

opsawg
Internet-Draft
Intended status: Best Current Practice
Expires: December 26, 2021

F. Brockners
Cisco
S. Bhandari, Ed.
Thoughtspot
D. Bernier
Bell Canada
T. Mizrahi, Ed.
Huawei
June 24, 2021

In-situ OAM Deployment
draft-brockners-opsawg-ioam-deployment-03

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document provides a framework for IOAM deployment and provides best current practices.

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 December 26, 2021.

Copyright Notice

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

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

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. IOAM Deployment: Domains And Nodes	4
4. Types of IOAM	5
4.1. Per-hop Tracing IOAM	6
4.2. Proof of Transit IOAM	8
4.3. Edge to Edge IOAM	8
4.4. Direct Export IOAM	8
5. IOAM Encapsulations	8
5.1. IPv6	9
5.2. NSH	9
5.3. GRE	9
5.4. Geneve	10
5.5. Segment Routing	10
5.6. Segment Routing for IPv6	10
5.7. VXLAN-GPE	10
6. IOAM Data Export	10
7. IOAM Deployment Considerations	11
7.1. IOAM Namespaces	12
7.2. IOAM Layering	13
7.3. IOAM Trace Option Types	14
7.4. Traffic-sets That IOAM Is Applied To	16
7.5. IOAM Loopback Mode	16
7.6. IOAM Active Mode	16
7.7. Brown Field Deployments: IOAM Unaware Nodes	17
8. IOAM Manageability	17
9. IANA Considerations	18
10. Security Considerations	18
11. Acknowledgements	19
12. References	19
12.1. Normative References	19
12.2. Informative References	20
Authors' Addresses	23

1. Introduction

"In-situ" Operations, Administration, and Maintenance (IOAM) records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than is being

sent within packets specifically dedicated to OAM. IOAM is to complement mechanisms such as Ping, Traceroute, or other active probing mechanisms. In terms of "active" or "passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. "In-situ" mechanisms do not require extra packets to be sent. IOAM adds information to the already available data packets and therefore cannot be considered passive. In terms of the classification given in [RFC7799] IOAM could be portrayed as Hybrid Type 1. IOAM mechanisms can be leveraged where mechanisms using e.g. ICMP do not apply or do not offer the desired results, such as proving that a certain traffic flow takes a pre-defined path, SLA verification for the live data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios in which probe traffic is potentially handled differently from regular data traffic by the network devices.

2. Conventions

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.

Abbreviations used in this document:

E2E	Edge to Edge
Geneve:	Generic Network Virtualization Encapsulation [I-D.ietf-nvo3-geneve]
GRE	Generic Routing Encapsulation
IOAM:	In-situ Operations, Administration, and Maintenance
MTU:	Maximum Transmit Unit
NSH:	Network Service Header [RFC8300]
OAM:	Operations, Administration, and Maintenance
POT:	Proof of Transit
SFC:	Service Function Chain
SID:	Segment Identifier
SR:	Segment Routing

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension [I-D.ietf-nvo3-vxlan-gpe]

3. IOAM Deployment: Domains And Nodes

IOAM is a network domain specific feature, with "network domain" being a set of network devices or entities within a single administration. IOAM is not targeted for a deployment on the global Internet. The part of the network which employs IOAM is referred to as the "IOAM-Domain". For example, an IOAM-domain can include an enterprise campus using physical connections between devices or an overlay network using virtual connections / tunnels for connectivity between said devices. An IOAM-domain is defined by its perimeter or edge. The operator of an IOAM-domain is expected to put provisions in place to ensure that packets which contain IOAM data fields do not leak beyond the edge of an IOAM domain, e.g. using for example packet filtering methods. The operator should consider the potential operational impact of IOAM to mechanisms such as ECMP processing (e.g. load-balancing schemes based on packet length could be impacted by the increased packet size due to IOAM), path MTU (i.e. ensure that the MTU of all links within a domain is sufficiently large to support the increased packet size due to IOAM) and ICMP message handling.

An IOAM-Domain consists of "IOAM encapsulating nodes", "IOAM decapsulating nodes" and "IOAM transit nodes". The role of a node (i.e. encapsulating, transit, decapsulating) is defined within an IOAM-Namespace (see below). A node can have different roles in different IOAM-Namespaces.

An "IOAM encapsulating node" incorporates one or more IOAM-Option-Types into packets that IOAM is enabled for. If IOAM is enabled for a selected subset of the traffic, the IOAM encapsulating node is responsible for applying the IOAM functionality to the selected subset.

An "IOAM transit node" updates one or more of the IOAM-Data-Fields. If both the Pre-allocated and the Incremental Trace Option-Types are present in the packet, each IOAM transit node will update at most one of these Option-Types. A transit node does not add new IOAM-Option-Types to a packet, and does not change the IOAM-Data-Fields of an IOAM Edge-to-Edge Option-Type.

An "IOAM decapsulating node" removes IOAM-Option-Type(s) from packets.

The role of an IOAM-encapsulating, IOAM-transit or IOAM-decapsulating node is always performed within a specific IOAM-Namespace. This means that an IOAM node which is e.g. an IOAM-decapsulating node for

IOAM-Namespace "A" but not for IOAM-Namespace "B" will only remove the IOAM-Option-Types for IOAM-Namespace "A" from the packet. An IOAM decapsulating node situated at the edge of an IOAM domain removes all IOAM-Option-Types and associated encapsulation headers for all IOAM-Namespaces from the packet.

IOAM-Namespaces allow for a namespace-specific definition and interpretation of IOAM-Data-Fields. An interface-id could for example point to a physical interface (e.g., to understand which physical interface of an aggregated link is used when receiving or transmitting a packet) whereas in another case it could refer to a logical interface (e.g., in case of tunnels). Please refer to Section 7.1 for a discussion of IOAM-Namespaces.

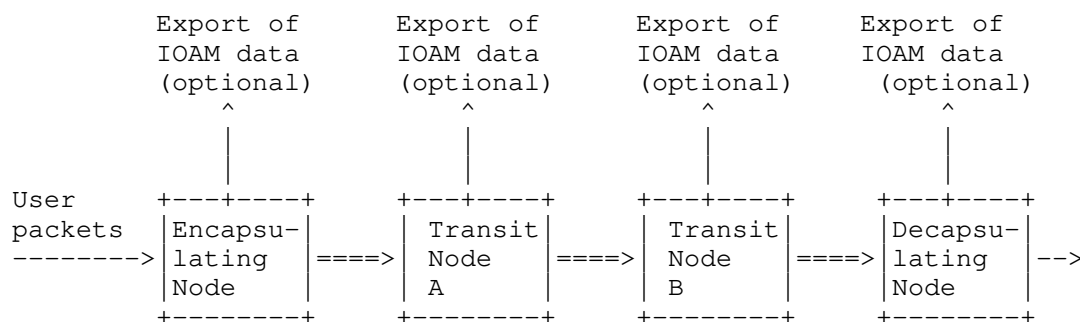


Figure 1: Roles of IOAM nodes

IOAM nodes which add or remove the IOAM-Data-Fields can also update the IOAM-Data-Fields at the same time. Or in other words, IOAM encapsulating or decapsulating nodes can also serve as IOAM transit nodes at the same time. Note that not every node in an IOAM domain needs to be an IOAM transit node. For example, a deployment might require that packets traverse a set of firewalls which support IOAM. In that case, only the set of firewall nodes would be IOAM transit nodes rather than all nodes.

4. Types of IOAM

IOAM supports different modes of operation, which are differentiated by the type of IOAM data fields being carried in the packet, the data being collected, the type of nodes which collect or update data as well as whether and how nodes export IOAM information.

