07.yang"
module ietf-acl-tls {
    yang-version 1.1;
    prefix ietf-acl-tls;

    import iana-tls-profile {
        prefix ianatp;
        reference
            "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices";
    }

    import ietf-crypto-types {
        prefix ct;
        reference
            "RFC CCCC: Common YANG Data Types for Cryptography";
    }

    import iana-hash-algs {
        prefix iha;
        reference
            "RFC IIII: Common YANG Data Types for Hash algorithms";
    }

    import ietf-access-control-list {
        prefix acl;
        reference
            "RFC 8519: YANG Data Model for Network Access Control Lists (ACLs)";
    }

    organization
        "IETF OPSAWG (Operations and Management Area Working Group)";
    contact
        "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
        WG List: opsawg@ietf.org
        Author: Konda, Tirumaleswar Reddy
        TirumaleswarReddy_Konda@McAfee.com"
    description
        "This YANG module defines a component that augments the IETF description of an access list to allow (D)TLS profile as matching criteria.

        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

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

revision 2020-11-02 {
    description
        "Initial revision";
    reference
        "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices";
}

feature tls-1-2 {
    description
        "TLS Protocol Version 1.2 is supported.";
    reference
}

feature tls-1-3 {
    description
        "TLS Protocol Version 1.3 is supported.";
    reference
}

feature dtls-1-2 {
    description
        "DTLS Protocol Version 1.2 is supported.";
    reference
        "RFC 6346: Datagram Transport Layer Security Version 1.2";
}

feature dtls-1-3 {
    description
        "DTLS Protocol Version 1.3 is supported.";
    reference
        "draft-ietf-tls-dtls13: Datagram Transport Layer Security 1.3";
}
feature match-on-tls-dtls {
  description
  "The networking device can support matching on
  (D)TLS parameters."
}

  if-feature "match-on-tls-dtls"
  description
  "(D)TLS specific matches."
  container client-profile {
    description
    "A grouping for (D)TLS profiles."
    container client-profile {
      description
      "A grouping for DTLS profiles."
      list tls-dtls-profiles {
        key "profile-name"
        description
        "A list of (D)TLS version profiles supported by
         the client."
        leaf profile-name {
          type string {
            length "1..64"
          }
          description
          "The name of (D)TLS profile; space and special
           characters are not allowed."
        }
        leaf-list supported-tls-versions {
          type ianatp:tls-version;
          description
          "TLS versions supported by the client."
        }
        leaf-list supported-dtls-versions {
          type ianatp:dtls-version;
          description
          "DTLS versions supported by the client."
        }
      }
      list cipher-suites {
        key "cipher hash";
        leaf cipher {
          type ianatp:cipher-algorithm;
          description
          "AEAD encryption algorithm as defined in RFC5116."
        }
        leaf hash {
          type ianatp:hash-algorithm;
        }
      }
    }
  }
}

description
"Hash algorithm used with HKDF as defined in RFC5869.";
}
description
"A list of Cipher Suites supported by the client.";
}
leaf-list extension-types {
  type ianatp:extension-type;
  description
  "A list of Extension Types supported by the client.";
}
leaf-list acceptlist-ta-certs {
  type ct:trust-anchor-cert-cms;
  description
  "A list of trust anchor certificates used by the client.";
}
container spki {
  description
  "A grouping for spki.";
  leaf-list spki-pin-sets {
    type ianatp:spki-pin-set;
    description
    "A list of SPKI pin sets pre-configured on the client to validate self-signed server certificate or raw public key.";
  }
  leaf spki-hash-algorithm {
    type iha:hash-algorithm-type;
    description
    "Cryptographic hash algorithm used to generate the SPKI pinset.";
  }
}
leaf-list psk-key-exchange-modes {
  if-feature "tls-1-3 or dtls-1-3";
  type ianatp:psk-key-exchange-mode;
  description
  "Pre-shared key exchange modes.";
}
leaf-list supported-groups {
  type ianatp:supported-group;
  description
  "A list of named groups supported by the client.";
}
leaf-list signature-algorithms-cert {
  if-feature "tls-1-3 or dtls-1-3";
  type ianatp:signature-algorithm;
  description
  "Signature algorithms used with self-signed server certificate or raw server certificate.";
}
"A list signature algorithms the client can validate in X.509 certificates."
}
leaf-list signature-algorithms {
  type ianatp:signature-algorithm;
  description
    "A list signature algorithms the client can validate in the CertificateVerify message."
}
leaf-list application-protocols {
  type ianatp:application-protocol;
  description
    "A list application protocols supported by the client."
}
leaf-list cert-compression-algorithms {
  if-feature "tls-1-3 or dtls-1-3";
  type ianatp:cert-compression-algorithm;
  description
    "A list certificate compression algorithms supported by the client."
}
leaf-list certificate-authorities {
  if-feature "tls-1-3 or dtls-1-3";
  type ianatp:certificate-authority;
  description
    "A list of the distinguished names of certificate authorities acceptable to the client."
}

5.3. IANA (D)TLS profile YANG Module

The TLS and DTLS IANA registries are available from https://www.iana.org/assignments/tls-parameters/tls-parameters.txt and https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.txt.

The values for all the parameters in the "iana-tls-profile" YANG module are defined in the TLS and DTLS IANA registries excluding the tls-version, dtls-version, spki-pin-set, and certificate-authority parameters. The values of spki-pin-set and certificate-authority parameters will be specific to the IoT device.
The TLS and DTLS IANA registries do not maintain (D)TLS version numbers. In (D)TLS 1.2 and below, "legacy_version" field in the ClientHello message is used for version negotiation. However in (D)TLS 1.3, the "supported_versions" extension is used by the client to indicate which versions of (D)TLS it supports. TLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0x0303 and a "supported_versions" extension present with 0x0304 as the highest version. DTLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0xfefd and a "supported_versions" extension present with 0x0304 as the highest version.

In order to ease updating the "iana-tls-profile" YANG module with future (D)TLS versions, new (D)TLS version registries are defined in Section 10.2 and Section 10.3. Whenever a new (D)TLS protocol version is defined, the registry will be updated using expert review; the "iana-tls-profile" YANG module will be automatically updated by IANA.

The "iana-tls-profile" YANG module is defined as follows:

```
<CODE BEGINS> file "iana-tls-profile@2020-10-07.yang"
module iana-tls-profile {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-tls-profile";
  prefix ianatp;

  organization
    "IANA";
  contact
    "Internet Assigned Numbers Authority

    Postal: ICANN
    12025 Waterfront Drive, Suite 300
    Los Angeles, CA 90094-2536
    United States

    Tel: +1 310 301 5800
    E-Mail: iana@iana.org";
  description
    "This module contains YANG definition for the (D)TLS profile.

    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
```

typedef extension-type {
  type uint16;
  description
    "Extension type in the TLS ExtensionType Values registry as defined in Section 7 of RFC8447.";
}

typedef supported-group {
  type uint16;
  description
    "Supported Group in the TLS Supported Groups registry as defined in Section 9 of RFC8447.";
}

typedef spki-pin-set {
  type binary;
  description
    "Subject Public Key Info pin set as discussed in Section 2.4 of RFC7469.";
}

typedef signature-algorithm {
  type uint16;
  description
    "Signature algorithm in the TLS SignatureScheme registry as defined in Section 11 of RFC8446.";
}

typedef psk-key-exchange-mode {
  type uint8;
  description
    "Pre-shared key exchange mode in the TLS PskKeyExchangeMode registry as defined in Section 11 of RFC8446.";
}
typedef application-protocol {
    type string;
    description
        "Application-Layer Protocol Negotiation (ALPN) Protocol ID registry as defined in Section 6 of RFC7301.";
}

typedef cert-compression-algorithm {
    type uint16;
    description
        "Certificate compression algorithm in TLS Certificate Compression Algorithm IDs registry as defined in Section 7.3 of ietf-tls-certificate-compression";
}

typedef certificate-authority {
    type string;
    description
        "Distinguished Name of Certificate authority as discussed in Section 4.2.4 of RFC8446.";
}

typedef cipher-algorithm {
    type uint8;
    description
        "AEAD encryption algorithm in TLS Cipher Suites registry as discussed in Section 11 of RFC8446.";
}

typedef hash-algorithm {
    type uint8;
    description
        "Hash algorithm used with HMAC-based Extract-and-Expand Key Derivation Function (HKDF) in TLS Cipher Suites registry as discussed in Section 11 of RFC8446.";
}

typedef tls-version {
    type enumeration {
        enum tls-1.2 {
            value 1;
            description
                "TLS Protocol Version 1.2.

                TLS 1.2 ClientHello contains 0x0303 in 'legacy_version'.";
            reference
                "RFC 5246: The Transport Layer Security (TLS) Protocol"
        }
    }
}
Version 1.2;
}
enum tls-1.3 {
    value 2;
    description
    "TLS Protocol Version 1.3.
    
    TLS 1.3 ClientHello contains a
    supported_versions extension with 0x0304
    contained in its body and the ClientHello contains
    0x0303 in 'legacy_version'.";
    reference
    Version 1.3";
}
}
description
"Indicates the TLS version.";
}
typedef dtls-version {
    type enumeration {
        enum dtls-1.2 {
            value 1;
            description
            "DTLS Protocol Version 1.2.
            
            DTLS 1.2 ClientHello contains
            0xfefd in 'legacy_version'.";
            reference
            "RFC 6346: Datagram Transport Layer Security 1.2";
        }
        enum dtls-1.3 {
            value 2;
            description
            "DTLS Protocol Version 1.3.
            
            DTLS 1.3 ClientHello contains a
            supported_versions extension with 0x0304
            contained in its body and the ClientHello contains
            0xfefd in 'legacy_version'.";
            reference
            "RFC DDDD: Datagram Transport Layer Security 1.3";
        }
    }
    description
    "Indicates the DTLS version.";
}
5.4. MUD (D)TLS Profile Extension

This document augments the "ietf-mud" MUD YANG module to indicate whether the device supports (D)TLS profile. If the "ietf-mud-tls" extension is supported by the device, MUD file is assumed to implement the "match-on-tls-dtls" ACL model feature defined in this specification. Furthermore, only "accept" or "drop" actions SHOULD be included with the (D)TLS profile similar to the actions allowed in Section 2 of [RFC8520].

This document defines the YANG module "ietf-mud-tls", which has the following tree structure:

```
module: ietf-mud-tls
  augment /ietf-mud:mud:
    +--rw is-tls-dtls-profile-supported?   boolean
```

The model is defined as follows:

```
<CODE BEGINS> file "iana-tls-mud@2020-10-20.yang"
module ietf-mud-tls {
  yang-version 1.1;
  prefix ietf-mud-tls;

  import ietf-mud {
    prefix ietf-mud;
  }

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

    Author: Konda, Tirumaleswar Reddy
    TirumaleswarReddy_Konda@McAfee.com
  ";
  description
    "Extension to a MUD module to indicate (D)TLS profile support."

  copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.
```
6. Processing of the MUD (D)TLS Profile

The following text outlines the rules for a network security service (e.g., firewall) to follow to process the MUD (D)TLS Profile:

* If the (D)TLS parameter observed in a (D)TLS session is not specified in the MUD (D)TLS profile and the parameter is recognized by the firewall, it can identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software or malware on an endpoint. The firewall can take several actions like block the (D)TLS session or raise an alert to quarantine and remediate the compromised device. For example, if the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA in the ClientHello message is not specified in the MUD (D)TLS profile and the cipher suite is recognized by the firewall, it can identify unexpected TLS usage.
* If the (D)TLS parameter observed in a (D)TLS session is not specified in the MUD (D)TLS profile and the (D)TLS parameter is not recognized by the firewall, it can ignore the unrecognized parameter and the correct behavior is not to block the (D)TLS session. The behaviour is functionally equivalent to the compliant TLS middlebox description in Section 9.3 of [RFC8446] to ignore all unrecognized cipher suites, extensions, and other parameters. For example, if the cipher suite TLS_CHACHA20_POLY1305_SHA256 in the ClientHello message is not specified in the MUD (D)TLS profile and the cipher suite is not recognized by the firewall, it can ignore the unrecognized cipher suite.

* Deployments update at different rates, so an updated MUD (D)TLS profile may support newer parameters. If the firewall does not recognize the newer parameters, an alert should be triggered to the firewall vendor and the IoT device owner or administrator. A firewall must be readily updatable, so that when new parameters in the MUD (D)TLS profile are discovered that are not recognized by the firewall, it can be updated quickly. Most importantly, if the firewall is not readily updatable, its protection efficacy to identify emerging malware will decrease with time. For example, if the cipher suite TLS_AES_128_CCM_8_SHA256 specified in the MUD (D)TLS profile is not recognized by the firewall, an alert will be triggered. Similarly, if the (D)TLS version specified in the MUD file is not recognized by the firewall, an alert will be triggered.

7. MUD File Example

The example below contains (D)TLS profile parameters for a IoT device used to reach servers listening on port 443 using TCP transport. JSON encoding of YANG modelled data [RFC7951] is used to illustrate the example.

```json
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://example.com/IoTDevice",
    "last-update": "2019-18-06T03:56:40.105+10:00",
    "cache-validity": 100,
    "extensions": [
      "ietf-mud-tls"
    ],
    "ietf-mud-tls:is-tls-dtls-profile-supported": "true",
    "is-supported": true,
    "systeminfo": "IoT device name",
    "from-device-policy": {
      "...
```
"access-lists": {
    "access-list": [
      {
        "name": "mud-7500-profile"
      }
    ]
  },
"ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-7500-profile",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv6": {
                  "protocol": 6
                },
                "tcp": {
                  "ietf-mud:direction-initiated": "from-device",
                  "destination-port": {
                    "operator": "eq",
                    "port": 443
                  }
                }
              },
              "ietf-acl-tls:client-profile": {
                "tls-dtls-profiles": [
                  {
                    "supported-tls-versions": ["tls-1.3"],
                    "cipher-suites": [
                      {
                        "cipher": 19,
                        "hash": 1
                      },
                      {
                        "cipher": 19,
                        "hash": 2
                      }
                    ],
                    "extension-types": [10,11,13,16,24],
                    "supported-groups": [29]
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
"actions": {

The following illustrates the example scenarios for processing the above profile:

* If the extension type "encrypt_then_mac" (code point 22) [RFC7366] in the ClientHello message is recognized by the firewall, it can identify unexpected TLS usage.

* If the extension type "token_binding" (code point 24) [RFC8472] in the MUD (D)TLS profile is not recognized by the firewall, it can ignore the unrecognized extension. Because the extension type "token_binding" is specified in the profile, an alert will be triggered to the firewall vendor and the IoT device owner or administrator to notify the firewall is not up to date.

8. Security Considerations

Security considerations in [RFC8520] need to be taken into consideration. The middlebox must adhere to the invariants discussed in Section 9.3 of [RFC8446] to act as a compliant proxy.

Although it is challenging for a malware to mimic the TLS behavior of various IoT device types and IoT device models from several manufacturers, malicious agents have a very low probability of using the same (D)TLS profile parameters as legitimate agents on the IoT device to evade detection. Network security services should also rely on contextual network data to detect false negatives. In order to detect such malicious flows, anomaly detection (deep learning techniques on network data) can be used to detect malicious agents using the same (D)TLS profile parameters as legitimate agent on the IoT device. In anomaly detection, the main idea is to maintain rigorous learning of "normal" behavior and where an "anomaly" (or an attack) is identified and categorized based on the knowledge about the normal behavior and a deviation from this normal behavior.
9. Privacy Considerations

Privacy considerations discussed in Section 16 of [RFC8520] to not reveal the MUD URL to an attacker need to be taken into consideration. The MUD URL can be stored in Trusted Execution Environment (TEE) for secure operation, enhanced data security, and prevent exposure to unauthorized software.