- o Per-hop tracing: OAM information about each IOAM node a packet traverses is collected and stored within the user data packet as

the packet progresses through the IOAM domain. Potential uses of IOAM per-hop tracing include:

- * Optimization: Understand the different paths different packets traverse between a source and a sink in a network that uses load balancing such as Equal Cost Load Balancing (ECMP). This information could be used to tune the algorithm for ECMP for optimized network resource usage.
- * Operations/Troubleshooting: Understand which path a particular packet or set of packets take as well as what amount of jitter and delay different nodes in the path contribute to the overall end-to-end delay and jitter.
- o Proof-of-transit: Information that a verifier node can use to verify whether a packet has traversed all nodes that is supposed to traverse is stored within the user data packet. Proof-of-transit could for example be used to verify that a packet indeed passes through all services of a service function chain (e.g. verify whether a packet indeed traversed the set of firewalls that it is expected to traverse), or whether a packet indeed took the expected path.
- o Edge-to-edge: OAM information which is specific to the edges of an IOAM domain is collected and stored within the user data packet. Edge-to-Edge OAM could be used to gather operational information about a particular network domain, such as the delay and jitter incurred by that network domain or the traffic matrix of the network domain.
- o Direct export: OAM information about each IOAM node a packet traverses is collected and immediately exported to a collector. Direct export could be used in situations where per-hop tracing information is desired, but gathering the information within the packet - as with per-hop tracing - is not feasible. Rather than automatically correlating the per-hop tracing information, as done with per-hop tracing, direct export requires a collector to correlate the information from the individual nodes. In addition, all nodes enabled for direct export need to be capable to export the IOAM information to the collector.

4.1. Per-hop Tracing IOAM

"IOAM tracing data" is expected to be collected at every IOAM transit node that a packet traverses to ensure visibility into the entire path a packet takes within an IOAM-Domain. I.e., in a typical deployment all nodes in an IOAM-Domain would participate in IOAM and thus be IOAM transit nodes, IOAM encapsulating or IOAM decapsulating

nodes. If not all nodes within a domain are IOAM capable, IOAM tracing information (i.e., node data, see below) will only be collected on those nodes which are IOAM capable. Nodes which are not IOAM capable will forward the packet without any changes to the IOAM-Data-Fields. The maximum number of hops and the minimum path MTU of the IOAM domain is assumed to be known.

IOAM offers two different trace Option-Types, the "incremental" Option-Type as well as the "pre-allocated" Option-Type. For a discussion which of the two option types is the most suitable for an implementation and/or deployment, see Section 7.3.

Every node data entry holds information for a particular IOAM transit node that is traversed by a packet. The IOAM decapsulating node removes the IOAM-Option-Type(s) and processes and/or exports the associated data. All IOAM-Data-Fields are defined in the context of an IOAM-Namespace.

IOAM tracing can collect the following types of information:

- o Identification of the IOAM node. An IOAM node identifier can match to a device identifier or a particular control point or subsystem within a device.
- o Identification of the interface that a packet was received on, i.e. ingress interface.
- o Identification of the interface that a packet was sent out on, i.e. egress interface.
- o Time of day when the packet was processed by the node as well as the transit delay. Different definitions of processing time are feasible and expected, though it is important that all devices of an in-situ OAM domain follow the same definition.
- o Generic data: Format-free information where syntax and semantic of the information is defined by the operator in a specific deployment. For a specific IOAM-Namespace, all IOAM nodes should interpret the generic data the same way. Examples for generic IOAM data include geo-location information (location of the node at the time the packet was processed), buffer queue fill level or cache fill level at the time the packet was processed, or even a battery charge level.
- o Information to detect whether IOAM trace data was added at every hop or whether certain hops in the domain weren't IOAM transit nodes.

- o Data that relates to how the packet traversed a node (transit delay, buffer occupancy in case the packet was buffered, queue depth in case the packet was queued)

The Option-Types of incremental tracing and pre-allocated tracing are defined in [I-D.ietf-ippm-ioam-data].

4.2. Proof of Transit IOAM

IOAM Proof of Transit Option-Type is to support path or service function chain [RFC7665] verification use cases. Proof-of-transit uses methods like nested hashing or nested encryption of the IOAM data or mechanisms such as Shamir's Secret Sharing Schema (SSSS).

The IOAM Proof of Transit Option-Type consist of a fixed size "IOAM proof of transit option header" and "IOAM proof of transit option data fields". For details see [I-D.ietf-ippm-ioam-data].

4.3. Edge to Edge IOAM

The IOAM Edge-to-Edge Option-Type is to carry data that is added by the IOAM encapsulating node and interpreted by IOAM decapsulating node. The IOAM transit nodes may process the data but must not modify it.

The IOAM Edge-to-Edge Option-Type consist of a fixed size "IOAM Edge-to-Edge Option-Type header" and "IOAM Edge-to-Edge Option-Type data fields". For details see [I-D.ietf-ippm-ioam-data].

4.4. Direct Export IOAM

Direct Export is an IOAM mode of operation within which IOAM data to be directly exported to a collector rather than being collected within the data packets. The IOAM Direct Export Option-Type consist of a fixed size "IOAM direct export option header". Direct Export for IOAM is defined in [I-D.ietf-ippm-ioam-direct-export].

5. IOAM Encapsulations

IOAM data fields and associated data types for in-situ OAM are defined in [I-D.ietf-ippm-ioam-data]. The in-situ OAM data field can be transported by a variety of transport protocols, including NSH, Segment Routing, Geneve, IPv6, etc.

5.1. IPv6

IOAM encapsulation for IPv6 is defined in [I-D.ietf-ippm-ioam-ipv6-options]. IOAM deployment considerations for IPv6 networks are found in [I-D.ioametal-ippm-6man-ioam-ipv6-deployment].

5.2. NSH

IOAM encapsulation for NSH is defined in [I-D.ietf-sfc-ioam-nsh].

5.3. GRE

IOAM encapsulation for GRE is outlined as part of the "EtherType Protocol Identification of In-situ OAM Data" in [I-D.weis-ippm-ioam-eth], though no example protocol header stacks are provided in the document. When used with GRE, the IOAM-Option-Types (the below diagram uses "IOAM" as shorthand for IOAM-Option-Types) are sequenced in behind the GRE header that follows the "outer" IP header. Figure 2 shows two example protocol header stacks that use GRE along with IOAM.

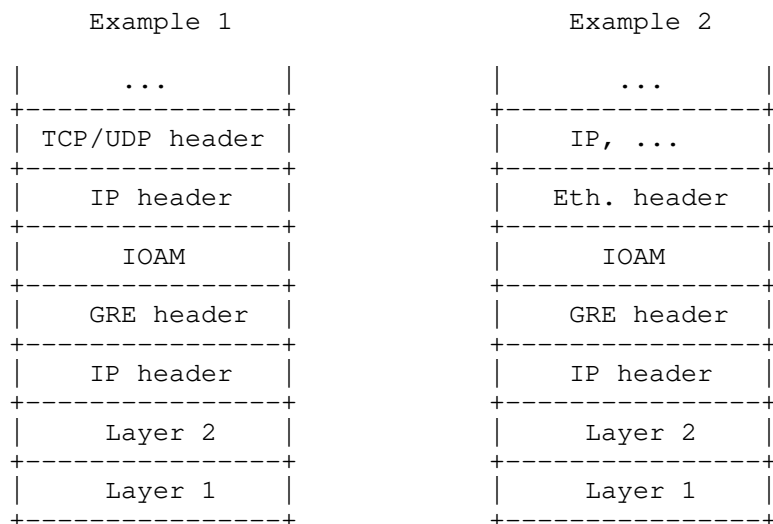


Figure 2: GRE with IOAM examples

5.4. Geneve

IOAM encapsulation for Geneve is defined in [I-D.brockners-ippm-ioam-geneve].

5.5. Segment Routing

IOAM encapsulation for Segment Routing is defined in [I-D.gandhi-spring-ioam-sr-mpls].

5.6. Segment Routing for IPv6

IOAM encapsulation for Segment Routing over IPv6 is defined in [I-D.ali-spring-ioam-srv6].

5.7. VXLAN-GPE

IOAM encapsulation for VXLAN-GPE is defined in [I-D.brockners-ippm-ioam-vxlan-gpe].

6. IOAM Data Export

IOAM nodes collect information for packets traversing a domain that supports IOAM. IOAM decapsulating nodes as well as IOAM transit nodes can choose to retrieve IOAM information from the packet, process the information further and export the information using e.g., IPFIX.