Full handshake inspection (Section 4.1) requires a TLS proxy device which needs to decrypt traffic between the IoT device and its server(s). There is a tradeoff between privacy of the data carried inside TLS (especially e.g., personally identifiable information and protected health information) and efficacy of endpoint security. It is strongly RECOMMENDED to avoid a TLS proxy whenever possible. For example, an enterprise firewall administrator can configure the middlebox to bypass TLS proxy functionality or payload inspection for connections destined to specific well-known services. Alternatively, a IoT device could be configured to reject all sessions that involve proxy servers to specific well-known services. In addition, mechanisms based on object security can be used by IoT devices to enable end-to-end security and the middlebox will not have any access to the packet data. For example, Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] is a proposal that protects CoAP messages by wrapping them in the COSE format [RFC8152].

10. IANA Considerations

10.1. (D)TLS Profile YANG Modules

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

  Registrant Contact: The IESG.
  XML: N/A; the requested URI is an XML namespace.

  Registrant Contact: The IESG.
  XML: N/A; the requested URI is an XML namespace.

  Registrant Contact: The IESG.
  XML: N/A; the requested URI is an XML namespace.
IANA is requested to create an IANA-maintained YANG Module called "iana-tls-profile", based on the contents of Section 5.3, which will allow for new (D)TLS parameters and (D)TLS versions to be added to "client-profile". The registration procedure will be Expert Review, as defined by [RFC8126].

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

```plaintext
name: iana-tls-profile
maintained by IANA: Y
prefix: ianatp
reference: RFC XXXX

name: ietf-acl-tls
maintained by IANA: N
prefix: ietf-acl-tls
reference: RFC XXXX

name: ietf-mud-tls
maintained by IANA: N
prefix: ietf-mud-tls
reference: RFC XXXX
```

IANA is requested to create an the initial version of the IANA-maintained YANG Module called "iana-tls-profile", based on the contents of Section 5.3, which will allow for new (D)TLS parameters and (D)TLS versions to be added. IANA is requested to add this note:

* tls-version and dtls-version values must not be directly added to the iana-tls-profile YANG module. They must instead be respectively added to the "ACL TLS Version Codes", and "ACL DTLS Version Codes" registries.

* (D)TLS parameters must not be directly added to the iana-tls-profile YANG module. They must instead be added to the "ACL (D)TLS Parameters" registry.

When a 'tls-version' or 'dtls-version' value is respectively added to the "ACL TLS Version Codes" or "ACL DTLS Version Codes" registry, a new "enum" statement must be added to the iana-tls-profile YANG module. The following "enum" statement, and substatements thereof, should be defined:
"enum": Replicates the label from the registry.

"value": Contains the IANA-assigned value corresponding to the 'tls-version' or 'dtls-version'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry and adds the title of the document.

When a (D)TLS parameter is added to "ACL (D)TLS Parameters" registry, a new "type" statement must be added to the iana-tls-profile YANG module. The following "type" statement, and substations thereof, should be defined:

"derived type": Replicates the parameter name from the registry.

"built-in type": Contains the built-in YANG type.

"description": Replicates the description from the registry.

When the iana-tls-profile YANG module is updated, a new "revision" statement must be added in front of the existing revision statements.

IANA is requested to add this note to "ACL TLS Version Codes", "ACL DTLS Version Codes", and "ACL (D)TLS Parameters" registries:

When this registry is modified, the YANG module iana-tls-profile must be updated as defined in [RFCXXXX].

The registration procedure for "ietf-acl-tls" YANG module will be Specification Required, as defined by [RFC8126].

10.2. ACL TLS Version registry

IANA is requested to create a new registry titled "ACL TLS Version Codes". Codes in this registry are used as valid values of 'tls-version' parameter. Further assignments are to be made through Expert Review [RFC8126].
10.3. ACL DTLS version registry

IANA is requested to create a new registry titled "ACL DTLS Version Codes". Codes in this registry are used as valid values of 'dtls-version' parameter. Further assignments are to be made through Expert Review [RFC8126].

<table>
<thead>
<tr>
<th>Value</th>
<th>Label</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dtls-1.2</td>
<td>DTLS Version 1.2</td>
<td>[RFC6346]</td>
</tr>
<tr>
<td>2</td>
<td>dtls-1.3</td>
<td>DTLS Version 1.3</td>
<td>[draft-ietf-tls-dtls13]</td>
</tr>
</tbody>
</table>

10.4. ACL (D)TLS Parameters registry

IANA is requested to create a new registry titled "ACL (D)TLS parameters".

The values for all the (D)TLS parameters in the registry are defined in the TLS and DTLS IANA registries (https://www.iana.org/assignments/tls-parameters/tls-parameters.txt and https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.txt) excluding the tls-version, dtls-version, spki-pin-set and certificate-authority parameters. Further assignments are to be made through Expert Review [RFC8126]. The registry is initially populated with the following parameters:
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>YANG Type</th>
<th>JSON Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension-type</td>
<td>uint16</td>
<td>Number</td>
<td>Extension type</td>
</tr>
<tr>
<td>supported-group</td>
<td>uint16</td>
<td>Number</td>
<td>Supported group</td>
</tr>
<tr>
<td>spki-pin-set</td>
<td>binary</td>
<td>String</td>
<td>Subject public key info</td>
</tr>
<tr>
<td>signature-algorithm</td>
<td>uint16</td>
<td>Number</td>
<td>Signature algorithm</td>
</tr>
<tr>
<td>psk-key-exchange-mode</td>
<td>uint8</td>
<td>Number</td>
<td>pre-shared key exchange mode</td>
</tr>
<tr>
<td>application-protocol</td>
<td>string</td>
<td>String</td>
<td>Application protocol</td>
</tr>
<tr>
<td>cert-compression-algorithm</td>
<td>uint16</td>
<td>Number</td>
<td>Certificate compression algorithm</td>
</tr>
<tr>
<td>certificate-authority</td>
<td>string</td>
<td>String</td>
<td>Distinguished name of Certificate Authority</td>
</tr>
<tr>
<td>cipher-algorithm</td>
<td>uint8</td>
<td>Number</td>
<td>AEAD encryption algorithm</td>
</tr>
<tr>
<td>hash-algorithm</td>
<td>uint8</td>
<td>Number</td>
<td>Hash algorithm</td>
</tr>
<tr>
<td>tls-version</td>
<td>enumeration</td>
<td>String</td>
<td>TLS version</td>
</tr>
<tr>
<td>dtls-version</td>
<td>enumeration</td>
<td>String</td>
<td>DTLS version</td>
</tr>
</tbody>
</table>
10.5. MUD Extensions registry

IANA is requested to create a new MUD Extension Name "ietf-mud-tls" in the MUD Extensions IANA registry
https://www.iana.org/assignments/mud/mud.xhtml.

11. Acknowledgments

Thanks to Flemming Andreasen, Shashank Jain, Michael Richardson, Piyush Joshi, Eliot Lear, Harsha Joshi, Qin Wu, Mohamed Boucadair, Ben Schwartz, Eric Rescorla, Panwei William, Nick Lamb and Nick Harper for the discussion and comments.

12. References

12.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021,

[I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", Work in Progress, Internet-Draft, draft-ietf-tls-certificate-compression-10, 6 January 2020,

[I-D.ietf-tls-dtls13]

[RFC2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC3688]
Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,

[RFC6347]
Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347,
DOI 10.17487/RFC6347, January 2012,

[RFC8174]
Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174,
DOI 10.17487/RFC8174, May 2017,

[RFC8446]
DOI 10.17487/RFC8446, August 2018,

[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,
12.2. Informative References

[crypt-of-mud]

[crypto-vulnerability]

[I-D.ietf-opsawg-mud-iot-dns-considerations]

[I-D.ietf-tls-esni]

[I-D.ietf-uta-tls13-iot-profile]

[malware]


Authors’ Addresses
Abstract

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

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 2 April 2022.
1. Introduction

The IETF has specified YANG data modules for VPN services, e.g., Layer 3 VPN Service Model (L3SM) [RFC8299] or Layer 2 VPN Service Model (L2SM) [RFC8466]. Other relevant YANG models are the Layer 3 VPN Network Model (L3NM) [I-D.ietf-opsawg-l3sm-l3nm] and the Layer 2 VPN Network Model (L2NM) [I-D.ietf-opsawg-l2nm]. There are common data nodes and structures that are present in all of these models or at least a subset of them.

This document defines a common YANG module that is meant to be reused by various VPN-related modules such as L3NM [I-D.ietf-opsawg-l3sm-l3nm] and 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 possible future revisions of existing models (e.g., L3SM [RFC8299] or L2SM [RFC8466]).

2. Terminology

The terminology for describing YANG modules is defined in [RFC7950]. The meaning of the symbols in tree diagrams is defined in [RFC8340]. The reader may refer to [RFC4026] and [RFC4176] for VPN-related terms.

The document inherits many terms from [RFC8299] and [RFC8466] (e.g., Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), Massive Machine Type Communications (mMTC)).

3. Description of the VPN Common YANG Module

The "ietf-vpn-common" module defines a set of common VPN-related features, including:

Encapsulation features such as:
* Dot1q [IEEE802.1Q],
  * QinQ [IEEE802.1ad],
  * link aggregation [IEEE802.1AX], and
  * Virtual eXtensible Local Area Network (VXLAN) [RFC7348].

Multicast [RFC6513].

Routing features such as:
* BGP [RFC4271],
  * OSPF [RFC4577][RFC65565],
  * IS-IS [ISO10589],
  * RIP [RFC2080][RFC2453],
  * Bidirectional Forwarding Detection (BFD) [RFC5880][RFC7880], and
* Virtual Router Redundancy Protocol (VRRP) [RFC5798].

Also, the module defines a set of identities, including:

'service-type': Used to identify the VPN service type. Examples of supported service types are:

* L3VPN,
* Virtual Private LAN Service (VPLS) using BGP [RFC4761],
* VPLS using Label Distribution Protocol (LDP) [RFC4762],
* Virtual Private Wire Service (VPWS) [RFC8214],
* BGP MPLS-Based Ethernet VPN [RFC7432],
* Ethernet VPN (EVPN) [RFC8365], and
* Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN) [RFC7623].

'vpn-signaling-type': Used to identify the signaling mode used for a given service type. Examples of supported VPN signaling types are:

* L2VPNs using BGP [RFC6624].
* LDP [RFC5036], and
* Layer Two Tunneling Protocol (L2TP) [RFC3931].

The module covers both IPv4 [RFC0791] and IPv6 [RFC8200] identities. It also includes multicast related identities such as Internet Group Management Protocol version 1 (IGMPv1) [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376], Multicast Listener Discovery version 1 (MLDv1) [RFC2710], MLDv2 [RFC3810], and Protocol Independent Multicast (PIM) [RFC7761].

The reader should refer to Section 4 for the full list of supported identities (identities related to address families, VPN topologies, network access types, operational and administrative status, site or node roles, VPN service constraints, routing protocols, routes imports and exports, bandwidth and Quality of Service (QoS), etc.).

The "ietf-vpn-common" module also contains a set of reusable VPN-related groupings. The tree diagram of the "ietf-vpn-common" module that depicts the common groupings is provided in Figure 1.
module: ietf-vpn-common

grouping vpn-description
  +-- vpn-id?            vpn-id
  +-- vpn-name?          string
  +-- vpn-description?   string
  +-- customer-name?     string

grouping vpn-profile-cfg
  +-- valid-provider-identifiers
    |  +-- external-connectivity-identifier* [id]
    |     (external-connectivity)?
    |     +-- 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 oper-status-timestamp
  +--ro status?         identityref
  +--ro last-change?   yang:date-and-time

grouping service-status
  +-- status
    |  +-- admin-status
    |     |  +-- status?         identityref
    |     |  +-- last-change?   yang:date-and-time
    |  +-- oper-status
    |     +--ro status?         identityref
    |     +--ro last-change?   yang:date-and-time

grouping underlay-transport
  +-- (type)?
    |  +-- (abstract)
    |     |  +-- transport-instance-id?   string
    |  +-- (protocol)
    |     +-- protocol*               identityref

grouping vpn-route-targets
  +-- vpn-target* [id]
    |  +-- id       uint8
    |  +-- 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 route-distinguisher
...
grouping vpn-components-group
  +-- groups
    +-- group* [group-id]
      +-- group-id    string

grouping placement-constraints
  +-- constraint* [constraint-type]
    +-- constraint-type?   identityref
    +-- target
      +-- (target-flavor)?
        +--:(id)
          +--:(all-accesses)
          |   +-- group* [group-id]
          |     +-- group-id    string
          |   +-- all-other-accesses?   empty
          +--:(all-groups)
            +-- all-other-groups?     empty

grouping ports
...
grouping qos-classification-policy
...

Figure 1: VPN Common Tree

The description of the common groupings is provided below:

'vpn-description':
A YANG grouping that provides common administrative VPN
information such as an identifier, a name, a textual
description, and a customer name.

'vpn-profile-cfg':
A YANG grouping that defines a set of valid profiles
(encryption, routing, forwarding, etc.) that can be bound to a
Layer 2/3 VPN. This document does not make any assumption
about the structure of such profiles, but allows "gluing" a VPN
service with other parameters that can be required locally to
provide added value features to requesting customers.
For example, a service provider may provide an external connectivity to a VPN customer (e.g., to a private or public cloud, Internet). Such service may involve tweaking both filtering and NAT rules (e.g., bind a Virtual Routing and Forwarding (VRF) interface with a NAT instance as discussed in Section 2.10 of [RFC8512]). These added value features may be bound to all or a subset of network accesses. Some of these added value features may be implemented in nodes other than PEs (e.g., a P node or even a dedicated node that hosts the NAT function).

It is out of the scope of this document to elaborate the structure of these profiles.

`'oper-status-timestamp'`: A YANG grouping that defines the operational status updates of a VPN service or 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 or an endpoint.

`'underlay-transport'`: A YANG grouping that defines the type of the underlay transport for a VPN service or how that underlay is set.

The underlay transport can be expressed as an abstract transport instance (e.g., an identifier of a VPN+ instance [I-D.ietf-teas-enhanced-vpn], a virtual network identifier [I-D.ietf-teas-actn-vn-yang][RFC8453], or a network slice name [I-D.ietf-teas-ietf-network-slices]) or as an ordered list of the actual protocols to be enabled in the network.

The module supports a rich set of protocol identifiers that can be used, e.g., to refer to an underlay transport. Examples of supported protocols are:

- IP-in-IP [RFC2003][RFC2473],
- GRE [RFC1701][RFC1702][RFC7676],
- MPLS-in-UDP [RFC7510],
- Generic Network Virtualization Encapsulation (GENEVE) [RFC8926],
- Segment Routing (SR) [RFC8660][RFC8663][RFC8754],
- Resource ReSerVation Protocol (RSVP) with traffic engineering extensions [RFC3209], and
- BGP with labeled prefixes [RFC8277].

'vpn-route-targets':
A YANG grouping that defines Route Target (RT) import/export rules used in a BGP-enabled VPN. This grouping can be used for both L3VPNs [RFC4364] and L2VPNs [RFC4664]. Note that this is modelled as a list to ease the reuse of this grouping in modules where an RT identifier is needed (e.g., associate an operator with RTs).

$route-distinguisher':
A YANG grouping that defines Route Distinguishers (RDs).

As depicted in Figure 2, the module supports these RD assignment modes: direct assignment, automatic assignment from a given pool, automatic assignment, and no assignment.

Also, the module accommodates deployments where only the Assigned Number subfield of RDs (Section 4.2 of [RFC4364]) is assigned from a pool while the Administrator subfield is set to, e.g., the router-id that is assigned to a VPN node. The module supports these modes for managing the Assigned Number subfield: explicit assignment, auto-assignment from a pool, and full auto-assignment.
grouping route-distinguisher
   +-- (rd-choice)?
      |   +--:(directly-assigned)
      |      |   +-- rd? rt-types:route-distinguisher
      |   +--:(directly-assigned-suffix)
      |      |   +-- rd-suffix? uint16
      |   +--:(auto-assigned)
      |      |   +-- rd-auto
      |      |      +-- (auto-mode)?
      |      |         |   +--:(from-pool)
      |      |         |      |   +-- rd-pool-name? string
      |      |         |      |   +--:(full-auto)
      |      |         |      |     |   ++ auto? empty
      |      |   ---ro auto-assigned-rd? rt-types:route-distinguisher
      |      |   +--:(auto-assigned-suffix)
      |      |      |   +-- rd-auto-suffix
      |      |      |      +-- (auto-mode)?
      |      |      |         |   +--:(from-pool)
      |      |      |         |      |   +-- rd-pool-name? string
      |      |      |         |      |   +--:(full-auto)
      |      |      |         |      |     |   ++ auto? empty
      |      |      |   ---ro auto-assigned-rd-suffix? uint16
      |      |   +--:(no-rd)
      |      |     ++ no-rd? empty

Figure 2: Route Distinguisher Grouping Subtree

'vpn-components-group':
   A YANG grouping that is used to group VPN nodes, VPN network accesses, or sites. For example, diversity or redundancy constraints can be applied on a per-group basis.

'placement-constraints':
   A YANG grouping that is used to define the placement constraints of a VPN node, VPN network access, or site.

'ports':
   A YANG grouping that defines ranges of source and destination port numbers and operators. The subtree of this grouping is depicted in Figure 3.
grouping ports
  +-- (source-port)?
    +--:(source-port-range-or-operator)
      +-- source-port-range-or-operator
      +-- (port-range-or-operator)?
        +--:(range)
          |  +-- lower-port inet:port-number
          |  +-- upper-port inet:port-number
          +--:(operator)
          |  +-- operator? operator
          |  +-- port inet:port-number
  +-- (destination-port)?
    +--:(destination-port-range-or-operator)
      +-- destination-port-range-or-operator
        +-- (port-range-or-operator)?
          +--:(range)
          |  +-- lower-port inet:port-number
          |  +-- upper-port inet:port-number
          +--:(operator)
          |  +-- operator? operator
          |  +-- port inet:port-number

Figure 3: Port Numbers Grouping Subtree

'qos-classification-policy':
A YANG grouping that defines a set of QoS classification policies based on various match Layer 3/4 and application criteria. The subtree of this grouping is depicted in Figure 4.

The QoS match criteria reuse groupings that are defined in the packet fields module "ietf-packet-fields" (Section 4.2 of [RFC8519]).

Any layer 4 protocol can be indicated in the 'protocol' data node under 'l3', but only TCP and UDP specific match criteria are elaborated in this version as these protocols are widely used in the context of VPN services. Future revisions can be considered to add other Layer 4 specific parameters (e.g., Stream Control Transmission Protocol [RFC4960]), if needed.
Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria shown in Figure 4, part of the TCP/UDP payload, or a combination thereof. This version of the module does not support such advanced match criteria. Future revisions of the module may consider adding match criteria based on the transport protocol payload (e.g., by means of a bitmask match).

```
grouping qos-classification-policy
    +-- rule* [id]
        |     +-- id                         string
        |     +-- (match-type)?
        |        +--:(match-flow)
        |           +-- (l3)?
        |           |     +--:(ipv4)
        |           |        +-- ipv4
        |           |            +-- dscp?                             inet:dscp
        |           |            +-- ecn?                              uint8
        |           |            +-- length?                           uint16
        |           |            +-- ttl?                              uint8
        |           |            +-- protocol?                         uint8
        |           |            +-- ihl?                              uint8
        |           |            +-- flags?                            bits
        |           |            +-- offset?                           uint16
        |           |            +-- identification?                   uint16
        |           |       +-- (destination-network)?
        |           |           +--:(destination-ipv4-network)
        |           |             +-- destination-ipv4-network?        inet:ipv4-prefix
        |           |       +-- (source-network)?
        |           |         +--:(source-ipv4-network)
        |           |             +-- source-ipv4-network?        inet:ipv4-prefix
        |           +--:(ipv6)
        |            +-- ipv6
        |               +-- dscp?                             inet:dscp
        |               +-- ecn?                              uint8
        |               +-- length?                           uint16
        |               +-- ttl?                              uint8
        |               +-- protocol?                         uint8
        |               +-- (destination-network)?
        |               |     +--:(destination-ipv6-network)
        |               |       +-- destination-ipv6-network?        inet:ipv6-prefix
        |               |       +-- (source-network)?
        |               |         +--:(source-ipv6-network)
        |               |           +-- source-ipv6-network?        inet:ipv6-prefix
        +--:(source-ipv4-network)?
```
4. Layer 2/3 VPN Common Module

This module uses types defined in [RFC6991], [RFC8294], and [RFC8519]. It also uses the extension defined in [RFC8341].

<CODE BEGINS> file "ietf-vpn-common@2021-09-10.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";
  }

Figure 4: QoS Classification Subtree

+-- target-class-id?
  +-- string {qos}?

+-- (destination-port)?
  +-- (destination-port-range-or-operator)
    +-- destination-port-range-or-operator
      +-- port
        +-- inet:port-number
      +-- (range)
        +-- lower-port
          +-- inet:port-number
        +-- upper-port
          +-- inet:port-number
      +-- (operator)
        +-- operator?    operator
        +-- port
          +-- inet:port-number
      +-- (match-application)
        +-- match-application?    identityref

Figure 4: QoS Classification Subtree
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types, Section 3";
}
import ietf-packet-fields {
  prefix packet-fields;
  reference
    "RFC 8519: YANG Data Model for Network Access
      Control Lists (ACLs)";
}

organization
  "IETF OPSAWG (Operations and Management Area Working Group)";
contact
  "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
  WG List:  <mailto:opsawg@ietf.org>
  Editor: Mohamed Boucadair
    <mailto:mohamed.boucadair@orange.com>
  Author: Samier Barguil
    <mailto:samier.barguilgiraldo.ext@telefonica.com>
  Author: Oscar Gonzalez de Dios
    <mailto:oscar.gonzalezdedios@telefonica.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) 2021 IETF Trust and the persons identified as
authors of the code. All rights reserved.

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

This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.";
revision 2021-09-10 {
description
    "Initial revision.";
reference
    "RFC XXXX: A Layer 2/3 VPN Common YANG Model";
}

/******* Collection of VPN-related Features *******/
/*
 * Features related to encapsulation schemes
 */

feature dot1q {
    description
        "Indicates the support for the Dot1q encapsulation.";
    reference
        "IEEE Std 802.1Q: Bridges and Bridged Networks";
}

feature qinq {
    description
        "Indicates the support for the QinQ encapsulation.";
    reference
        "IEEE Std 802.1ad: Provider Bridges";
}

feature vxlan {
    description
        "Indicates the support for the Virtual eXtensible Local Area Network (VXLAN) encapsulation.";
    reference
        "RFC 7348: Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks";
}

feature qinany {
    description
        "Indicates the support for the QinAny encapsulation. The outer VLAN tag is set to a specific value but the inner VLAN tag is set to any.";
}

feature lag-interface {
    description
        "Indicates the support for Link Aggregation Group (LAG) between VPN network accesses.";
    reference
        "IEEE Std. 802.1AX: Link Aggregation";
}
feature multicast {
    description
        "Indicates multicast capabilities support in a VPN.";
    reference
        "RFC 6513: Multicast in MPLS/BGP IP VPNs";
}

feature igmp {
    description
        "Indicates support for Internet Group Management Protocol (IGMP).";
    reference
        "RFC 1112: Host Extensions for IP Multicasting
        RFC 3376: Internet Group Management Protocol, Version 3";
}

feature mld {
    description
        "Indicates support for Multicast Listener Discovery (MLD).";
    reference
        "RFC 2710: Multicast Listener Discovery (MLD) for IPv6
        RFC 3810: Multicast Listener Discovery Version 2 (MLDv2)
            for IPv6";
}

feature pim {
    description
        "Indicates support for Protocol Independent Multicast (PIM).";
    reference
        "RFC 7761: Protocol Independent Multicast - Sparse Mode
            (PIM-SM): Protocol Specification (Revised)";
}

feature ipv4 {
    description
        "Indicates IPv4 support in a VPN. That is, IPv4 traffic
            can be carried in the VPN, IPv4 addresses/prefixes can
be assigned to a VPN network access, IPv4 routes can be installed for the CE/PE link, etc.

reference
"RFC 791: Internet Protocol"
}

feature ipv6 {
  description
  "Indicates IPv6 support in a VPN. That is, IPv6 traffic can be carried in the VPN, IPv6 addresses/prefixes can be assigned to a VPN network access, IPv6 routes can be installed for the CE/PE link, etc.";
  reference
  "RFC 8200: Internet Protocol, Version 6 (IPv6)"
}

/*
 * Features related to routing protocols
 */

feature rtg-ospf {
  description
  "Indicates support for the OSPF as the Provider Edge (PE)/Customer Edge (CE) routing protocol.";
  reference
  "RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)
RFC 6565: OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol"
}

feature rtg-ospf-sham-link {
  description
  "Indicates support for OSPF sham links.";
  reference
  "RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs), Section 4.2.7
RFC 6565: OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol, Section 5"
}

feature rtg-bgp {
  description
  "Indicates support for BGP as the PE/CE routing protocol.";
  reference
  "RFC 4271: A Border Gateway Protocol 4 (BGP-4)"
}
feature rtg-rip {
  description
    "Indicates support for RIP as the PE/CE routing protocol.";
  reference
    "RFC 2453: RIP Version 2
      RFC 2080: RIPvng for IPv6";
}

feature rtg-isis {
  description
    "Indicates support for IS-IS as the PE/CE routing protocol.";
  reference
    "ISO10589: Intermediate System to Intermediate System intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)";
}

feature rtg-vrrp {
  description
    "Indicates support for the Virtual Router Redundancy Protocol (VRRP) in CE/PE link.";
  reference
    "RFC 5798: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6";
}

feature bfd {
  description
    "Indicates support for Bidirectional Forwarding Detection (BFD) between the CE and the PE.";
  reference
    "RFC 5880: Bidirectional Forwarding Detection (BFD)";
}

/*
 * Features related to VPN service constraints
 */

feature bearer-reference {
  description
    "A bearer refers to properties of the CE-PE attachment that are below Layer 3. This feature indicates support for the bearer reference access constraint. That is, the reuse of a network connection that was already ordered to the service provider apart from the IP VPN site.";
}
feature placement-diversity {
  description
    "Indicates support for placement diversity constraints in the
    customer premises. An example of these constraints may be to
    avoid connecting a site network access to the same Provider
    Edge as a target site network access.";
}

/*
 * Features related to bandwidth and Quality of Service (QoS)
 */

feature qos {
  description
    "Indicates support for Classes of Service (CoSes) in the VPN.";
}

feature inbound-bw {
  description
    "Indicates support for the inbound bandwidth in a VPN. That is,
    support for specifying the download bandwidth from the service
    provider network to the VPN site. Note that the L3SM uses
    'input' to identify the same feature. That terminology should
    be deprecated in favor of the one defined in this module.";
}

feature outbound-bw {
  description
    "Indicates support for the outbound bandwidth in a VPN. That is,
    support for specifying the upload bandwidth from the VPN site
    to the service provider network. Note that the L3SM uses
    'output' to identify the same feature. That terminology should
    be deprecated in favor of the one defined in this module.";
}

/*
 * Features related to security and resilience
 */

feature encryption {
  description
    "Indicates support for encryption in the VPN.";
}

feature fast-reroute {
  description
    "Indicates support for fast rerouting in the VPN.";
}
"Indicates support for Fast Reroute (FRR) capabilities for a VPN site."
}

/*
 * Features related to advanced VPN options
 */

feature external-connectivity {
    description
        "Indicates support for the VPN to provide external connectivity (e.g., Internet, private or public cloud).";
    reference
        "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 11";
}

feature extranet-vpn {
    description
        "Indicates support for extranet VPNs. That is, the capability of a VPN to access a list of other VPNs."
    reference
        "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 1.1";
}

feature carriers-carrier {
    description
        "Indicates support for Carrier-of-Carrier VPNs."
    reference
        "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 9";
}

/*
 * Address family related identities
 */

identity address-family {
    description
        "Defines a type for the address family."
}

identity ipv4 {
    base address-family;
    description
        "Identity for IPv4 address family."
}
identity ipv6 {
    base address-family;
    description
        "Identity for IPv6 address family.";
}

identity dual-stack {
    base address-family;
    description
        "Identity for IPv4 and IPv6 address family.";
}

/*
 * Identities related to VPN topology
 */

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

identity any-to-any {
    base vpn-topology;
    description
        "Identity for any-to-any VPN topology. All VPN sites
        can communicate with each other without any restrictions.";
}

identity hub-spoke {
    base vpn-topology;
    description
        "Identity for Hub-and-Spoke VPN topology. All Spokes can
        communicate only with Hubs but not with each other. Hubs
        can communicate with each other.";
}

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 topologies where the role of the nodes
        is not strictly Hub or Spoke. The VPN topology is controlled by
the import/export policies. The custom topology reflects more complex VPN nodes such as VPN node that acts as Hub for certain nodes and Spoke to others."