Raw data export of IOAM data using IPFIX is discussed in [I-D.spiegel-ippm-ioam-rawexport]. "Raw export of IOAM data" refers to a mode of operation where a node exports the IOAM data as it is received in the packet. The exporting node neither interprets, aggregates nor reformats the IOAM data before it is exported. Raw export of IOAM data is to support an operational model where the processing and interpretation of IOAM data is decoupled from the operation of encapsulating/updating/decapsulating IOAM data, which is also referred to as IOAM data-plane operation. The figure below shows the separation of concerns for IOAM export: Exporting IOAM data is performed by the "IOAM node" which performs IOAM data-plane operation, whereas the interpretation of IOAM data is performed by one or several IOAM data processing systems. The separation of concerns is to off-load interpretation, aggregation and formatting of IOAM data from the node which performs data-plane operations. In other words, a node which is focused on data-plane operations, i.e. forwarding of packets and handling IOAM data will not be tasked to also interpret the IOAM data, but can leave this task to another system or a set of systems. For scalability reasons, a single IOAM node could choose to export IOAM data to several IOAM data processing

systems. Similarly, there several monitoring systems or analytics systems can be used to further process the data received from the IOAM preprocessing systems. Figure 3 shows an overview of IOAM export, including IOAM data processing systems and monitoring/ analytics systems.

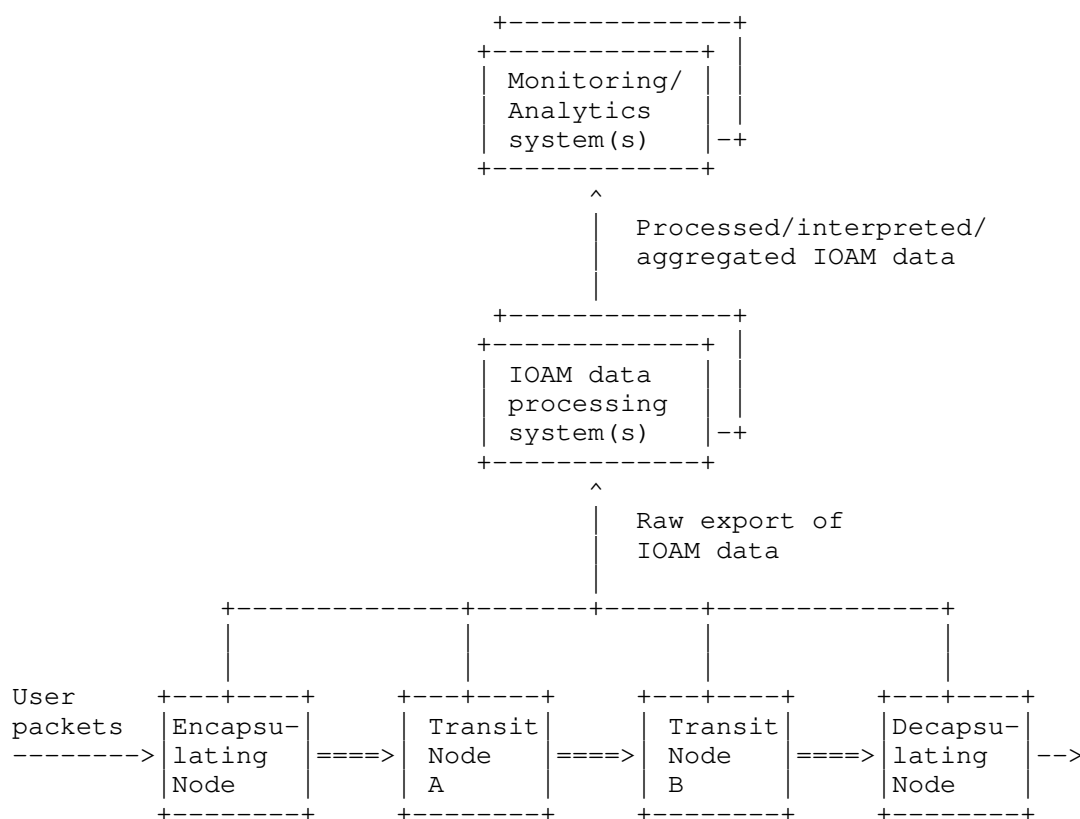


Figure 3: IOAM framework with data export

7. IOAM Deployment Considerations

This section discusses several aspects of an IOAM deployment, including IOAM Namespaces, IOAM Layering, traffic-sets that IOAM is applied to and IOAM loopback mode.

7.1. IOAM Namespaces

IOAM-Namespaces add further context to IOAM-Option-Types and associated IOAM-Data-Fields. IOAM-Namespaces support several different uses:

- o IOAM-Namespaces can be used by an operator to distinguish different operational domains. Devices at domain edges can filter on Namespace-IDs to provide for proper IOAM-Domain isolation.
- o IOAM-Namespaces provide additional context for IOAM-Data-Fields and thus ensure that IOAM-Data-Fields are unique and can be interpreted properly by management stations or network controllers. While, for example, the node identifier field does not need to be unique in a deployment (e.g. an operator may wish to use different node identifiers for different IOAM layers, even within the same device; or node identifiers might not be unique for other organizational reasons, such as after a merger of two formerly separated organizations), the combination of node_id and Namespace-ID should always be unique. Similarly, IOAM-Namespaces can be used to define how certain IOAM-Data-Fields are interpreted: IOAM offers three different timestamp format options. The Namespace-ID can be used to determine the timestamp format. IOAM-Data-Fields (e.g. buffer occupancy) which do not have a unit associated are to be interpreted within the context of a IOAM-Namespace.
- o IOAM-Namespaces can be used to identify different sets of devices (e.g., different types of devices) in a deployment: If an operator desires to insert different IOAM-Data-Fields based on the device, the devices could be grouped into multiple IOAM-Namespaces. This could be due to the fact that the IOAM feature set differs between different sets of devices, or it could be for reasons of optimized space usage in the packet header. It could also stem from hardware or operational limitations on the size of the trace data that can be added and processed, preventing collection of a full trace for a flow.
 - * Assigning different IOAM Namespace-IDs to different sets of nodes or network partitions and using the Namespace-ID as a selector at the IOAM encapsulating node, a full trace for a flow could be collected and constructed via partial traces in different packets of the same flow. Example: An operator could choose to group the devices of a domain into two IOAM-Namespaces, in a way that at average, only every second hop would be recorded by any device. To retrieve a full view of the deployment, the captured IOAM-Data-Fields of the two IOAM-Namespaces need to be correlated.

- * Assigning different IOAM Namespace-IDs to different sets of nodes or network partitions and using a separate instance of an IOAM-Option-Type for each Namespace-ID, a full trace for a flow could be collected and constructed via partial traces from each IOAM-Option-Type in each of the packets in the flow. Example: An operator could choose to group the devices of a domain into two IOAM-Namespace, in a way that each IOAM-Namespace is represented by one of two IOAM-Option-Types in the packet. Each node would record data only for the IOAM-Namespace that it belongs to, ignoring the other IOAM-Option-Type with a IOAM-Namespace to which it doesn't belong. To retrieve a full view of the deployment, the captured IOAM-Data-Fields of the two IOAM-Namespace need to be correlated.

7.2. IOAM Layering

If several encapsulation protocols (e.g., in case of tunneling) are stacked on top of each other, IOAM-Data-Fields could be present in different protocol fields at different layers. Layering allows operators to instrument the protocol layer they want to measure. The behavior follows the ships-in-the-night model, i.e. IOAM-Data-Fields in one layer are independent from IOAM-Data-Fields in another layer. Or in other words: Even though the term "layering" often implies some form of hierarchy and relationship, in IOAM, layers are independent from each other and don't assume any relationship among them. The different layers could, but do not have to share the same IOAM encapsulation mechanisms. Similarly, the semantics of the IOAM-Data-Fields, can, do not have to be associated to across different layers. For example, a node which inserts node-id information into two different layers could use "node-id=10" for one layer and "node-id=1000" for the second layer.

Figure 4 shows an example of IOAM layering. The figure shows a Geneve tunnel carried over IPv6 which starts at node A and ends at node D. IOAM information is encapsulated in IPv6 as well as in Geneve. At the IPv6 layer, node A is IOAM encapsulating node (into IPv6), node D is the IOAM decapsulating node and node B and node C are IOAM transit nodes. At the Geneve layer, node A is IOAM encapsulating node (into Geneve) and node D is IOAM decapsulating node (from Geneve). The use of IOAM at both layers as shown in the example here could be used to reveal which nodes of an underlay (here the IPv6 network) are traversed by tunneled packet in an overlay (here the Geneve network) - which assumes that the IOAM information encapsulated by nodes A and D into Geneve and IPv6 is associated to each other.