*/
* Identities related to network access types
*/

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

identity point-to-point {
    base site-network-access-type;
    description
        "Point-to-point access type."
}

identity multipoint {
    base site-network-access-type;
    description
        "Multipoint access type."
}

identity irb {
    base site-network-access-type;
    description
        "Integrated Routing Bridge (IRB).
        Identity for pseudowire connections."
}

identity loopback {
    base site-network-access-type;
    description
        "Loopback access type."
}

/*@ 
* Identities related to operational and administrative status
*/

identity operational-status {
    description
        "Base identity for the operational status."
}
identity op-up {
  base operational-status;
  description
    "Operational status is Up/Enabled.";
}

identity op-down {
  base operational-status;
  description
    "Operational status is Down/Disabled.";
}

identity op-unknown {
  base operational-status;
  description
    "Operational status is Unknown.";
}

identity administrative-status {
  description
    "Base identity for administrative status.";
}

identity admin-up {
  base administrative-status;
  description
    "Administrative status is Up/Enabled.";
}

identity admin-down {
  base administrative-status;
  description
    "Administrative status is Down/Disabled.";
}

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

identity admin-pre-deployment {
  base administrative-status;
  description
    "Administrative status is pre-deployment phase. That is, prior to the actual deployment of a service.";
}
/*  * Identities related to site or node role  */

identity role {
  description
    "Base identity of a site or a node role.";
}

identity any-to-any-role {
  base role;
  description
    "Any-to-any role.";
}

identity spoke-role {
  base role;
  description
    "A node or a site is acting as a Spoke.";
}

identity hub-role {
  base role;
  description
    "A node or a site is acting as a Hub.";
}

identity custom-role {
  base role;
  description
    "VPN node with custom or complex role in the VPN. For some
    sources/destinations it can behave as a Hub, but for others it
    can act as a Spoke depending on the configured policy.";
}

/*
 * Identities related to VPN service constraints
 */

identity placement-diversity {
  description
    "Base identity for access placement constraints.";
}

identity bearer-diverse {
  base placement-diversity;
  description
    "Bearer diversity.";
The bearers should not use common elements.";

identity pe-diverse {
  base placement-diversity;
  description
    "PE diversity.";
}

identity pop-diverse {
  base placement-diversity;
  description
    "Point Of Presence (POP) diversity.";
}

identity linecard-diverse {
  base placement-diversity;
  description
    "Linecard diversity.";
}

identity same-pe {
  base placement-diversity;
  description
    "Having sites connected on the same PE.";
}

identity same-bearer {
  base placement-diversity;
  description
    "Having sites connected using the same bearer.";
}

/*
 * Identities related to service types
 */

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

identity l3vpn {
  base service-type;
  description
    "L3VPN service.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
identity vpls {
    base service-type;
    description "VPLS service.");
    reference
    "RFC 4761: Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling
    RFC 4762: Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling";
}

identity vpws {
    base service-type;
    description "Virtual Private Wire Service (VPWS) service.");
    reference
    "RFC 4664: Framework for Layer 2 Virtual Private Networks (L2VPNs), Section 3.1.1";
}

identity vpws-evpn {
    base service-type;
    description "EVPN used to support VPWS service.");
    reference
    "RFC 8214: Virtual Private Wire Service Support in Ethernet VPN";
}

identity pbb-evpn {
    base service-type;
    description "Provider Backbone Bridging (PBB) EVPNs service.");
    reference
    "RFC 7623: Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)";
}

identity mpls-evpn {
    base service-type;
    description "MPLS-based EVPN service.");
    reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN";
}

identity vxlan-evpn {
base service-type;
description
"VXLAN-based EVPN service."
reference
"RFC 8365: A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)";
}

/* Identites related to VPN signaling type */

identity vpn-signaling-type {
  description
  "Base identity for VPN signaling types";
}

identity bgp-signaling {
  base vpn-signaling-type;
  description
  "Layer 2 VPNs using BGP signaling.";
  reference
  "RFC 6624: Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling
RFC 7432: BGP MPLS-Based Ethernet VPN";
}

identity ldp-signaling {
  base vpn-signaling-type;
  description
  "Targeted Label Distribution Protocol (LDP) signaling.";
  reference
  "RFC 5036: LDP Specification";
}

identity l2tp-signaling {
  base vpn-signaling-type;
  description
  "Layer Two Tunneling Protocol (L2TP) signaling.";
  reference
  "RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3)";
}

/* Identites related to routing protocols */

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

identity static-routing {
  base routing-protocol-type;
  description
    "Static routing protocol.";
}

identity bgp-routing {
  if-feature "rtg-bgp";
  base routing-protocol-type;
  description
    "BGP routing protocol.";
  reference
    "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
}

identity ospf-routing {
  if-feature "rtg-ospf";
  base routing-protocol-type;
  description
    "OSPF routing protocol.";
  reference
    "RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)"
    "RFC 6565: OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol"
}

identity rip-routing {
  if-feature "rtg-rip";
  base routing-protocol-type;
  description
    "RIP routing protocol.";
  reference
    "RFC 2453: RIP Version 2"
    "RFC 2080: RIPng for IPv6"
}

identity isis-routing {
  if-feature "rtg-isis";
  base routing-protocol-type;
  description
    "IS-IS routing protocol.";
  reference
    "ISO10589: Intermediate System to Intermediate System intra-
identity vrrp-routing {
  if-feature "rtg-vrrp";
  base routing-protocol-type;
  description
    "VRRP protocol. This is to be used when LANs are directly connected to PEs.";
  reference
    "RFC 5798: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6";
}

identity direct-routing {
  base routing-protocol-type;
  description
    "Direct routing. This is to be used when LANs are directly connected to PEs and must be advertised in the VPN.";
}

identity any-routing {
  base routing-protocol-type;
  description
    "Any routing protocol. This can be, e.g., used to set policies that apply to any routing protocol in place.";
}

identity isis-level {
  if-feature "rtg-isis";
  description
    "Base identity for the IS-IS level.";
  reference
    "ISO10589: Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)";
}
identity level-1 {
    base isis-level;
    description
        "IS-IS level 1."
}

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

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

identity bfd-session-type {
    if-feature "bfd"
    description
        "Base identity for the BFD session type."
}

identity classic-bfd {
    base bfd-session-type;
    description
        "Classic BFD."
    reference
        "RFC 5880: Bidirectional Forwarding Detection (BFD)"
}

identity s-bfd {
    base bfd-session-type;
    description
        "Seamless BFD."
    reference
        "RFC 7880: Seamless Bidirectional Forwarding Detection (S-BFD)"
}

/*
 * Identities related to Routes Import and Export
 */

identity ie-type {
    description
        "Base identity for ‘import/export’ routing profiles. These profiles can be reused between VPN nodes."
}
identity import {
    base ie-type;
    description
    "'Import' routing profile.";
    reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 4.3.1";
}

identity export {
    base ie-type;
    description
    "'Export' routing profile.";
    reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), Section 4.3.1";
}

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

/*
 * Identities related to bandwidth and QoS
 */

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

identity inbound-bw {
    if-feature "inbound-bw";
    base bw-direction;
    description
    "Inbound bandwidth.";
}

identity outbound-bw {
    if-feature "outbound-bw";
    base bw-direction;
    description
    "Outbound bandwidth.";
}
identity bw-type {
    description
        "Base identity for the bandwidth type.";
}

identity bw-per-cos {
    if-feature "qos";
    base bw-type;
    description
        "The bandwidth is per-CoS.";
}

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

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

identity bw-per-service {
    base bw-type;
    description
        "The bandwidth is per-VPN service.";
}

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

identity site-to-wan {
    base qos-profile-direction;
    description
        "Customer site to provider’s network direction. This is typically the CE-to-PE direction.";
}

identity wan-to-site {
    base qos-profile-direction;
    description
        "Provider’s network to customer site direction.";
This is typically the PE-to-CE direction.

identity both {
    base qos-profile-direction;
    description "Both WAN-to-Site and Site-to-WAN directions.";
}

/*
 * Identities related to underlay transport instances
 */

identity transport-instance-type {
    description "Base identity for underlay transport instance type.";
}

identity virtual-network {
    base transport-instance-type;
    description "Virtual network.";
    reference "RFC 8453: Framework for Abstraction and Control of TE Networks (ACTN)";
}

identity enhanced-vpn {
    base transport-instance-type;
    description "Enhanced VPN (VPN+). VPN+ is an approach that is based on existing VPN and Traffic Engineering (TE) technologies but adds characteristics that specific services require over and above classical VPNs.";
    reference "I-D.ietf-teas-enhanced-vpn:
        A Framework for Enhanced Virtual Private Network (VPN+) Services";
}

identity ietf-network-slice {
    base transport-instance-type;
    description "IETF network slice. An IETF network slice is a logical network topology connecting a number of endpoints using a set of shared or dedicated network resources that are used to satisfy specific service objectives.";
}
identity protocol-type {
    description  "Base identity for Protocol Type.";
}

identity ip-in-ip {
    base protocol-type;
    description  "Transport is based on IP-in-IP.";
    reference    "RFC 2003: IP Encapsulation within IP
                 RFC 2473: Generic Packet Tunneling in IPv6 Specification";
}

identity ip-in-ipv4 {
    base ip-in-ip;
    description  "Transport is based on IP over IPv4.";
    reference    "RFC 2003: IP Encapsulation within IP";
}

identity ip-in-ipv6 {
    base ip-in-ip;
    description  "Transport is based on IP over IPv6.";
    reference    "RFC 2473: Generic Packet Tunneling in IPv6 Specification";
}

identity gre {
    base protocol-type;
    description  "Transport is based on Generic Routing Encapsulation (GRE).";
    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 gre-v4 {
    base gre;
    description "Transport is based on GRE over IPv4.";
    reference "RFC 1702: Generic Routing Encapsulation over IPv4 networks";
}

identity gre-v6 {
    base gre;
    description "Transport is based on GRE over IPv6.";
    reference "RFC 7676: IPv6 Support for Generic Routing Encapsulation (GRE)";
}

identity vxlan-trans {
    base protocol-type;
    description "Transport is based on VXLAN.";
    reference "RFC 7348: Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks";
}

identity geneve {
    base protocol-type;
    description "Transport is based on Generic Network Virtualization Encapsulation (GENEVE).";
    reference "RFC 8926: Geneve: Generic Network Virtualization Encapsulation";
}

identity ldp {
    base protocol-type;
    description "Transport is based on LDP.";
    reference "RFC 5036: LDP Specification";
}

identity mpls-in-udp {
    base protocol-type;
    description
"Transport is MPLS in UDP.";
reference
"RFC 7510: Encapsulating MPLS in UDP";
}

identity sr {
  base protocol-type;
  description
    "Transport is based on Segment Routing (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-mpls {
  base sr;
  description
    "Transport is based on SR with MPLS.";
  reference
    "RFC 8660: Segment Routing with the MPLS Data Plane"
}

identity srv6 {
  base sr;
  description
    "Transport is based on SR over IPv6.";
  reference
    "RFC 8754: IPv6 Segment Routing Header (SRH)"
}

identity sr-mpls-over-ip {
  base sr;
  description
    "Transport is based on SR over MPLS over IP.";
  reference
    "RFC 8663: MPLS Segment Routing over IP"
}

identity rsvp-te {
  base protocol-type;
  description
    "Transport setup relies upon RSVP-TE.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels"
}

identity bgp-lu {

base protocol-type;
description
  "Transport setup relies upon BGP-LU.";
reference
  "RFC 8277: Using BGP to Bind MPLS Labels to Address Prefixes";
}

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

/*
 * Identities related to encapsulations
 */

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

identity priority-tagged {
  base encapsulation-type;
  description
  "Priority-tagged interface.";
}

identity dot1q {
  if-feature "dot1q";
  base encapsulation-type;
  description
  "Dot1q encapsulation.";
}

identity qinq {
  if-feature "qinq";
  base encapsulation-type;
  description
  "QinQ encapsulation.";
}

identity qinany {
  if-feature "qinany";
  base encapsulation-type;
  description
  "QinAny encapsulation.";
}
identity vxlan {
  if-feature "vxlan";
  base encapsulation-type;
  description
    "VxLAN encapsulation.";
}

identity ethernet-type {
  base encapsulation-type;
  description
    "Ethernet encapsulation type.";
}

identity vlan-type {
  base encapsulation-type;
  description
    "VLAN encapsulation type.";
}

identity untagged-int {
  base encapsulation-type;
  description
    "Untagged interface type.";
}

identity tagged-int {
  base encapsulation-type;
  description
    "Tagged interface type.";
}

identity lag-int {
  if-feature "lag-interface";
  base encapsulation-type;
  description
    "LAG interface type.";
}

/*
 * Identities related to VLAN Tag
 */

identity tag-type {
  description
    "Base identity for the tag types.";
}

identity c-vlan {
base tag-type;

description
  "Indicates Customer VLAN (C-VLAN) tag, normally using
  the 0x8100 Ethertype.";
}

identity s-vlan {
  base tag-type;
  description
    "Indicates Service VLAN (S-VLAN) tag.";
}

identity s-c-vlan {
  base tag-type;
  description
    "Uses both an S-VLAN tag and a C-VLAN tag.";
}

/*
  * Identities related to VXLAN
  */

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

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

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

/*
  * Identities related to multicast
  */

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

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

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

identity ssm-tree-type {
    base multicast-tree-type;
    description
        "Source-Specific Multicast (SSM) tree type."
}

identity asm-tree-type {
    base multicast-tree-type;
    description
        "Any-Source Multicast (ASM) tree type."
}

identity bidir-tree-type {
    base multicast-tree-type;
    description
        "Bidirectional tree type."
}

identity multicast-rp-discovery-type {
    if-feature "multicast";
    description
        "Base identity for Rendezvous Point (RP) discovery type."
}

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

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

identity bsr-rp {
   base multicast-rp-discovery-type;
   description
   "Bootstrap Router (BSR) discovery type.";
}

identity group-management-protocol {
   if-feature "multicast";
   description
   "Base identity for multicast group management protocol.";
}

identity igmp-proto {
   base group-management-protocol;
   description
   "IGMP.";
   reference
   "RFC 1112: Host Extensions for IP Multicasting
RFC 3376: Internet Group Management Protocol, Version 3";
}

identity mld-proto {
   base group-management-protocol;
   description
   "MLD.";
   reference
   "RFC 2710: Multicast Listener Discovery (MLD) for IPv6
RFC 3810: Multicast Listener Discovery Version 2 (MLDv2)
for IPv6";
}

identity pim-proto {
   if-feature "pim";
   base routing-protocol-type;
   description
   "PIM.";
   reference
"RFC 7761: Protocol Independent Multicast - Sparse Mode
   (PIM-SM): Protocol Specification (Revised)";
}

identity igmp-version {
   if-feature "igmp";
   description
      "Base identity for IGMP version.";
}

identity igmpv1 {
   base igmp-version;
   description
      "IGMPv1.";
   reference
      "RFC 1112: Host Extensions for IP Multicasting";
}

identity igmpv2 {
   base igmp-version;
   description
      "IGMPv2.";
   reference
      "RFC 2236: Internet Group Management Protocol, Version 2";
}

identity igmpv3 {
   base igmp-version;
   description
      "IGMPv3.";
   reference
      "RFC 3376: Internet Group Management Protocol, Version 3";
}

identity mld-version {
   if-feature "mld";
   description
      "Base identity for MLD version.";
}

identity mldv1 {
   base mld-version;
   description
      "MLDv1.";
   reference
      "RFC 2710: Multicast Listener Discovery (MLD) for IPv6";
}
identity mldv2 {
    base mld-version;
    description
        "MLDv2.";
    reference
        "RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for IPv6";
}

/*
 * Identities related to traffic types
 */

identity tf-type {
    description
        "Base identity for the traffic type.";
}

identity multicast-traffic {
    base tf-type;
    description
        "Multicast traffic.";
}

identity broadcast-traffic {
    base tf-type;
    description
        "Broadcast traffic.";
}

identity unknown-unicast-traffic {
    base tf-type;
    description
        "Unknown unicast traffic.";
}

/*
 * Identities related to customer applications
 */

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

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

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

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

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

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

identity games {
    base customer-application;
    description
        "Gaming application.";
}

identity p2p {
    base customer-application;
    description
        "Peer-to-peer application.";
}

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

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

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

identity embb {
    base customer-application;
    description
       "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
       "Ultra-Reliable and Low Latency Communications
(URLLC) application. Note that an 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
       "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 bundling-type {
    description
       "The base identity for the bundling type. It supports a subset or
all CE-VLANs associated with an L2VPN service."
}

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

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

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

/*
 * Identities related to Ethernet Services
 */

identity control-mode {
  description
  "Base Identity for the type of control mode on Layer 2
  Control Protocol (L2CP).";
}

identity peer {
  base control-mode;
description
  "'peer' mode, i.e., participate in the protocol towards the CE.
  Peering is common for Link Aggregation Control Protocol (LACP)
  and the Ethernet Local Management Interface (E-LMI) and,
ocasionally, for Link Layer Discovery Protocol (LLDP).
  For VPLSs and VPWSs, the subscriber can also request that the
  peer service provider enables spanning tree.";
}

identity tunnel {
  base control-mode;
description
  "'tunnel' mode, i.e., pass to the egress or destination site. For
  Ethernet Private Lines (EPLs), the expectation is that L2CP
  frames are tunnelled.";
}
identity discard {
  base control-mode;
  description
    "Discard' mode, i.e., discard the frame."
} 

identity neg-mode {
  description
    "Base identity for the negotiation mode."
} 

identity full-duplex {
  base neg-mode;
  description
    "Full-duplex negotiation mode."
} 

identity auto-neg {
  base neg-mode;
  description
    "Auto-negotiation mode."
} 

/****** Collection of VPN-related Types ******/ 
typedef vpn-id {
  type string;
  description
    "Defines an identifier that is used with a VPN module. This can be, for example, a service identifier, a node identifier, etc."
} 

/****** VPN-related reusable groupings ******/

grouping vpn-description {
  description
    "Provides common VPN information."
  leaf vpn-id {
    type vpn-common:vpn-id;
    description
      "A VPN identifier that uniquely identifies a VPN. This identifier has a local meaning, e.g., within a service provider network."
  }
  leaf vpn-name {
    type string;
    description
}
"Used to associate a name with the service in order to facilitate the identification of the service."

leaf vpn-description {
  type string;
  description "Textual description of a VPN."
}

leaf customer-name {
  type string;
  description "Name of the customer that actually uses the VPN."
}

grouping vpn-profile-cfg {
  description "Grouping for VPN Profile configuration."
  container valid-provider-identifiers {
    description "Container for valid provider profile identifiers."
    list external-connectivity-identifier {
      if-feature "external-connectivity";
      key "id";
      description "List for profile identifiers that uniquely identify profiles governing how external connectivity is provided to a VPN. A profile indicates the type of external connectivity (Internet, cloud, etc.), the sites/nodes that are associated with a connectivity profile, etc. A profile can also indicate filtering rules and/or address translation rules. Such features may involve PE, P, or dedicated nodes as a function of the deployment."
      leaf id {
        type string;
        description "Identification of an external connectivity profile. The profile only has significance within the service provider’s administrative domain."
      }
    }
    list encryption-profile-identifier {
      key "id";
      description "List for encryption profile identifiers."
      leaf id {
        type string;
      }
    }
  }
}
description
   "Identification of the encryption profile to be used. The
   profile only has significance within the service provider’s
   administrative domain."
)
list qos-profile-identifier {
  key "id";
  description
   "List for QoS Profile Identifiers.";
  leaf id {
    type string;
    description
     "Identification of the QoS profile to be used. The
     profile only has significance within the service provider’s
     administrative domain."
  }
}
list bfd-profile-identifier {
  key "id";
  description
   "List for BFD profile identifiers.";
  leaf id {
    type string;
    description
     "Identification of the BFD profile to be used. The
     profile only has significance within the service provider’s
     administrative domain."
  }
}
list forwarding-profile-identifier {
  key "id";
  description
   "List for forwarding profile identifiers.";
  leaf id {
    type string;
    description
     "Identification of the forwarding profile to be used.
     The profile only has significance within the service
     provider’s administrative domain."
  }
}
list routing-profile-identifier {
  key "id";
  description
   "List for Routing Profile Identifiers.";
  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 referring VRF’s import/export policies.

The profile only has significance within the service
provider’s administrative domain.";

nacm:default-deny-write;


grouping oper-status-timestamp {
  description
    "This grouping defines some operational parameters for the
    service.";
  leaf status {
    type identityref {
      base operational-status;
    }
    config false;
    description
      "Operations status.";
  }
  leaf last-change {
    type yang:date-and-time;
    config false;
    description
      "Indicates the actual date and time of the service status
      change.";
  }
}

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

grouping underlay-transport {
  description
    "This grouping defines the type of underlay transport for the VPN service or how that underlay is set. It can include an identifier to an abstract transport instance to which the VPN is grafted or indicate a technical implementation that is expressed as an ordered list of protocols."
  choice type {
    description
        "A choice based on the type of underlay transport constraints."
    case abstract {
      description
        "Indicates that the transport constraint is an abstract concept."
      leaf transport-instance-id {
        type string;
        description
          "An optional identifier of the abstract transport instance."
      }
    }
    leaf instance-type {
      type identityref {
        base transport-instance-type;
      }
      description
        "Indicates a transport instance type. For example, it can be a VPN+, an IETF network slice, a virtual network, etc."
    }
  }
  case protocol {
    description

leaf-list protocol {
  type identityref {
    base protocol-type;
  }
  ordered-by user;
  description
    "A client ordered list of transport protocols.";
}

grouping vpn-route-targets {
  description
    "A grouping that specifies Route Target (RT) import-export rules
    used in a BGP-enabled VPN.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)
    RFC 4664: Framework for Layer 2 Virtual Private Networks
    (L2VPNs)";
  list vpn-target {
    key "id";
    description
      "Route targets. AND/OR operations may be defined
      based on the RTs assignment.";
    leaf id {
      type uint8;
      description
        "Identifies each VPN Target.";
    }
  }
  list route-targets {
    key "route-target";
    description
      "List of RTs.";
    leaf route-target {
      type rt-types:route-target;
      description
        "Conveys an RT value.";
    }
  }
  leaf route-target-type {
    type rt-types:route-target-type;
    mandatory true;
    description
      "Import/export type of the RT.";
  }
}
container vpn-policies {
    description
        "VPN service policies. It contains references to the import and export policies to be associated with the VPN service.";
    leaf import-policy {
        type string;
        description
            "Identifies the 'import' policy.";
    }
    leaf export-policy {
        type string;
        description
            "Identifies the 'export' policy.";
    }
}

grouping route-distinguisher {
    description
        "Grouping for route distinguisher (RD).";
    choice rd-choice {
        description
            "Route distinguisher choice between several options on providing the route distinguisher value.";
        case directly-assigned {
            description
                "Explicitly assign an RD value.";
            leaf rd {
                type rt-types:route-distinguisher;
                description
                    "Indicates an RD value that is explicitly assigned.";
            }
        }
        case directly-assigned-suffix {
            description
                "The value of the Assigned Number subfield of the RD. The Administrator subfield of the RD will be based on other configuration information such as router-id or ASN.";
            leaf rd-suffix {
                type uint16;
                description
                    "Indicates the value of the Assigned Number subfield that is explicitly assigned.";
            }
        }
    }
}
case auto-assigned {
  description
  "The RD is auto-assigned."
  container rd-auto {
    description
    "The RD is auto-assigned."
    choice auto-mode {
      description
      "Indicates the auto-assignment mode. RD can be automatically assigned with or without indicating a pool from which the RD should be taken.

      For both cases, the server will auto-assign an RD value 'auto-assigned-rd' and use that value operationally."
      case from-pool {
        leaf rd-pool-name {
          type string;
          description
          "The auto-assignment will be made from the pool identified by the rd-pool-name."
        }
      }
      case full-auto {
        leaf auto {
          type empty;
          description
          "Indicates an RD is fully auto-assigned."
        }
      }
    }
  }
  leaf auto-assigned-rd {
    type rt-types:route-distinguisher;
    config false;
    description
    "The value of the auto-assigned RD."
  }
}
}

leaf auto-assigned-suffix {
  description
  "The value of the Assigned Number subfield will be auto-assigned. The Administrator subfield will be based on other configuration information such as router-id or ASN."
  container rd-auto-suffix {
    description
"The Assigned Number subfield is auto-assigned.";
choice auto-mode {
    description
    "Indicates the auto-assignment mode of the Assigned Number
    subfield. This number can be automatically assigned
    with or without indicating a pool from which the value
    should be taken.
    
    For both cases, the server will auto-assign
    'auto-assigned-rd-suffix' and use that value to build
    the RD that will be used operationally.";
    case from-pool {
        leaf rd-pool-name {
            type string;
            description
            "The assignment will be made from the pool identified
            by the rd-pool-name.";
        }
    }
    case full-auto {
        leaf auto {
            type empty;
            description
            "Indicates that the Assigned Number is fully auto
            assigned.";
        }
    }
}
leaf auto-assigned-rd-suffix {
    type uint16;
    config false;
    description
    "Includes the value of the Assigned Number subfield that
    is auto-assigned.";
}
}
}

case no-rd {  
    description
    "Use the empty type to indicate RD has no value and is not to
    be auto-assigned.";
    leaf no-rd {
        type empty;
        description
        "No RD is assigned.";
    }
}
}
grouping vpn-components-group {
  description
  "Grouping definition to assign group-ids to associate VPN nodes, 
  sites, or network accesses.";
  container groups {
    description
    "Lists the groups to which a VPN node, a site, or a network 
    access belongs to.";
    list group {
      key "group-id";
      description
      "List of group-ids.";
      leaf group-id {
        type string;
        description
        "Is the group-id to which a VPN node, a site, or a network 
        access belongs to.";
      }
    }
  }
}

grouping placement-constraints {
  description
  "Constraints for placing a network access.";
  list constraint {
    key "constraint-type";
    description
    "List of constraints.";
    leaf constraint-type {
      type identityref {
        base placement-diversity;
      }
      description
      "Diversity constraint type.";
    }
    container target {
      description
      "The constraint will apply against this list of groups.";
      choice target-flavor {
        description
        "Choice for the group definition.";
        case id {
          list group {
            key "group-id";
            description
          }
        }
      }
    }
  }
}
leaf group-id {
  type string;
  description
      "The constraint will apply against this particular group-id.";
}

case all-accesses {
  leaf all-other-accesses {
    type empty;
    description
      "The constraint will apply against all other network accesses of a site.";
  }
}

case all-groups {
  leaf all-other-groups {
    type empty;
    description
      "The constraint will apply against all other groups that the customer is managing.";
  }
}

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

description

"Destination port definition."
uses packet-fields:port-range-or-operator;
}
}
}

grouping qos-classification-policy {
	description
	"Configuration of the traffic classification policy."

table rule {
	key "id";
	ordered-by user;
	description
	"List of marking rules."
	leaf id {
	yype string;
	description
	"An identifier of the QoS classification policy rule."
}
choice match-type {
	default "match-flow";
	description
	"Choice for classification."

case match-flow {
	option 13 {
		description
	"Either IPv4 or IPv6."
	oncontainer ipv4 {
		description
	"Rule set that matches IPv4 header."
	uses packet-fields:acl-ip-header-fields;
	uses packet-fields:acl-ipv4-header-fields;
	}
	container ipv6 {
		description
	"Rule set that matches IPv6 header."
	uses packet-fields:acl-ip-header-fields;
	uses packet-fields:acl-ipv6-header-fields;
	}
}
	option 14 {
		description
	"Includes Layer 4 specific information.
This version focuses on TCP and UDP."
	oncontainer tcp {
		description
	"Rule set that matches TCP header."
}
uses packet-fields:acl-tcp-header-fields;
uses ports;
}
container udp {
  description
    "Rule set that matches UDP header."
  uses packet-fields:acl-udp-header-fields;
  uses ports;
}
}
case match-application {
  leaf match-application {
    type identityref {
      base customer-application;
    }
    description
    "Defines the application to match."
  }
}
leaf target-class-id {
  if-feature "qos";
  type string;
  description
  "Identification of the class of service. This identifier is internal to the administration.";
}
}

<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. 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.

Modules that use the groupings that are defined in this document should identify the corresponding security considerations. For example, reusing some of these groupings will expose privacy-related information (e.g., customer-name). Disclosing such information may be considered as a violation of the customer-provider trust relationship.

6. IANA Considerations

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

   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
   maintained by IANA: N
   prefix: vpn-common
   reference: RFC XXXX

7. Acknowledgements

During the discussions of this work, helpful comments and reviews were received from (listed alphabetically): Alejandro Aguado, Raul Arco, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Tom Petch, Erez Segev, and Paul Sherratt. Many thanks to them.

This work is partially supported by the European Commission under Horizon 2020 grant agreement number 101015857 Secured autonomic traffic management for a Tera of SDN flows (Teraflow).
Many thanks to Radek Krejci for the yangdoctors review, Wesley Eddy for the tsvart review, Ron Bonica and Victoria Pritchard for the Rtgdir review, Joel Halpern for the genart review, Tim Wicinski for the opsdir review, and Suresh Krishnan for the intdir review.

Special thanks to Robert Wilton for the AD review.

Thanks to Roman Danyliw, Lars Eagert, Warren Kumari, Erik Kline, Zaheduzzaman Sarker, Benjamin Kaduk, and Eric Vyncke for the IESG review.

8. 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

9. References

9.1. Normative References


9.2. Informative References

[I-D.ietf-opsawg-l2nm]
[I-D.ietf-opsawg-l3sm-l3nm]
Barguil, S., Dios, O. G. D., Boucadair, M., Munoz, L. A.,
and A. Aguado, "A Layer 3 VPN Network YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-l3sm-l3nm-15,

[I-D.ietf-teas-actn-vn-yang]
Lee, Y., Dhody, D., Ceccarelli, D., Bryskin, I., and B. Y.
Yoon, "A YANG Data Model for VN Operation", Work in Progress, Internet-Draft, draft-ietf-teas-actn-vn-yang-12,

[I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A
Framework for Enhanced Virtual Private Network (VPN+)
Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-08, 12 July 2021,

[I-D.ietf-teas-ietf-network-slices]
Farrel, A., Gray, E., Drake, J., Rokui, R., Homma, S.,
Makhijani, K., Contreras, L. M., and J. Tantsura,

[IEEE802.1ad]
"Virtual Bridged Local Area Networks Amendment 4: Provider

[IEEE802.1AX]

[IEEE802.1Q]
"Bridges and Bridged Networks", IEEE Std 802.1Q-2018, 6
July 2018.

[ISO10589] ISO, "Intermediate System to Intermediate System intra-
domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)", 2002,


Barguil, et al. Expires 2 April 2022 [Page 65]


Appendix A. Example of Common Data Nodes in Early L2NM/L3NM Designs

In order to avoid data nodes duplication and to ease passing data among layers (i.e., from the service layer to the network layer and vice versa), early versions of the L3NM reused many of the data nodes that are defined in the L3SM. 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 to customers.

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 L2NM is shown in Figure 5. Such data nodes reuse was interpreted as if the deployment of the L2NM requires the support of the L3NM; which is not required.

module 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 5: Excerpt from the L2NM YANG Module

Authors’ Addresses

Barguïl, et al. Expires 2 April 2022
Abstract

Software bills of materials (SBOMs) are formal descriptions of what pieces of software are included in a product. This memo specifies a means for manufacturers to state how SBOMs may be retrieved through an extension to manufacturer usage descriptions (MUD).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 19, 2020.

Copyright Notice

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

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

Manufacturer Usage Descriptions (MUD) [RFC8520] provides a means for devices to identify what they are and what sort of network access they need. This memo specifies a YANG model [RFC6991] for reporting and a means for transmitting the report, and appropriate extensions to the MUD file to indicate how to report and how often.

Software bills of material (SBOMs) are descriptions of what software, including versioning and dependencies, a device contains. There are different SBOM formats such as Software Package Data Exchange [SPDX] and Software Identity Tags [SWID].