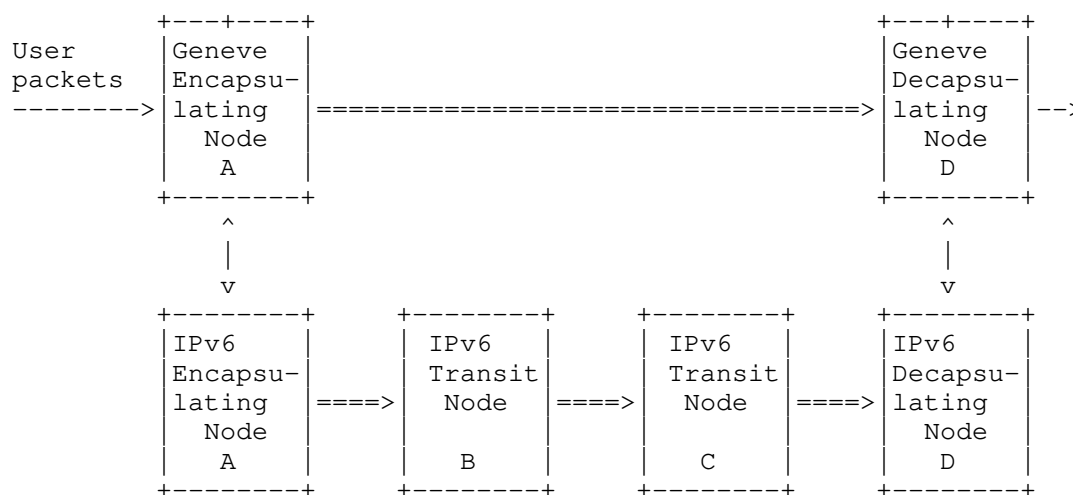


Figure 4: IOAM layering example

7.3. IOAM Trace Option Types

IOAM offers two different IOAM Option-Types for tracing:

"Incremental" Trace-Option-Type and "Pre-allocated" Trace-Option-Type. "Incremental" refers to a mode of operation where the packet is expanded at every IOAM node that adds IOAM-Data-Fields. "Pre-Allocated" describes a mode of operation where the IOAM encapsulating node allocates room for all IOAM-Data-Fields in the entire IOAM domain. More specifically:

Pre-allocated Trace-Option: This trace option is defined as a container of node data fields with pre-allocated space for each node to populate its information. This option is useful for implementations where it is efficient to allocate the space once and index into the array to populate the data during transit (e.g., software forwarders often fall into this class).

Incremental Trace-Option: This trace option is defined as a container of node data fields where each node allocates and pushes its node data immediately following the option header.

A deployment can choose to configure and support one or both of the IOAM Trace-Option-Types. The operator decides by means of configuration which Trace-Option-Type(s) will be used for a particular domain. Deployments can mix devices which include either the Incremental Trace-Option-Type or the Pre-allocated Trace-Option-Type, e.g. in case different types of packet forwarders and

associated different types of IOAM implementations exist in a deployment. As a result, both Option-Types can be present in a packet. IOAM decapsulating nodes remove both types of Trace-Option-Types from the packet.

The two different Option-Types cater to different packet forwarding infrastructures and are to allow an optimized implementation of IOAM tracing:

Pre-allocated Trace-Option: For some implementations of packet forwarders it is efficient to allocate the space for the maximum number of nodes that IOAM Trace Data-Fields should be collected from and insert/update information in the packet at flexible locations, based on a pointer retrieved from a field in the packet. The IOAM encapsulating node allocates an array of the size of the maximum number of nodes that IOAM Trace Data-Fields should be collected from. IOAM transit nodes index into the array to populate the data during transit. Software forwarders often fall into this class of packet forwarder implementations. The maximum number of nodes that IOAM information could be collected from is configured by the operator on the IOAM encapsulating node. The operator has to ensure that the packet with the pre-allocated array that carries the IOAM Data-Fields does not exceed the MTU of any of the links in the IOAM domain.

Incremental Trace-Option: Looking up a pointer contained in the packet and inserting/updating information at a flexible location in the packet as a result of the pointer lookup is costly for some forwarding infrastructures. Hardware-based packet forwarding infrastructures often fall into this category. Consequently, hardware-based packet forwarders could choose to support the incremental IOAM-Trace-Option-Type. The incremental IOAM-Trace-Option-Type eliminates the need for the IOAM transit nodes to read the full array in the Trace-Option-Type and allows packets to grow to the size of the MTU of the IOAM domain. IOAM transit nodes will expand the packet and insert the IOAM-Data-Fields as long as there is space available in the packet, i.e. as long as the size of the packet stays within the bounds of the MTU of any of the links in the IOAM domain. There is no need for the operator to configure the IOAM encapsulation node with the maximum number of nodes that IOAM information could be collected from. The operator has to ensure that the minimum MTU of any of the links in the IOAM domain is known to all IOAM transit nodes.

7.4. Traffic-sets That IOAM Is Applied To

IOAM can be deployed on all or only on subsets of the live user traffic, e.g. per interface, based on an access control list or flow specification defining a specific set of traffic, etc.

7.5. IOAM Loopback Mode

IOAM Loopback is used to trigger each transit device along the path of a packet to send a copy of the data packet back to the source. Loopback allows an IOAM encapsulating node to trace the path to a given destination, and to receive per-hop data about both the forward and the return path. Loopback is enabled by the encapsulating node setting the loopback flag. Looped-back packets use the source address of the original packet as destination address and the address of the node which performs the loopback operation as source address. Nodes which loop back a packet clear the loopback flag before sending the copy back towards the source. Loopback applies to IOAM deployments where the encapsulating node is either a host or the start of a tunnel: For details on IOAM loopback, please refer to [I-D.ietf-ippm-ioam-flags].

7.6. IOAM Active Mode

The IOAM active mode flag indicates that a packet is an active OAM packet as opposed to regular user data traffic. Active mode is expected to be used for active measurement using IOAM. Example use-cases include:

- o Endpoint detailed active measurement: Synthetic probe packets are sent between the source and destination, traversing the IOAM domain. These probe packets include a Trace Option-Type (i.e., either incremental or pre-allocated). Since the probe packets are sent between the endpoints, these packets are treated as data packets by the IOAM domain, and do not require special treatment at the IOAM layer. The encapsulating node can choose to set the Active flag, providing an explicit indication that these probe packets are meant for telemetry collection.
- o IOAM active measurement using probe packets: Probe packets are generated and transmitted by the IOAM encapsulating node, and are expected to be terminated by the decapsulating node. Probe packets include a Trace Option-Type (i.e., either incremental or pre-allocated) which has its Active flag set, indicating that the decapsulating node must terminate them.
- o IOAM active measurement using replicated data packets: Probe packets are created by the encapsulating node by selecting some or

all of the en route data packets and replicating them. A selected data packet that is replicated, and its (possibly truncated) copy is forwarded with one or more IOAM option, while the original packet is forwarded normally, without IOAM options. To the extent possible, the original data packet and its replica are forwarded through the same path. The replica includes a Trace Option-Type that has its Active flag set, indicating that the decapsulating node should terminate it.

For details on IOAM active mode, please refer to [I-D.ietf-ippm-ioam-flags].

7.7. Brown Field Deployments: IOAM Unaware Nodes

A network can consist of a mix of IOAM aware and IOAM unaware nodes. The encapsulation of IOAM-Data-Fields into different protocols (see also Section 5) are defined such that data packets that include IOAM-Data-Fields do not get dropped by IOAM unaware nodes. For example, packets which contain the IOAM-Trace-Option-Types in IPv6 Hop by Hop extension headers are defined with bits to indicate "00 - skip over this option and continue processing the header". This will ensure that when a node that is IOAM unaware receives a packet with IOAM-Data-Fields included, does not drop the packet.

Deployments which leverage the IOAM-Trace-Option-Type(s) could benefit from the ability to detect the presence of IOAM unaware nodes, i.e. nodes which forward the packet but do not update/add IOAM-Data-Fields in IOAM-Trace-Option-Type(s). The node data that is defined as part of the IOAM-Trace-Option-Type(s) includes a Hop_Lim field associated to the node identifier to detect missed nodes, i.e. "holes" in the trace. Monitoring/Analytics system(s) could utilize this information to account for the presence of IOAM unaware nodes in the network.

8. IOAM Manageability

The YANG model for configuring IOAM in network nodes which support IOAM is defined in [I-D.zhou-ippm-ioam-yang].

A deployment can leverage IOAM profiles is to limit the scope of IOAM features, allowing simpler implementation, verification, and interoperability testing in the context of specific use cases that do not require the full functionality of IOAM. An IOAM profile defines a use case or a set of use cases for IOAM, and an associated set of rules that restrict the scope and features of the IOAM specification, thereby limiting it to a subset of the full functionality. IOAM profiles are defined in [I-D.mizrahi-ippm-ioam-profile].

9. IANA Considerations

This document does not request any IANA actions.

10. Security Considerations

As discussed in [RFC7276], a successful attack on an OAM protocol in general, and specifically on IOAM, can prevent the detection of failures or anomalies, or create a false illusion of nonexistent ones.

The Proof of Transit Option-Type (Section 4.2) is used for verifying the path of data packets. The security considerations of POT are further discussed in [I-D.ietf-sfc-proof-of-transit].

Security considerations related to the use of IOAM flags, in particular the loopback flag are found in [I-D.ietf-ippm-ioam-flags].

IOAM data can be subject to eavesdropping. Although the confidentiality of the user data is not at risk in this context, the IOAM data elements can be used for network reconnaissance, allowing attackers to collect information about network paths, performance, queue states, buffer occupancy and other information. Recon is an improbable security threat in an IOAM deployment that is within a confined physical domain. However, in deployments that are not confined to a single LAN, but span multiple inter-connected sites (for example, using an overlay network), the inter-site links can be secured (e.g., by IPsec) in order to avoid external eavesdropping. Another possible mitigation approach is to use the "direct exporting" mode [I-D.ietf-ippm-ioam-direct-export]. In this case the IOAM related trace information would not be available in the customer data packets, but would trigger exporting of (secured) packet-related IOAM information at every node. IOAM data export and securing IOAM data export is outside the scope of this document.

IOAM can be used as a means for implementing Denial of Service (DoS) attacks, or for amplifying them. For example, a malicious attacker can add an IOAM header to packets or modify an IOAM header in en route packets in order to consume the resources of network devices that take part in IOAM or collectors that analyze the IOAM data. Another example is a packet length attack, in which an attacker pushes headers associated with IOAM Option-Types into data packets, causing these packets to be increased beyond the MTU size, resulting in fragmentation or in packet drops. Such DoS attacks can be mitigated by deploying IOAM in confined administrative domains, and by defining performance limits on IOAM encapsulation and IOAM exporting. By limiting the rate and/or percentage of packets that

are subject to IOAM encapsulation and the rate of exported traffic, it is possible to confine the impact of such attacks.

Since IOAM options may include timestamps, if network devices use synchronization protocols then any attack on the time protocol [RFC7384] can compromise the integrity of the timestamp-related data fields. Synchronization attacks can be mitigated by combining a secured time distribution scheme, e.g., [RFC8915], and by using redundant clock sources [RFC5905] and/or redundant network paths for the time distribution protocol [RFC8039].

At the management plane, attacks may be implemented by misconfiguring or by maliciously configuring IOAM-enabled nodes in a way that enables other attacks. Thus, IOAM configuration should be secured in a way that authenticates authorized users and verifies the integrity of configuration procedures.

Notably, IOAM is expected to be deployed in specific network domains, thus confining the potential attack vectors to within the network domain. Indeed, in order to limit the scope of threats to within the current network domain the network operator is expected to enforce policies that prevent IOAM traffic from leaking outside of the IOAM domain, and prevent IOAM data from outside the domain to be processed and used within the domain. Note that the Immediate Export mode (reference to be added in a future revision) can mitigate the potential threat of IOAM data leaking through data packets.

11. Acknowledgements

The authors would like to thank Tal Mizrahi, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Barak Gafni, Karthik Babu Harichandra Babu, Akshaya Nadahalli, LJ Wobker, Erik Nordmark, Vengada Prasad Govindan, Andrew Yourtchenko, Aviv Kfir, Tianran Zhou, Zhenbin (Robin), Joe Clarke, Al Morton, Tom Herbet, Haoyu song, and Mickey Spiegel for the comments and advice on IOAM.

12. References

12.1. Normative References

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

12.2. Informative References

[I-D.ali-spring-ioam-srv6]

Ali, Z., Gandhi, R., Filsfils, C., Brockners, F., Nainar, N., Pignataro, C., Li, C., Chen, M., and G. Dawra, "Segment Routing Header encapsulation for In-situ OAM Data", draft-ali-spring-ioam-srv6-03 (work in progress), November 2020.

[I-D.brockners-ippm-ioam-geneve]

Brockners, F., Bhandari, S., Govindan, V. P., Pignataro, C., Nainar, N. K., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Lapukhov, P., Gafni, B., Kfir, A., and M. Spiegel, "Geneve encapsulation for In-situ OAM Data", draft-brockners-ippm-ioam-geneve-05 (work in progress), November 2020.

[I-D.brockners-ippm-ioam-vxlan-gpe]

Brockners, F., Bhandari, S., Govindan, V. P., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., and M. Spiegel, "VXLAN-GPE Encapsulation for In-situ OAM Data", draft-brockners-ippm-ioam-vxlan-gpe-03 (work in progress), November 2019.

[I-D.gandhi-spring-ioam-sr-mpls]

Gandhi, R., Ali, Z., Filsfils, C., Brockners, F., Wen, B., and V. Kozak, "Segment Routing with MPLS Data Plane Encapsulation for In-situ OAM Data", draft-gandhi-spring-ioam-sr-mpls-02 (work in progress), August 2019.

[I-D.ietf-ippm-ioam-data]

Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-12 (work in progress), February 2021.

[I-D.ietf-ippm-ioam-direct-export]

Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F., Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ OAM Direct Exporting", draft-ietf-ippm-ioam-direct-export-03 (work in progress), February 2021.

[I-D.ietf-ippm-ioam-flags]

Mizrahi, T., Brockners, F., Bhandari, S., Sivakolundu, R., Pignataro, C., Kfir, A., Gafni, B., Spiegel, M., and J. Lemon, "In-situ OAM Flags", draft-ietf-ippm-ioam-flags-04 (work in progress), February 2021.

- [I-D.ietf-ippm-ioam-ipv6-options]
Bhandari, S., Brockners, F., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., Spiegel, M., Krishnan, S., Asati, R., and M. Smith, "In-situ OAM IPv6 Options", draft-ietf-ippm-ioam-ipv6-options-05 (work in progress), February 2021.
- [I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [I-D.ietf-nvo3-vxlan-gpe]
(Editor), F. M., (editor), L. K., and U. E. (editor), "Generic Protocol Extension for VXLAN (VXLAN-GPE)", draft-ietf-nvo3-vxlan-gpe-11 (work in progress), March 2021.
- [I-D.ietf-sfc-ioam-nsh]
Brockners, F. and S. Bhandari, "Network Service Header (NSH) Encapsulation for In-situ OAM (IOAM) Data", draft-ietf-sfc-ioam-nsh-05 (work in progress), December 2020.
- [I-D.ietf-sfc-proof-of-transit]
Brockners, F., Bhandari, S., Mizrahi, T., Dara, S., and S. Youell, "Proof of Transit", draft-ietf-sfc-proof-of-transit-08 (work in progress), November 2020.
- [I-D.ioametal-ippm-6man-ioam-ipv6-deployment]
Bhandari, S., Brockners, F., Mizrahi, T., Kfir, A., Gafni, B., Spiegel, M., Krishnan, S., and M. Smith, "Deployment Considerations for In-situ OAM with IPv6 Options", draft-ioametal-ippm-6man-ioam-ipv6-deployment-03 (work in progress), March 2020.
- [I-D.mizrahi-ippm-ioam-profile]
Mizrahi, T., Brockners, F., Bhandari, S., Sivakolundu, R., Pignataro, C., Kfir, A., Gafni, B., Spiegel, M., Zhou, T., and J. Lemon, "In Situ OAM Profiles", draft-mizrahi-ippm-ioam-profile-04 (work in progress), February 2021.
- [I-D.spiegel-ippm-ioam-rawexport]
Spiegel, M., Brockners, F., Bhandari, S., and R. Sivakolundu, "In-situ OAM raw data export with IPFIX", draft-spiegel-ippm-ioam-rawexport-04 (work in progress), November 2020.