This memo extends the MUD YANG schema to provide location information of an SBOM.

These SBOMs are typically found in one of three ways:

- 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?
o Are there other retrieval mechanisms that need to be specified?
o Do we need to be more specific in how to authenticate and retrieve SBOMs?
o What are the implications if the MUD URL is an extension in a certificate (e.g. an IDevID cert)?

2. The mud-sbom extension model extension

We now formally define this extension. This is done in two parts. First, the extension name "sbom" is listed in the "extensions" array of the MUD file.

Second, the "mud" container is augmented with a list of SBOM sources. This is done as follows:

```yaml
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

```yaml
<CODE BEGINS>file "ietf-mud-sbom@2020-03-06.yang"
module ietf-mud-sbom {
  yang-version 1.1;
  prefix mud-sbom;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-mud {
    prefix mud;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact

  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
```
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.


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
4.2. Located on the Device

```json
{
  "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",
      }
    ]
  }
}
```
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-request@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-frto",
"matches": {
"ipv4": {
"ietf-acldns:src-dnsname": "cloud-service.example.com"
}
},
"actions": {
"forwarding": "accept"
}
}
}
}
]
"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"
        }
      }
    ]
  }]
}
"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]:
7. References

7.1. Normative References


7.2. Informative References


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
IP Flow Information Export (IPFIX) Information Elements Extension for Forwarding Exceptions
draft-mvmd-opsawg-ipfix-fwd-exceptions-04

Abstract

This draft proposes a couple of new forwarding exceptions related Information Elements (IEs) and Templates for the IP Flow Information Export (IPFIX) protocol. These new Information Elements and Exception Template can be used to export information about any forwarding errors in a network. This essential information is adequate to correlate packet drops to any control plane entity and map it to an impacted service. Once exceptions are correlated to a particular entity, an action can be assigned to mitigate such problems essentially enabling self-driving networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

All networks are susceptible to traffic drops due to a number of factors. Traffic drops can go unnoticed unless they are service impacting. In a multi-layered network architecture, it is tedious manual work to localize and root cause traffic blackholing issues. Transient drops are even harder to detect. Existing methodologies that rely on periodically monitoring interfaces on several hosts in a network does not guarantee timely detection, and are not scalable for large networks.
In order to eliminate this tedious monitoring work-flow, objective is to simplify localization and build correlation of dropped packets to particular entity. The network entity shall identify the dropped packets by monitoring dropped counters or doing a deep packet inspection of the packet discarded by the forwarding ASIC. The implementation of the method used to detect the drop is outside the scope of this document. Dropped packets will be sampled in the forwarding-path and sent to a host or software queue along with type of exception, in/out interface information and other relevant meta data. This will be a push model where the node encountering the error will emit the information about dropped packets and associated meta-data. Techniques for IP Packet Selection [RFC5475] describes Sampling and Filtering techniques for IP packet selection either using Systematic Sampling or Random Sampling.

The IPFIX Protocol Specification [RFC7011] defines a generic exchange mechanism for collecting flow information. It supports source-triggered export of information via the push model approach. The IPFIX Information Model [IANA-IPFIX] defines a list of standard Information Elements (IEs) which can be carried by the IPFIX protocol.

This document focuses on telemetry information for dropped packet exceptions, and proposes an extension to IPFIX message format for collecting sampled exceptions. Some of the IPFIX Information Elements (IEs) already exist, some will be defined along with corresponding formats. It is also possible to achieve sampling of the dropped packets by using sampling methods like SFLOW but details of other sampling methods are outside the scope of this document.

1.1. Terminology

IPFIX-specific terminology (e.g. Information Element, Template, Template Record, Options Template Record, Template Set, Collector, Exporter, Data Record) used in this document is defined in Section 2 of [RFC7011]. As in [RFC7011] these IPFIX-specific terms have the first letter of a word capitalized. This document also makes use of the same terminology and definitions as Section 2 of [RFC5470].

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. Scope

This document specifies the information model used for reporting packet-based forwarding exceptions. [RFC7011] provides guidance on the choices of the transport protocols used for IPFIX and their effects. Encoded IPFIX exception packets need to be reliably transported to the collector. The choice of the actual transport protocol is beyond the scope of this document.

This document assumes that all devices reporting exceptions will use existing IPFIX framework/module to send encoded packets to the collector. This would mean that the network device will specify the template that it is going to use for each of the events. The templates can be of varying length, and there could be multiple templates that a network device could use to encode the exceptions.

The implementation details of the collector application are beyond the scope of this document.

3. Information Elements

The Exception template could contain a subset of the IEs shown in Table 1, depending upon the exception reported.

Whenever packet drop happens inside forwarding plane, following information is key to understanding the issue: reason for packet drop, flow which encountered the drop (packet content), additional meta-data e.g. flow direction (ingress/egress), nexthop index, input interface, output interface, etc. on which this packet was flowing.

The following table includes all the existing IEs that a device reporting IPFIX Exceptions using various Exception Templates would typically need. The formats of IEs and IPFIX IDs are listed in the table below.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Size (bits)</th>
<th>IANA IPFIX ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flowDirection</td>
<td>8</td>
<td>61</td>
<td>The direction of the Flow observed at Observation point.</td>
</tr>
<tr>
<td>ingressInterface</td>
<td>32</td>
<td>10</td>
<td>Index of IP interface where packets of this flow are being received.</td>
</tr>
<tr>
<td>egressInterface</td>
<td>32</td>
<td>14</td>
<td>Index of IP interface where packets of this flow are being sent.</td>
</tr>
<tr>
<td>dataLinkFrameSize</td>
<td>16</td>
<td>312</td>
<td>Specified length of data link frame.</td>
</tr>
<tr>
<td>dataLinkFrameSection</td>
<td>65535</td>
<td>315</td>
<td>Carries n octets from data link frame of selected frame.</td>
</tr>
<tr>
<td>commonPropertiesID</td>
<td>64</td>
<td>137</td>
<td>Identifier of a set of common properties that is unique per observation domain.</td>
</tr>
</tbody>
</table>

Table 1: Forwarding Exception Information Elements
4. New Information Elements

4.1. Proposed New Information Elements

The proposed new IEs that a device reporting Exceptions using Exception template would need are listed in Table 2 below.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Abstract Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwardingExceptionCode</td>
<td>unsigned32</td>
<td>Unique code for every exception</td>
</tr>
<tr>
<td>forwardingNextHopID</td>
<td>unsigned64</td>
<td>Forwarding NH - index associated with packet that encountered this exception</td>
</tr>
<tr>
<td>forwardingLookupType</td>
<td>unsigned8</td>
<td>Last Lookup type performed on the packet in ingress path. For instance, IPV4, IPV6, Bridge, MPLS, Unknown etc.</td>
</tr>
<tr>
<td>underlyingIngressInterface</td>
<td>unsigned32</td>
<td>Underlying interface from which a packet arrived in ingress path. For instance, child interface of aggregate interface on which packet came in ingress; where aggregate interface is captured in ingressInterface</td>
</tr>
</tbody>
</table>

Table 2: New Information Elements
Figure 2: New Information Elements

The Information Elements defined in Table 2 are proposed to be incorporated into the IANA IPFIX Information Elements registry [IANA-IPFIX]

4.2. Definition of Exceptions

Every network will encounter issues like packet loss, from time to time. Some of the causes for such a loss of traffic or a block in transmission of data packets include overloaded system conditions, misconfiguration, profiles and policies that restrict the bandwidth or priority of traffic, network outages, or disruption with physical cable faults. Packet loss could also happen because of incorrect stitching of the forwarding path or a mismatch between control plane and data plane state. Exception code entails the reason/error code due to which this packet has been dropped.

4.2.1. forwardingExceptionCode

forwardingExceptionCode will be defined in "IPFIX Information Elements" registry. This list can be expanded in the future as necessary. The data record will have corresponding exception code value to indicate forwarding error that caused the traffic drop.

An implementation may choose to encode device internal exception codes as forwardingExceptionCode. In such scenarios, Enterprise Bit MUST be set to 1 and corresponding Enterprise Number MUST be present as described in [RFC7011]

There is an existing IE 89 - forwardingStatus [IANA-IPFIX] but it allows a very limited number of exceptions to be reported from the system (6-bit reason code). The exception codes also need to be standardized for use. Different forwarding ASICs would have different pipelines and hence discard reasons (which could be very specific to that pipeline) cannot be generalized. Hence it makes sense to have a standalone IE for reporting exception which not only provides support to report larger number of exceptions but also provides freedom for reporting application specific exceptions using the enterprise bit.

forwardingExceptionCode will also describe status of the flow with first two bits. An implementation may choose to export forwardingExceptionCode instead of IE89 - forwardingStatus.

A list of commonly used forwarding Exception codes will be identified and listed as part of Table 3 below.
<table>
<thead>
<tr>
<th>Forwarding Exception Code</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FIREWALL_DISCARD</td>
</tr>
<tr>
<td>2</td>
<td>TTL_EXPIRY</td>
</tr>
<tr>
<td>3</td>
<td>DISCARD_ROUTE</td>
</tr>
<tr>
<td>4</td>
<td>BAD_IPV4_CHECKSUM</td>
</tr>
<tr>
<td>5</td>
<td>REJECT_ROUTE</td>
</tr>
<tr>
<td>6</td>
<td>BAD_IPV4_HEADER (Version incorrect or IHL &lt; 5)</td>
</tr>
<tr>
<td>7</td>
<td>BAD_IPV6_HEADER (Version incorrect)</td>
</tr>
<tr>
<td>8</td>
<td>BAD_IPV4_HEADER_LENGTH (V4 frame is too short)</td>
</tr>
<tr>
<td>9</td>
<td>BAD_IPV6_HEADER_LENGTH</td>
</tr>
<tr>
<td>10</td>
<td>BAD_IPV6_OPTIONS_PACKET (too many option headers)</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

Figure 3: Table 3: Exception Codes

4.2.2. forwardingNextHopId

In terms of a network device, next hop is the gateway to which packet should be forwarded corresponding to the path to final destination. A given router doesn’t need to store the entire forwarding path information for a destination. As long as it can identify the next hop to be used for forwarding to a destination, the end to end forwarding can happen. This helps reduce size of forwarding table. The nexthop index uniquely identifies the egress path a packet would take to reach the destination. This could include information about the outgoing interface, layer 2 address to be used, forwarding features configured for the packet path etc.

For instance, consider we have a L3VPN topology like below

CE1 -------- PE1 ----- MPLS Network ----- PE2 ------- CE2

Figure 4: Figure 1: MPLS VPN Network

Figure 1 above illustrates an example where reporting of exception can provide an insight into the error scenario. CE1 and CE2 communicate with each other over an MPLS VPN network. The labels are typically advertised using protocols like RSVP or LDP. When a packet is received from core network on PE1, a lookup on MPLS label results in packet getting forwarded towards CE1. The entries in MPLS table are populated by corresponding protocol. If label entries don’t get populated in the MPLS table due to a probable glitch in the protocol configuration or some software inconsistency, the packets traversing on that LSP tunnel path shall get discarded on PE1.
In case of route lookups, that result in hierarchical forwarding chains, the mis-programming may manifest at different levels of the forwarding structure. The forwarding lookup may fail on any level of the hierarchy in the forwarding chain. It is expected that software at least report the nexthop where the lookup terminates. Its desirable for software to report the top level nexthop in the chain.

Using the mechanism described in this RFC, it will be possible to capture such packets and report them in IPFIX format with corresponding exception set (eg. DISCARD_ROUTE) along with relevant packet bytes and meta-data. This can help the operator/software to immediately understand root cause of the problem and take appropriate action.

An implementation may choose to report linecard number, linecard type, forwarding ASIC type and forwarding ASIC number on which an exception occurs, but mechanism to export these fields is out of the scope of this document.

4.2.3. forwardingLookupType

A packet might undergo multiple lookups in the forwarding chain. Lookup may fail at any level of the lookup hierarchy. When an exception is reported in such cases, type of the last lookup performed on the packet may help in identifying nature of the erroneous path.

For instance, a Firewall Discard may happen for Layer2 or Layer3 packet. All such packets may be treated as FIREWALL_DISCARD for generic exception reporting purposes. However, exact place of error in the pipeline (IPV4, IPV6, MPLS etc.) may help with easily correlating the exception.

4.2.4. underlyingIngressInterface

A packet can arrive on an aggregate ethernet(ae) interface where the receive interface is the ae but actual physical interface is a child member of this ae. If such a packet gets dropped because of an exception, it will be very useful not only to know about the ae on which it arrived but also the child link of that ae on which the packet was received.

underlyingIngressInterface represents the interface underlying the received interface (which in case of ae would be its child link) on which the packet arrived in ingress. This helps in providing more context about the nature of the packet processing for this path.
5. Exception Templates

This section presents a list of templates for reporting exceptions using newly proposed IEs in addition to few existing Information Elements (IEs).

Templates listed below are sample templates to demonstrate the utility of newly introduced Information Elements in conjunction with existing Information Elements to report meaningful data to the collector. A specific implementation may add or remove Information Elements from below templates based on their reporting requirements.

5.1. IPFIX Exception Template 1 for Forwarding Exceptions

Exception Template defined in Figure 1 demonstrates a sample set of data to export forwarding Exceptions.

```
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| Set ID = 2                                | Length = N octets                         |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| Template ID = 256                         | Field Count = N                           |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| forwardingExceptionCode                   | Field Length = 4                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| forwardingNextHopId                       | Field Length = 8                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| forwardingLookupType                      | Field Length = 1                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| flowDirection                             | Field Length = 1                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| ingressInterface                          | Field Length = 4                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| egressInterface                           | Field Length = 4                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| dataLinkFrameSize                         | Field Length = 2                          |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| dataLinkFrameSection                      | Field Length = 65535                       |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
| Padding (opt)                             |                                           |
+-------------------------------------------+-------------------------------------------+-------------------------------------------+
```

Figure 5: IPFIX Exception Template for Forwarding Exceptions
5.2. IPFIX Exception Template 2 for Forwarding Exceptions

Alternatively, Exception Template defined in Figure 2 is a sample template to export forwarding exceptions. This template demonstrates the use of Information Element 137 to represent following fields: forwardingExceptionCode, forwardingNexthopId, ingressInterface, underlyingIngressInterface and egressInterface.