- [I-D.weis-ippm-ioam-eth]
Weis, B., Brockners, F., Hill, C., Bhandari, S., Govindan, V. P., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., and M. Spiegel, "EtherType Protocol Identification of In-situ OAM Data", draft-weis-ippm-ioam-eth-04 (work in progress), May 2020.
- [I-D.zhou-ippm-ioam-yang]
Zhou, T., Guichard, J., Brockners, F., and S. Raghavan, "A YANG Data Model for In-Situ OAM", draft-zhou-ippm-ioam-yang-08 (work in progress), July 2020.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <<https://www.rfc-editor.org/info/rfc7276>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8039] Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multipath Time Synchronization", RFC 8039, DOI 10.17487/RFC8039, December 2016, <<https://www.rfc-editor.org/info/rfc8039>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/info/rfc8915>>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari (editor)
Thoughtspot
3rd Floor, Indiqube Orion, 24th Main Rd, Garden Layout, HSR Layout
Bangalore, KARNATAKA 560 102
India

Email: shwetha.bhandari@thoughtspot.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Tal Mizrahi (editor)
Huawei
8-2 Matam
Haifa 3190501
Israel

Email: tal.mizrahi.phd@gmail.com

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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on October 25, 2021.

Copyright Notice

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

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

Table of Contents

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

1. Terminology

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

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

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

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

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

DAG: Directed Acyclic Graph.

ECMP: Equal Cost Multiple Paths

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

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

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

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

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

Metric: Information retrieved from a network device.

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

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

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

Service Instance: A specific instance of a service.

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

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

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

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

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

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

2. Introduction

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

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

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

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

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

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

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

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

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

3. Architecture

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

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

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

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

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

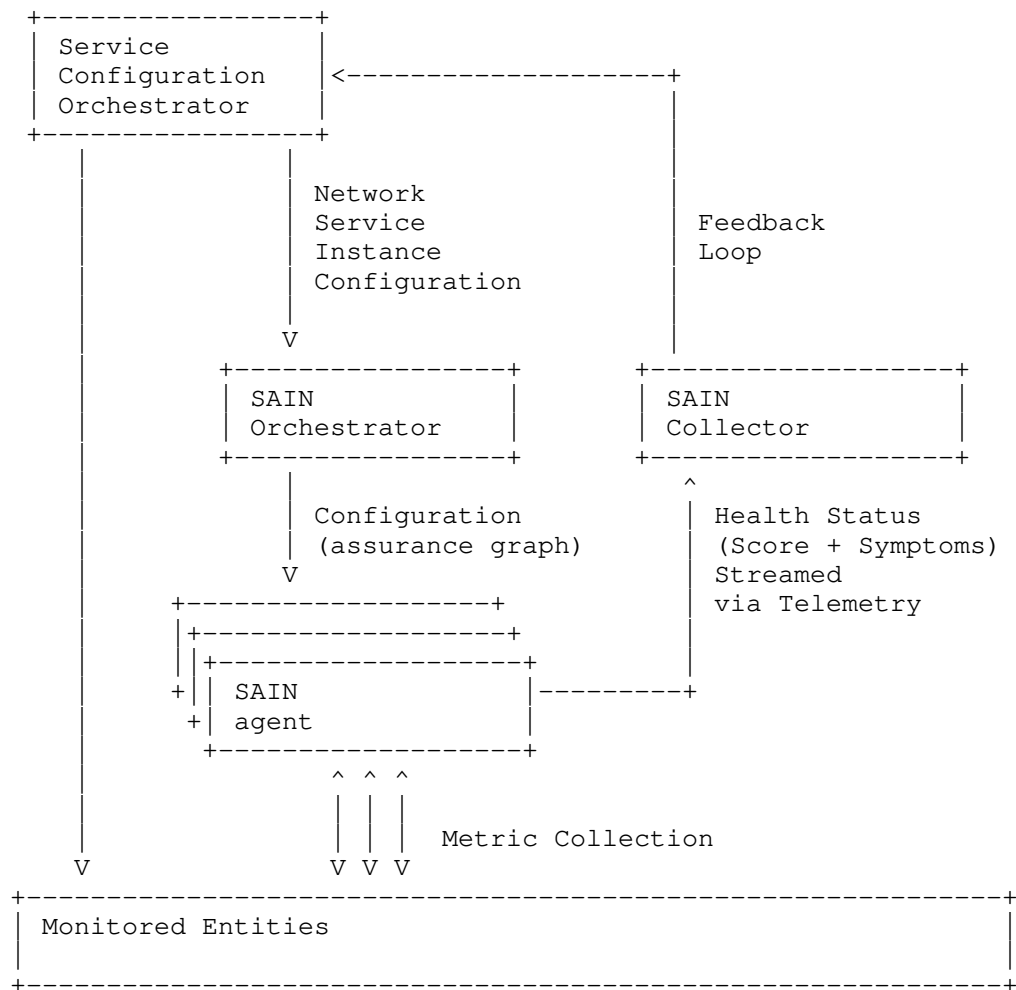


Figure 1: SAIN Architecture

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

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

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

3.1. Decomposing a Service Instance Configuration into an Assurance Graph

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

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

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

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

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

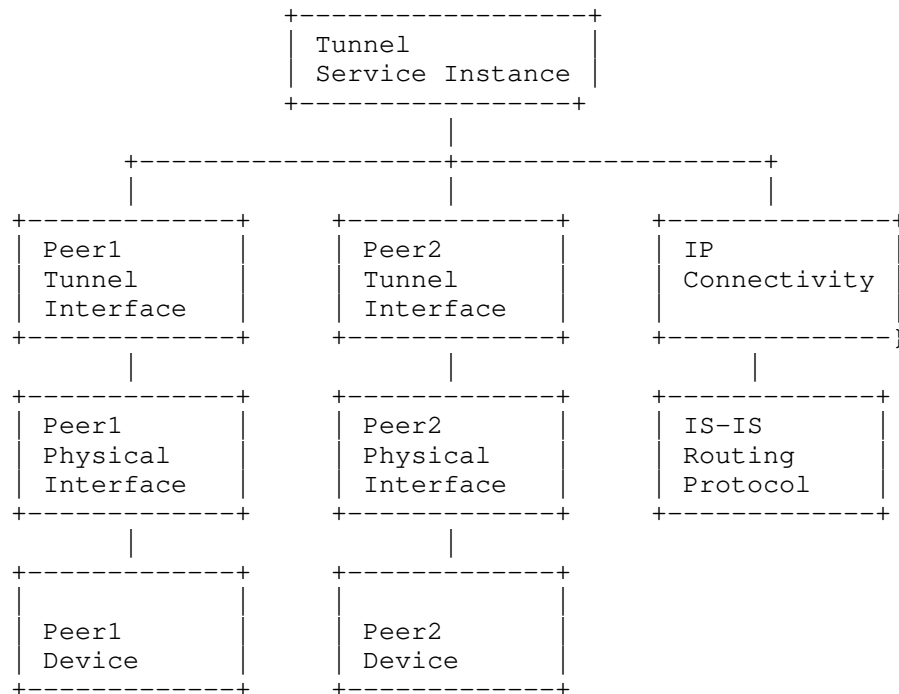


Figure 2: Assurance Graph Example

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

3.2. Intent and Assurance Graph

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

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

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

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

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

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

3.3. Subservices

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

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

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

3.4. Building the Expression Graph from the Assurance Graph

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

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

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

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

3.5. Building the Expression from a Subservice

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

3.6. Open Interfaces with YANG Modules

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

This module is intended for the following use cases:

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

3.7. Handling Maintenance Windows

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

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

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

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

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

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

3.8. Flexible Architecture

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

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

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

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

3.9. Timing

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

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

3.10. New Assurance Graph Generation

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

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

4. Security Considerations

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

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

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

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

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

- o Youssef El Fathi
- o Eric Vyncke

7. Open Issues

Refer to the Intent-based Networking NMRG documents

8. References

8.1. Normative References

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

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

8.2. Informative References

- [I-D.claise-opsawg-service-assurance-yang] Claise, B. and J. Quilbeuf, "Service Assurance for Intent-based Networking Architecture", February 2020.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8907] Dahm, T., Ota, A., Medway Gash, D., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <<https://www.rfc-editor.org/info/rfc8907>>.

Appendix A. Changes between revisions

v02 - v03

- o Timing Concepts
- o New Assurance Graph Generation

v01 - v02

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

v00 - v01

- o Terminology clarifications
- o Figure 1 improved

Acknowledgements

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

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Jean Quilbeuf
Independent

Email: jean@quilbeuf.net

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

Email: diego.r.lopez@telefonica.com

Dan Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

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

Email: tarumuga@cisco.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: 4 October 2020

A. Gray
Charter Communications
L.J. Wobker
Cisco Systems
2 April 2020

Sampled Traffic Streaming
draft-gray-sampled-streaming-03

Abstract

This document standardizes both 1) a means of requesting a stream of packet samples from any device generating, routing, or forwarding traffic, and 2) receiving metadata information from the network element about these packet samples, and the structure of said stream metadata. A main design requirement is to provide network elements with widely varying capabilities (e.g., ASICs, NPUs, NICs, vSwitches, CPUs) a mechanism to sample and export packets at high rates, by allowing communication of the specific bit formats of internal data headers applied to the packet flow, in a way that enhances interoperability between traffic sources and sinks. Historically, Netflow and similar mechanisms have been used for these use cases; however, the increasing packet rates of very high-speed devices and increasing variance in the information available to data planes lends itself to both a less-prescriptive set of packet formats as well as a decoupling of the sampling action from the collection and analysis mechanisms.

Status of This Memo

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

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

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

This Internet-Draft will expire on 4 October 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Terminology	3
1.3. Motivation for Disaggregation of Telemetry	3
1.4. Comparisons with PSAMP	4
2. Use Cases	5
2.1. Use Case 1: Traffic Analytics	5
2.2. Use Case 2: Network Behavior Verification	6
2.3. Use Case 3: Standardization	6
2.4. Use Case 4: Security Automation	6
3. Stream Setup	7
3.1. Client queries Replicator for Points	7
3.2. Client submits a request to the Replicator	8
3.2.1. Filtering Details	9
3.3. Replicator offers Proposals	9
3.4. Client selects a Proposal	10
3.5. Ending sampling and cleanup	11
4. Data Stream Format	11
5. IANA Considerations	15
6. Security Considerations	15
7. Acknowledgments	16
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Appendix A. Yang Model Tree Reference	17
Appendix B. Yang Model	21
Authors' Addresses	39

1. Introduction

1.1. 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].

1.2. Terminology

The following terms are used within this document:

Client: The device configuring the Replicator.

Receiver: The device receiving the packet stream.

Replicator: The device performing the actual packet replication, as requested by a Client, and sending the resulting replicated packet stream to a Receiver.

Point: The location inside the Replicator (e.g., a forwarding ASIC) that performs the actual packet replication. There may be multiple physical interfaces serviced by one Point, or one interface may be serviced by multiple Points, that may have different capabilities.

1.3. Motivation for Disaggregation of Telemetry

A key concept for this proposal is to enable very high rate sample generation for network elements, while at the same time separating the sampling mechanism itself from specific analysis or transport protocols. If we separate the component functions of how these problems have been traditionally solved, these functions lend themselves to being viewed as a layered stack such as the one in the figure:

Figure: Packet sampling and analysis viewed as a layered stack

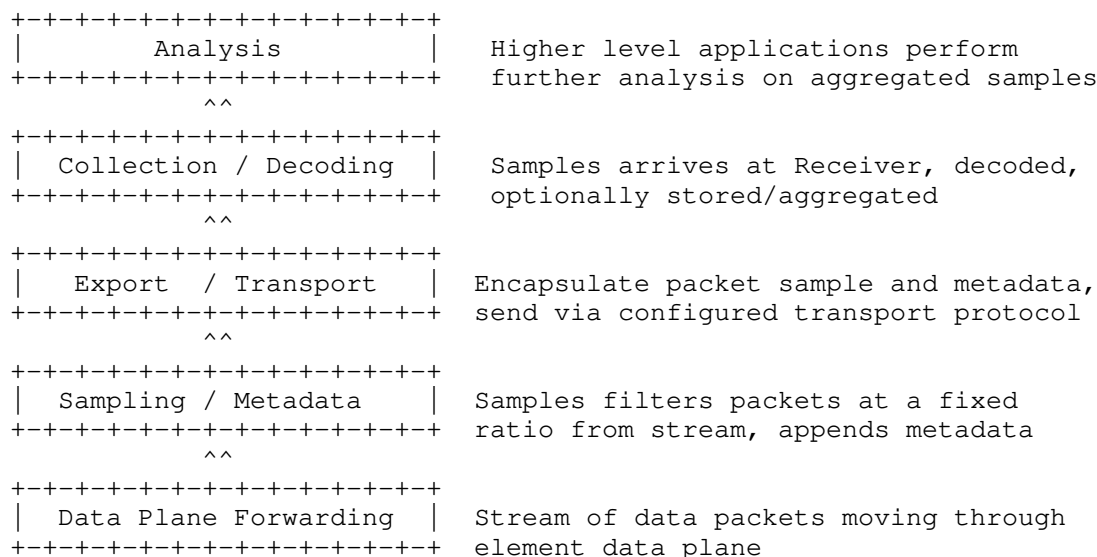


Figure 1

The primary advantage of the stack model is the ability to disaggregate functions from each other. For example, providing a self-describing, flexible format for the metadata abstracts the data plane -- in other words the upper layers do not need to know how many bits wide a metadata field is, they only need to know that it is present and the semantics. Separating the transport function allows for multiple use cases: a router wishing to sample packets for internal consumption within the same system might use a locally defined (perhaps even proprietary) transport header, while putting the sampled metadata and packet into a UDP packet allows for it to be transported to any IP-reachable collector, regardless of the geographic or topological distance from the Replicator itself.

This document standardizes the "Sampling / Metadata" and "Export / Transport" components of the above stack.

1.4. Comparisons with PSAMP

Packet Sampling (PSAMP) from RFC 5476 [RFC5476] shares some of the characteristics of Sampled Streaming, and parts of its YANG model as documented in RFC 6728 [RFC6728] are in fact imported into this one to share concepts where possible (notably re-using the concepts of observation points and selectors). However, Sampled Streaming differs primarily in the ability to include information that is normally internal to device that provides information about the

packet's handling through the device, and to have the Replicator specify the outgoing packet format in a very dynamic fashion that suits itself as best as possible. This is done to allow this replication to be done natively on relatively low feature set forwarding hardware and to ensure the only usage of high-capability CPU resources on the Replicator is in the initial setup and negotiation. All other aspects have been made to allow the Replicator to do the least amount of work as possible, to extract as much information as possible, and get it sent to the Receiver who is presumed to have orders of magnitudes greater compute capability available. Other changes to the setup and configuration are wrapped around this primary goal.

2. Use Cases

This document is designed around the following current and foreseeable use cases that operators have today.

2.1. Use Case 1: Traffic Analytics

Operators typically use a mix of NetFlow, IPFIX, and in-line traffic samplers spread throughout the network to gather data for analytics. With the next generation of hardware, 400Gb/s interfaces are becoming available, with higher data rates under development in their respective standards bodies. This will require at least an augmentation of any in-line traffic samplers, which are quite expensive. Additionally, the pace of growth in the data plane is outgrowing the pace of growth of the control plane. This is especially visible with relatively control plane or CPU-heavy protocols such as NetFlow, where current sampling rates are simply not going to be sustainable long-term, primarily due to on-box control plane hardware limitations. Being able to capture a filtered, sampled collection of actual packets throughout the network is very valuable for understanding how the network is being used, to provide hard data to justify network topology augments and/or technology changes.

This proposal addresses this use case by: 1) making the data replication mechanism as simple as possible, reducing the need for high levels of complexity in the data plane; 2) decoupling the sampling/collection of packets from the analysis, which in turn allows for the analysis to be performed on distributed, horizontally-scalable platforms rather than being constrained to the compute and storage capabilities of a local network element.