```
+-----------------+------------------+
|      Set ID = 2  |      Length = N  |
|-----------------+------------------|
+-----------------+------------------+
|        Template ID = 256 | Field Count = N |
|-----------------+------------------|
|0| commonPropertiesId1 | Field Length = 4 |
+-----------------+------------------+
|0| flowDirection | Field Length = 1 |
+-----------------+------------------+
|0| forwardingLookupType | Field Length = 1 |
+-----------------+------------------+
|0| commonPropertiesId2 | Field Length = 8 |
+-----------------+------------------+
|0| commonPropertiesId3 | Field Length = 8 |
+-----------------+------------------+
|0| commonPropertiesId4 | Field Length = 8 |
+-----------------+------------------+
|0| commonPropertiesId5 | Field Length = 8 |
+-----------------+------------------+
|0| dataLinkFrameSize | Field Length = 2 |
+-----------------+------------------+
|0| dataLinkFrameSection | Field Length = 65535 |
+-----------------+------------------+
|                      Padding (opt)                      |
+-----------------+------------------+
```

Figure 6: IPFIX Exception Template 2 for Forwarding Exceptions

6. IANA Considerations

6.1. Information Elements

IANA manages the IPFIX Information Elements registry at [IANA-IPFIX]. This document introduces two new IPFIX Information Elements.

Name: forwardingExceptionCode ElementID: TBD Description: Exception code is an identifier uniquely describing cause of irregularity or traffic drop on a device. Abstract Data Type: unsigned32 Data Type Semantics: identifier
Name: forwardingNexthopId ElementID: TBD Description: Nexthop ID is a unique identifier for a Nexthop on a device. Abstract Data Type: unsigned64 Data Type Semantics: identifier

Name: forwardingLookupType ElementID: TBD Description: Represents the last lookup performed on the packet in forwarding path. Abstract Data Type: unsigned8 Data Type Semantics: identifier

Name: underlyingIngressInterface ElementID: TBD Description: The underlying interface index of the interface from where packet of a given flow are received in ingress. For example, child interface of an aggregate ethernet interface. Abstract Data Type: unsigned32 Data Type Semantics: identifier

6.2. Forwarding Exception Codes

This document requests addition of a new registry for Forwarding Exception Codes.

<table>
<thead>
<tr>
<th>Forwarding Exception Code</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FIREWALL_DISCARD</td>
</tr>
<tr>
<td>2</td>
<td>TTL_EXPIRY</td>
</tr>
<tr>
<td>3</td>
<td>DISCARD_ROUTE</td>
</tr>
<tr>
<td>4</td>
<td>BAD_IPV4_CHECKSUM</td>
</tr>
<tr>
<td>5</td>
<td>REJECT_ROUTE</td>
</tr>
<tr>
<td>6</td>
<td>BAD_IPV4_HEADER (Version incorrect or IHL &lt; 5)</td>
</tr>
<tr>
<td>7</td>
<td>BAD_IPV4_HEADER (Version incorrect)</td>
</tr>
<tr>
<td>8</td>
<td>BAD_IPV4_HEADER_LENGTH (V4 frame is too short)</td>
</tr>
<tr>
<td>9</td>
<td>BAD_IPV6_HEADER_LENGTH</td>
</tr>
<tr>
<td>10</td>
<td>BAD_IPV6_OPTIONS_PACKET (too many option headers)</td>
</tr>
</tbody>
</table>

Figure 7: Table 3: Exception Codes

All assignments in this registry are to be performed via Expert Review.
7. Security Considerations

Security Considerations listed in detail for IPFIX in [RFC7011] apply to this document as well. As described in [RFC7011], the IPFIX messages exchanged between network device and collector MUST be protected to provide confidentiality, integrity, and authenticity. Without those characteristics, the messages are subject to various kinds of attacks. These attacks are described in great detail in [RFC7011].

8. Contributors

Manikandan Musuvathi Poornachary
Juniper Networks, Inc.
Electra Exora Business Park~Marathahalli~Sarjapur Outer Ring Road,
Bangalore, KA - 560103
India
Email: mpoornachary@juniper.net

Vishnu Pavan Beeram
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089
USA
Email: vbeeram@juniper.net

Raveendra Torvi
Juniper Networks, Inc.
10 Technology Park Dr
Westford, MA 01886
USA
Email: rtorvi@juniper.net

9. References

9.1. Normative References


9.2. Informative References


Authors' Addresses

Venkata Naga Chaitanya Munukutla
Juniper Networks, Inc.
10 Technology Park Dr
Westford, MA 01886
United States of America
Email: vmunuku@juniper.net

Shivam Vaid
Juniper Networks, Inc.
Electra, Exora Business Park- Marathahalli-Sarjapur Outer Ring Road
Bangalore 560103
Karnataka
India
Email: shivamv@juniper.net

Aditya Mahale
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America

Email: amahale@google.com

Devang Patel  
Google, Inc.  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
United States of America  

Email: pateldevang@google.com
A YANG Model for Network and VPN Service Performance Monitoring
draft-www-opsawg-yang-vpn-service-pm-03

Abstract

The data model defined in RFC8345 introduces vertical layering relationships between networks that can be augmented to cover network/service topologies. This document defines a YANG model for both Network Performance Monitoring and VPN Service Performance Monitoring that can be used to monitor and manage network performance on the topology at higher layer or the service topology between VPN sites.

This document does not define metrics for network performance or mechanisms for measuring network performance. The YANG model defined in this document is designed as an augmentation to the network topology YANG model defined in RFC 8345 and draws on relevant YANG types defined in RFC 6991, RFC 8299, RFC 8345, and RFC 8532.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 25, 2021.

Copyright Notice

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

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

Table of Contents

1. Introduction ......................................................... 3
2. Terminology ............................................................ 3
3. Network and VPN Service Performance Monitoring Model Usage ........ 3
   3.1. Retrieval via Pub/Sub Mechanism ................................. 4
   3.2. On demand Retrieval via RPC Model ............................... 5
4. Description of the Data Model ........................................ 5
   4.1. Layering Relationship Between Multiple Layers of Topology .... 5
   4.2. Network Level .................................................... 6
   4.3. Node Level ........................................................ 7
   4.4. Link and Termination Point Level ................................. 8
5. Example of I2RS Pub/Sub Retrieval .................................. 11
6. Example of RPC-based Retrieval ..................................... 12
7. Network and VPN Service Assurance YANG Module .................... 14
8. Security Considerations .............................................. 26
9. IANA Considerations .................................................. 26
10. Acknowledgements .................................................... 27
11. Contributors .......................................................... 27
12. References ............................................................ 27
   12.1. Normative References .......................................... 27
   12.2. Informative References ....................................... 29
Authors’ Addresses ....................................................... 30
1. Introduction

[RFC4176] provides a framework for L3VPN operations and management and specifies that performance management is required after service configuration. This document defines a YANG Model for both network performance monitoring and VPN service performance monitoring that can be used to monitor and manage network performance on the topology level or the service topology between VPN sites.

This document does not introduce new metrics for network performance or mechanisms for measuring network performance, but uses the existing mechanisms and statistics to show the performance monitoring statistics at the network and service layers. The YANG model defined in this document is designed as an augmentation to the network topology YANG model defined in [RFC8345] and draws on relevant YANG types defined in [RFC6991], [RFC8299], [RFC8345], and [RFC8532].

2. Terminology

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

Tree diagrams used in this document follow the notation defined in [RFC8340].

3. Network and VPN Service Performance Monitoring Model Usage

Models are key for automatic management operations. According to [I-D.ietf-opsawg-model-automation-framework], together with service and network models, performance measurement telemetry model can monitor network performance to meet specific service SLA requirements. The model defined in this document is to derive VPN or network level performance data based on lower-level data collected via monitoring counters in the devices.
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.

```
+----------------------+          +-----------------------+
|ietf-network          |          |Network and VPN Service|
|ietf-network-topology |<--------|Performance Monitoring |
|                      |        |Model                  |
|+---------------------+        +----------------------+

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:
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.
o "vpn-summary-statistics": IPv4 statistics, and IPv6 statistics have been specified separately. And MAC statistics could be extended for L2VPN.

augment /nw:networks/nw:network/nw:node:
  +--rw node-attributes
    |  +--rw node-type? identityref
    |  +--rw site-id? string
    |  +--rw site-role? identityref
  +--rw vpn-summary-statistics
    +--rw ipv4
    |  +--rw total-routes? uint32
    |  +--rw total-active-routes? uint32
    +--rw ipv6
    |  +--rw total-routes? uint32
    |  +--rw total-active-routes? uint32

Figure 5: Node Level View of the hierarchies

4.4. Link and Termination Point Level

The link nodes are classified into two types: one is topology link defined in [RFC8345], and the other is abstract link of a VPN between PEs.

The performance data of the link is a collection of counters that report the performance status. The data for the topology link can be based on BGP-LS [RFC8571]. The statistics of the VPN abstract links can be collected based on VPN OAM mechanisms, e.g. TWAMP etc. Alternatively, the data can base on the underlay technology OAM mechanism, for example, GRE tunnel OAM.
augment /nw:networks/nw:network/nt:link:
  +--rw link-type?  identityref
augment /nw:networks/nw:network/nt:link:
  +--rw low-percentile?  percentile
  +--rw middle-percentile?  percentile
  +--rw high-percentile?  percentile
  +--rw reference-time?  yang:date-and-time
  +--rw measurement-interval?  uint32
  +--ro link-telemetry-attributes
  |  +--ro loss-statistics
  |  |  +--ro packet-loss-count?  yang:counter32
  |  |  +--ro packet-reorder-count?  yang:counter32
  |  |  +--ro packets-out-of-seq-count?  yang:counter32
  |  |  +--ro packets-dup-count?  yang:counter32
  |  |  +--ro loss-ratio?  percentage
  |  +--ro delay-statistics
  |  |  +--ro direction?  identityref
  |  |  +--ro unit-value?  identityref
  |  |  +--ro min-delay-value?  yang:gauge64
  |  |  +--ro max-delay-value?  yang:gauge64
  |  |  +--ro low-delay-percentile?  yang:gauge64
  |  |  +--ro middle-delay-percentile?  yang:gauge64
  |  |  +--ro high-delay-percentile?  yang:gauge64
  +--ro jitter-statistics
  |  +--ro unit-value?  identityref
  |  +--ro min-jitter-value?  yang:gauge32
  |  +--ro max-jitter-value?  yang:gauge32
  |  +--ro low-jitter-percentile?  yang:gauge32
  |  +--ro middle-jitter-percentile?  yang:gauge32
  |  +--ro high-jitter-percentile?  yang:gauge32
augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--ro tp-telemetry-attributes
  |  +--ro inbound-octets?  yang:counter64
  |  +--ro inbound-unicast?  yang:counter64
  |  +--ro inbound-nunicast?  yang:counter64
  |  +--ro inbound-discards?  yang:counter32
  |  +--ro inbound-errors?  yang:counter32
  |  +--ro inbound-unknown-protocol?  yang:counter32
  |  +--ro outbound-octets?  yang:counter64
  |  +--ro outbound-unicast?  yang:counter64
  |  +--ro outbound-nunicast?  yang:counter64
  |  +--ro outbound-discards?  yang:counter32
  |  +--ro outbound-errors?  yang:counter32
  |  +--ro outbound-qlen?  uint32

Figure 6: Link and Termination point Level View of the hierarchies
For the nodes of the link in the figure, this module defines the following minimal set of link level performance attributes:

- **"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.

```xml
<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>
        </network>
        <node
        <node-attributes
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
          <node-type>pe</node-type>
        </node-attributes>
      </networks>
      <termination-point
        <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>
    </stream-subtree-filter>
  </establish-subscription>
</rpc>
```
<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>
  <link-telemetry-attributes
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
    <loss-statistics>
      <packet-loss-count>100</packet-loss-count>
    </loss-statistics>
  </link-telemetry-attributes>
</link>
</network>
</networks>
</stream-subtree-filter>
<period
xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
  500
</period>
</establish-subscription>
</rpc>

6. Example of RPC-based Retrieval

This example shows the way for the client to use RPC model to fetch performance data on demand, e.g., the client requests "packet-loss-count" between PE1 in site 1 and PE2 in site 2 belonging to the same VPN1.

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="1">
  <report
xmlns="urn:ietf:params:xml:ns:yang:example-service-pm-report">
    <networks
 xmlns="urn:ietf:params:xml:ns:yang:ietf-network-topo">
      <network>
<network-id>vpn1</network-id>
<node>
  <node-id>A</node-id>
  <node-attributes
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
    <node-type>pe</node-type>
  </node-attributes>
  <termination-point
    <tp-id>1-0-1</tp-id>
    <tp-telemetry-attributes
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
      <inbound-octets>100</inbound-octets>
      <outbound-octets>150</outbound-octets>
    </tp-telemetry-attributes>
  </termination-point>
</node>

<node>
  <node-id>B</node-id>
  <node-attributes
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm">
    <node-type>pe</node-type>
  </node-attributes>
  <termination-point
    <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>
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>"
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
    "Jitter statistics for the link";
    leaf-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:gauge64;
      description
        "Minimum jitter value observed.";
    }
    leaf max-jitter-value {
      type yang:gauge64;
      description
        "Maximum jitter value observed.";
    }
    leaf low-jitter-percentile {
      type yang:gauge64;
      description
        "Low percentile of the jitter observed with specific measurement method.";
    }
    leaf middle-jitter-percentile {
      type yang:gauge64;
      description
        "Middle percentile of the jitter observed with specific measurement method.";
    }
    leaf high-jitter-percentile {
      type yang:gauge64;
      description
        "High percentile of the jitter observed with specific measurement method.";
    }
  }
}

'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 topology 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.";
    }
}
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:


9. IANA Considerations

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

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.
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


12.2. Informative References


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
Abstract

This document describes how to find and authenticate geofeed data.

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 15, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

Providers of Internet content and other services may wish to customize those services based on the geographic location of the user of the service. This is often done using the source IP address used to contact the service. Also, infrastructure and other services might wish to publish the locale of their services. [RFC8805] defines geofeed, a syntax to associate geographic locales with IP addresses. But it does not specify how to find the relevant geofeed data given an IP address.

This document specifies how to augment the Routing Policy Specification Language (RPSL) [RFC2622] inetnum: class [INETNUM] to refer to geofeed data, and how to prudently use them. In all places inetnum: is used, inet6num: should also be assumed [INET6NUM].

An optional, but utterly awesome, means for authenticating geofeed data is also defined.

1.1. 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. Geofeed Files

Geofeed files are described in [RFC8805]. They provide a facility for an IP address resource 'owner' to associate those IP addresses to geographic locale(s).

Content providers and other parties who wish to locate an IP address to a geographic locale need to find the relevant geofeed data. In Section 3 this document specifies how to find the relevant geofeed file given an IP address.

Geofeed data for large providers with significant horizontal scale and high granularity can be quite large. The size of a file can be even larger if an unsigned geofeed file combines data for many prefixes, as may be likely if the location data are maintained by a different department than address management, dual IPv4/IPv6 spaces are represented, etc.

This document also suggests optional data for geofeed files to provide stronger authenticity to the data.

3. inetnum: Class

RPSL, [RFC2622], as used by the Regional Internet Registries (RIRs), has been augmented with the inetnum: [INETNUM] and the inet6num: [INET6NUM] classes; each of which describes an IP address range and its attributes.

Ideally, RPSL would be augmented to define a new RPSL geofeed: attribute in the inetnum: class. Until such time, this document defines the syntax of a Geofeed remarks: attribute which contains an HTTPS URL of a geofeed file. The format MUST be as in this example, "remarks: Geofeed " followed by a URL which will vary.

```
inetnum: 192.0.2.0/24 # example
   remarks: Geofeed https://example.com/geofeed.csv
```

Any particular inetnum: object MAY have, at most, one geofeed reference, whether a remark: or a proper geofeed: attribute when one is defined.

```
inetnum: objects form a hierarchy, see [INETNUM] Section 4.2.4.1, Hierarchy of INETNUM Objects. Geofeed references SHOULD be at the lowest applicable inetnum: object. When fetching, the most specific inetnum: object with a geofeed reference MUST be used.
```

When geofeed references are provided by multiple inetnum: objects which have identical address ranges, then the geofeed reference on
the inetnum: with the most recent last-modified: attribute SHOULD be preferred.

It is significant that geofeed data may have finer granularity than the inetnum: which refers to them.

Currently, the registry data published by ARIN is not the same RPSL as the other registries; therefore, when fetching from ARIN via FTP [RFC0959], whois [RFC3912], RDAP [RFC7482], or whatever, the "NetRange" attribute/key must be treated as "inetnum" and the "Comment" attribute must be treated as "remarks".

4. Authenticating Geofeed Data

The question arises of whether a particular geofeed data set is valid, i.e. authorized by the 'owner' of the IP address space and is authoritative in some sense. The inetnum: which points to the geofeed file provides some assurance. Unfortunately the RPSL in many repositories is weakly authenticated at best. An approach where RPSL was signed a la [RFC7909] would be good, except it would have to be deployed by all RPSL registries.

An optional authenticator MAY be appended to a geofeed file. It would be essentially a digest of the main body of the file signed by the private key of the relevant RPKI certificate for the covering address range. One needs a format that bundles the relevant RPKI certificate with the signature and the digest of the geofeed text.

Borrowing detached signatures from [RFC5485], after text file canonicalization (Sec 2.2), the Cryptographic Message Syntax (CMS) [RFC3852] would be used to create a detached DER encoded signature which is then BASE64 encoded and line wrapped to 72 or fewer characters.

Both the address ranges of the signing certificate and of the inetnum: MUST cover all prefixes in the geofeed file; and the address range of the signing certificate must cover that of the inetnum:

An address range A 'covers' address range B if the range of B is identical to or a subset of A. 'Address range' is used here because inetnum: objects and RPKI certificates need not align on CIDR prefix boundaries, while those of geofeed lines must.

As the signer would need to specify the covered RPKI resources relevant to the signature, the RPKI certificate covering the inetnum: object’s address range would be included in the [RFC3852] CMS SignedData certificates field.
Identifying the private key associated with the certificate, and getting the department with the HSM to sign the CMS blob is left as an exercise for the implementor. On the other hand, verifying the signature requires no complexity; the certificate, which can be validated in the public RPKI, has the needed public key.

Until [RFC8805] is updated to formally define such an appendix, it MUST be 'hidden' as a series of "#" comments at the end of the geofeed file. This is a cryptographically incorrect, albeit simple example. A correct and full example is in Appendix A.

```
# RPKI Signature: 192.0.2.0/24
# MIIGlwYJKoZIhvcNAQcCoIIgiDCCBoQCAQMxDTALBgglhgkgBZQMEAgEwQYLKoZ
# IjvcNAQkQAS+ggg5xMIIErTCCA5WwIBAgIUIJ605QIFX8rW5m4Zw3Wyw7hZu
...  
# imwYkXpiMxw44EzqDj136MsRsDldg0jJBbcGibwuyAwGeR46k5raZCGvxG+4xa
# OPBDxTf1YWAhaBjRBKAgqA7xh5xHfm58jUXsZJ1eq1S7G6Kk=
# End Signature: 192.0.2.0/24
```

5. Operational Considerations

To create the needed inetnum: objects, an operator wishing to register the location of their geofeed file needs to coordinate with their RIR/NIR and/or any provider LIR which has assigned prefixes to them. RIRs/NIRs provide means for assignees to create and maintain inetnum: objects. They also provide means of [sub-]assigning IP address resources and allowing the assignee to create whois data, including inetnum: objects, and thereby referring to geofeed files.

The geofeed files SHOULd be published over and fetched using https [RFC2818].

When using data from a geofeed file, one MUST ignore data outside of the referring inetnum: object's attribute address range.

Iff the geofeed file is not signed per Section 4, then multiple inetnum: objects MAY refer to the same geofeed file, and the consumer MUST use only geofeed lines where the prefix is covered by the address range of the inetnum: object they have followed.

To minimize the load on RIR whois [RFC3912] services, use of the RIR's FTP [RFC0959] services SHOULD be the preferred access. This also provides bulk access instead of fetching with a tweezers.

Currently, geolocation providers have bulk whois data access at all the RIRs. An anonymized version of such data is openly available for all RIRs except ARIN, which requires an authorization. However, for users without such authorization the same result can be achieved with...
extra RDAP effort. There is open source code to pass over such data across all RIRs, collect all geofeed references, and process them [geofeed-finder].

An entity fetching geofeed data using these mechanisms MUST NOT do frequent real-time look-ups to prevent load on RPSL and geofeed servers. [RFC8805] Section 3.4 suggests use of the [RFC7234] HTTP Expires Caching Header to signal when geofeed data should be refetched. As the data change very infrequently, in the absence of such an HTTP Header signal, collectors MUST NOT fetch more frequently than weekly. It would be wise not to fetch at magic times such as midnight UTC, the first of the month, etc., because too many others are likely to do the same.

6. Security Considerations

It would be generally prudent for a consumer of geofeed data to also use other sources to cross-validate the data. All of the Security Considerations of [RFC8805] apply here as well.

As mentioned in Section 4, many RPSL repositories have weak if any authentication. This would allow spoofing of inetnum: objects pointing to malicious geofeed files. Section 4 suggests an unfortunately complex method for stronger authentication based on the RPKI.

If an inetnum: for a wide prefix (e.g. a /16) points to an RPKI-signed geofeed file, a customer or attacker could publish a unsigned equal or narrower (e.g. a /24) inetnum: in a whois registry which has weak authorization.

The RPSL providers have had to throttle fetching from their servers due to too-frequent queries. Usually they throttle by the querying IP address or block. Similar defenses will likely need to be deployed by geofeed file servers.

7. IANA Considerations

IANA is asked to register object identifiers for one content type in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry as follows:

<table>
<thead>
<tr>
<th>Description</th>
<th>OID</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>id-ct-geofeedCSVwithCRLF</td>
<td>1.2.840.113549.1.9.16.1.47</td>
<td>[RFC-TBD]</td>
</tr>
</tbody>
</table>
8. Acknowledgements

Thanks to Rob Austein for CMS and detached signature clue. George Michaelson for the first, and a substantial, external review. Erik Kline who was too shy to agree to co-authorship. Additionally, we express our gratitude to early implementors, including Menno Schepers, Flavio Luciani, Eric Dugas, and Kevin Pack. Also to geolocation providers that are consuming geofeeds with this described solution, Jonathan Kosgei (ipdata.co), and Ben Dowling (ipinfo.io).

9. References

9.1. Normative References


9.2. Informative References


Appendix A. Example

This appendix provides an example, including a trust anchor, a CA certificate subordinate to the trust anchor, an end-entity
certificate subordinate to the CA for signing the geofeed, and a
detached signature.

The trust anchor is represented by a self-signed certificate. As
usual in the RPKI, the trust anchor has authority over all IPv4
address blocks, all IPv6 address blocks, and all AS numbers.

-----BEGIN CERTIFICATE-----
MIIEpjCCAyagAwIBAgIUPsUFJ4e/7pKZ6E14aBdkbYzmslgwDQYJKoZIhvcNAQEL
BQAwFETETMBEGA1UEAxMKZXhhbXBsZS10YTAeFw0yMDc5MDMxMTIwNzRaFw0xMDA5
MDEwODUwNTRaMBUxExARBgNVBAMTCmV4YW1wbGwgdGhlIENyZWF0aW9uc2Vx
AQUA4IIBAwgEKACeBQAGSCEgEAMBIGCjAGCSqGSIb3DQEJ
-----END CERTIFICATE-----

The CA certificate is issued by the trust anchor. This certificate
grants authority over one IPv4 address block (192.0.2.0/24) and two
AS numbers (64496 and 64497).

-----BEGIN CERTIFICATE-----
THE CA certificate is issued by the trust anchor. This certificate
grants authority over one IPv4 address block (192.0.2.0/24) and two
AS numbers (64496 and 64497).
The end-entity certificate is issued by the CA. This certificate grants signature authority for one IPv4 address block (192.0.2.0/24). Signature authority for AS numbers is not needed for geofeed data signatures, so no AS numbers are included in the certificate.
The end-entity certificate is displayed below in detail. For brevity, the other two certificates are not.

```
0 1197: SEQUENCE {
  4 917:  SEQUENCE {
    8  3:   [0] {
      10  1:  INTEGER 2
        :  }
    13  20:  INTEGER 27AD394083D7F2B599B8670C775B296EE166E3
    35  13:  SEQUENCE {
      37  9:   OBJECT IDENTIFIER
        :     sha256WithRSAEncryption (1 2 840 113549 1 1 11)
      48  0:   NULL
        :     }
    50  51:  SEQUENCE {
      52  49:  SET {
        :   }
      56  3:   OBJECT IDENTIFIER commonName (2 5 4 3)
    61  40:  PrintableString
```

486 435:  SEQUENCE {
490 29:   SEQUENCE {
492 3:    OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
497 22:    OCTET STRING, encapsulates {
499 20:     OCTET STRING
501 :      91 46 52 A3 BD 51 C1 44 26 01 98 88 9F 5C 45 AB
503 :      F0 53 A1 87
505 :   }
521 31:   SEQUENCE {
523 3:    OBJECT IDENTIFIER authorityKeyIdentifier (2 5 29 35)
528 24:    OCTET STRING, encapsulates {
530 22:     SEQUENCE {
532 20:      [0]
534 :       3A CE 2C EF 4F B2 1B 7D 11 E3 E1 84 EF C1 E2 97
536 :       B3 77 86 42
538 :     }
540 :   }
554 12:   SEQUENCE {
556 3:    OBJECT IDENTIFIER basicConstraints (2 5 29 19)
561 1:     BOOLEAN TRUE
564 2:     OCTET STRING, encapsulates {
566 0:      SEQUENCE {}
568 :     }
568 14:   SEQUENCE {
570 3:    OBJECT IDENTIFIER keyUsage (2 5 29 15)
575 1:     BOOLEAN TRUE
578 4:     OCTET STRING, encapsulates {
580 2:      BIT STRING 7 unused bits
582 :       '1'B (bit 0)
584 :     }
584 24:   SEQUENCE {
586 3:    OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
591 1:     BOOLEAN TRUE
594 14:    OCTET STRING, encapsulates {
596 12:     SEQUENCE {
598 10:      SEQUENCE {
600 8:       OBJECT IDENTIFIER
602 :         resourceCertificatePolicy (1 3 6 1 5 5 7 14 2)
604 :       }
606 :     }
610 97:   SEQUENCE {
612 3:    OBJECT IDENTIFIER cRLDistributionPoints (2 5 29 31)
OCTET STRING, encapsulates {
  SEQUENCE {
    [0] {
      [0] {
        [6]
        : 'rsync://rpki.example.net/repository/3ACE2CEF4F'
        : 'B21B7D11E3E184EFC1E297B3778642.crl'
        : }
        : }
        : }
    }
  }
}

OBJECT IDENTIFIER authorityInfoAccess
: (1 3 6 1 5 5 7 1 1)

OCTET STRING, encapsulates {
  SEQUENCE {
    OBJECT IDENTIFIER caIssuers (1 3 6 1 5 5 7 48 2)
    [6]
    : 'rsync://rpki.example.net/repository/3ACE2CEF4F'
    : 'B21B7D11E3E184EFC1E297B3778642.cer'
    : }
    : }
    : }
}

OBJECT IDENTIFIER ipAddrBlocks (1 3 6 1 5 5 7 1 7)

BOOLEAN TRUE

OCTET STRING, encapsulates {
  SEQUENCE {
    OCTET STRING 00 01
    NULL
    : }
    OCTET STRING 00 02
    NULL
    : }
    OCTET STRING, encapsulates {
To allow reproduction of the signature results, the end-entity private key is provided. For brevity, the other two private keys are not.
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAsnE0Kzm/6gdlt4tyovD4QPwxFsootk4BqaYAsDvZbCES0mW
/5Pmkollj/ZEn5XUELtwlcK+toU0GqiKMATdAS9HctF+ZNYpiXYuanTN57yrMDP
Ap6EddbfwKUCbk7mZq+caYV0xppps71VS4lt1dbq2g7V1paHsprnYellifihg48D1
ztOy1wXowazhTV4Whs3tPMuAz36/0v7VyTgZu0M0Kbwzmy2Lrn6a2LuOZYhraqj/
eFH165En13d+gChs6kXQnHnxFvzeVBNRTB526BKIkraC6CqAbdCJDhR1ngvxxIHm
gXV13uOvXXQva0H7ecOoOnJsRvmmA3SBaD+M6wIDAQABaoIBAQCyB0FeMuKm9bRo
18AkjFG5PcOz153srIz5bvuUGi192TBelez72znL61ym260j+5th+1CHGO/dq1hXio
p150CSyc9TFb1b/ECOsucuqKfj28CD3GVsHozXKJeMM+/o5YZXQrORj6unwT0z
ol/JE5pIGUCIgsXXt6zt9s5BP31UAuVqHsV6+vEVLxQ3wj/1vIL80/CN036EV0GJ
mpkwmygPjfETC9wbWo0yn3jxJb36+M/QjJUP28oNIVn/KIoPZRnxchEbucCJ651
IsaFsgti1Thm4WZtvC/IDq+6/dmcumTjIRcyW7fhFjpl1LVpe9c/XompWEQvF
3A3WUt5AoGBANs4764yHxo4mctLIE7G71/tf9bP4KKUyW4RbyEcoucMC4yhtm
MFCfOFLOQet71OWCkjp2L/7EKUe9yx7G5KmxAY6jOjvcRkvGsl61WOfsQ8p126M
Y9hmGzMoMjtsdhAImm0Wzjvm4WqfMggqPe+PnjjSVkgTt+7BxpIuGBAAoGBANcg
26FF5cDlpixOd3zAlYsOqgwCaw3Pli7vUZRPa/zBELEtEyeobfkAENWn071
nE+lAZxwm+29PDTOncFE91teyjznQaLO5kkAdJjFvVu3icLQO39frnJbKensm
FGSli+3KxQhCNIJfgWzq4bEo0oAMjdGbyX2yxIYQFAoGBAM6tuDj36KDU+hIS6wu6
O2TPSzfHf/zPo3pCWQ78/Qdb+Zwd4fEiqoBA7F4NPVqg9Y/8HUTX9r/vege7hPoo
Ok7NpifsMktTHk5Xf262Zn90LPGoKba40a1kXordwEUS2YRoAtAE9f6/Rog6PYyz
veLpqscRbu0XQhlkN+z7bgb5a0AGBAKsDsbDEB/dbqbyaAypmwhH2sdRSKkgp7N1wc
DNNm9Wx1lJ62w1+M871G8a8REEU1AIVqHWVlr/ROBQ6NTIjUc5/qFet2XUkgf
tMkVv61tuyZK3sTnzmnHoHzDUpWJeWnCEUB+ZYVdmo522Gw2A75rdzLL2r9Dc
PvDxVubRARoAdqesWloUxuzXZx8sraKQsTYxNa0rWQz0eUL55vVe8nK257UdQZ
E3nq2j5XPTWli+aNGFeJG0NcCQv060/sfZUhu52sq9mWynHNltB5aP8x+pV
iFc2OLJyEcN6P+YQKFUt1rAI1MGM5RDNvU1L2xfCYyb7FzV6Y=
-----END RSA PRIVATE KEY-----

Signing of "192.0.2.0/24,US,WA,Seattle," (terminated by CR and LF),
yields the following detached CMS signature.
# Authors’ Addresses

Massimo Candela  
NTT  
Siriusdreef 70-72  
Hoofddorp  
2312 WT  
Netherlands

Email: massimo@ntt.net
Randy Bush
IIJ & Arrcus
5147 Crystal Springs
Bainbridge Island, Washington  98110
United States of America

Email: randy@psg.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA  94043
US

Email: warren@kumari.net

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA  20170
USA

Email: housley@vigilsec.com