2.2. Use Case 2: Network Behavior Verification

This use case focuses on the potential ability to have the ASICs stream discarded packets, along with an indication as to the reason for the drop. With fields denoting the reason for dropped packets such as QoS policies, buffer contention, ACLs, etc., such discarded traffic could be streamed (potentially at a sampling rate of 1:1, i.e. every packet) off-box for analysis to determine if the observed behavior was expected, or trigger alerts that QoS policies may be having adverse effects on the network. The ability to include the packet payload provides additional context, allowing examination of the platform behavior and affected policies.

This proposal addresses this use case by allowing samplers which have such capabilities to communicate to the receiver: 1) drop codes(reasons) that are known, 2) the semantics of those codes, and 3) the specific bit formats for the receiver to use when decoding.

2.3. Use Case 3: Standardization

Standardizing the way these data streams are formed and communicated between the Replicators, Clients, and Receivers in a fashion that allows vendors flexibility in what work the ASIC has to do to support sampled streaming (by allowing communicating of an extremely dynamic header in a manner than control planes can manage) allows systems to be used between all platforms in an interoperable fashion. The alternative is to build independent systems for each packet replication solution that may end up being developed, resulting in much higher costs for an overall solution.

This proposal addresses this use case by allowing the sampled packet header to provide varying metadata fields, without mandating specific positions or widths. This arrangement of fields and their format is a function of the Replicator, and information about how to handle this data is exchanged between the Replicator, Client, and Receiver at the initialization of the session. The motivation for such latitude in encoding and sizing is quite intentional, as it permits widely varying capabilities within the Replicators.

2.4. Use Case 4: Security Automation

An automated security platform can utilize this proposal to set up a "normal security analysis" stream at a very low sampling rate (for example, 1 in 20,000) for constant monitoring at various points throughout the network. Upon seeing something it deems 'interesting', or by manual input, it can add in an additional, targeted, stream, at a very high sampling rate (potentially 1:1) for detailed analysis and mitigation efforts.

Examples of past incidents where this may have been useful are the NTP MONLIST attacks, DNS attacks, or DDoS attacks (although 1:1 would most likely not be used in a DDoS case, unless performing the initial data collection).

The security platform could potentially then use the collected packets to generate an auto-mitigation plan based on heuristics (i.e., 99% of this sudden burst of traffic has something in common, deploy mitigation targeting that.)

3. Stream Setup

The configuration and setup between the Client and the Replicator utilizes the YANG model as listed in Appendix B and any supported configuration method (NETCONF, RESTCONF, gRPC, etc.). The tree output of this model, as provided in Appendix A is provided as an aid to understanding the interactions and tree structure as described in this document.

3.1. Client queries Replicator for Points

A Client MUST first request from the Replicator the available configurations via the 'points' branch, which provides the following information:

- * 'name' - The name of the Point. This serves as a key, and SHOULD NOT be interpreted by software as anything other than a possibly-human-readable uniquely identifying value. A Replicator MAY choose to use an internal path, an encoded address, or any other value of its choosing.
- * 'interfaces' - The physical interfaces this Point is servicing. A Replicator MAY offer the same interfaces under different Points, with a different set of options. A Replicator MAY not offer a Point for every interface available on the system.
- * 'filters' - What filters can be applied (for example, against certain IP fields, against parts of the frame, etc.). A Replicator MAY not be able to honor every combination of filters submitted in a request, or MAY not offer any filtering capability at all. A Replicator MAY only be able to support a limited number of filters, which MAY be returned in in the 'max-filters' branch.
- * 'min-ratio' and 'max-ratio' - Minimum and maximum sampling rates possible at this point. These are provided as a number N, denoting one sample will be returned for every N. A Replicator MAY not be able to offer a 'min-ratio' of 1 (i.e. every packet).

- * 'samplers' - A list of any current samplers already active on this Point as requested by this Client, and the branch manipulated in the next section. A Replicator SHOULD NOT inform a Client about the sampling sessions from other Clients.
- * Optionally, the maximum frame length the Point can replicate into the sample in 'max-frame-length-copy'.
- * Optionally, the maximum offset into a frame the Point can inspect in 'max-frame-depth-inspect'.
- * Optionally, the maximum number of samplers that this Point can accommodate in 'max-samplers'. A Client MUST still check for success, as highly complex filters may reduce the amount of replication the Point can do from this stated maximum.

3.2. Client submits a request to the Replicator

The Client then can request one or more streams to be set up on the Replicator, taking into consideration the provided information. This is performed by sending a request via adding an entry to the 'samplers' list in the 'points' branch and filling in the parameters listed below:

- * 'name' MUST be unique in the list, and MAY be any valid string value up to 255 characters. The Replicator MUST isolate namespaces between Clients (as one Client SHOULD NOT be able to see other Clients' entries).
- * 'destination' sets the transport mechanism and Receiver address. It should be noted that the Client and Receiver MAY be separate devices. The mechanism of exchanging information between the Client and Receiver about this setup process is outside the scope of this document. At present, the only supported transport mechanism is a UDP tunnel, as detailed below in Section 4.
- * 'client-heartbeat' MUST be set to 0.
- * The desired sampling rate ('ratio'), along with what degree of variance the Client can accept ('min-ratio' and 'max-ratio'). For example, the client may request a 1 in 2000 rate, but specify a range in the variance of 1900-2100. A proposal may come back with the sampling rate offered of 1 in 2048, due to restrictions on the Replicator.
- * Optionally, one, or more filters in the 'filters' container, as seen in the 'filter-type' typedef in the Yang model. Generally, a

Client would filter at least on a specific interface and direction, but many other filter options are possible.

When the client is done with its configuration, it MUST set 'status' to the 'client-request-complete' value, and the 'request' branch MUST be read-only from this point forward.

3.2.1. Filtering Details

The filtering discussed above is designed to be as flexible as the Replicator can realistically support. There are a few cases worth discussing in detail, which are covered here.

3.2.1.1. Interfaces

All of the use cases focus on filtering to specific interface(s) to filter on. A Replicator MAY, at its discretion, offer some or all of its possible physical interfaces, offer logical interfaces (i.e. routed interfaces on a port or VLAN, or subscriber interfaces), or LAG interfaces. LAGs may be especially tricky, as the member ports of the LAG may span line cards of different capturing capabilities. Replicators SHOULD make an attempt to offer LAGs if all ports are of identical capability, and MAY offer them in the case where they are not, with a lowest-common capability set. Clients SHOULD NOT expect LAG functionality to be present, and SHOULD be prepared to set up separate sessions on each of the individual member ports if the Replicator does not offer the LAG, or offers it with an insufficient set.

3.3. Replicator offers Proposals

Upon receiving the 'status' change to 'client-request-complete', the Replicator updates the 'proposals' branch. This branch details zero, one, or more ways the Replicator can fulfill the sampling request. While generally there will only be zero or one proposals, a Replicator MAY offer more. For example, matching a sampling rate exactly would result in performance loss but a 'close enough' option can be offered that does not, or offers of what headers can be captured in the resulting stream. Each proposal includes a unique ID number, allowing the Client to select one, as detailed below.

If the Replicator is unable to provide any Proposals, the 'proposals' list MUST be empty, a human-readable error message MAY be returned in the 'proposal-error' field, then the 'status' field MUST be set to 'replicator-proposal-error'.

If the Replicator was able to provide Proposals, it MUST set the 'status' field to 'replicator-proposals-available' when it is

finished, and the 'proposals' branch MUST be read-only until the Client finishes the Proposal selection step below.

Part of each Proposal is a 'stream-format' branch, which informs the Client of the packet format the Receiver will be receiving. This format completely defines the entirety of the resulting data flow format besides the outer UDP wrapper - there is no normative format. A couple non-normative examples of what may result are provided in Section 4.

To adequately addresses the use cases stated above, a Replicator SHOULD support as a minimum set of capabilities:

- * An action field that denotes a pass or drop (ideally with drop reason)
- * Capturing at least 128 octets of payload
- * The original frame length
- * Sampling rates up to 1:1 (i.e. every packet is replicated), and down to 1:20000 or smaller.
- * Having different sampling sessions having different sampling rates (to allow a "general" session to be watching a broad selection of traffic, and more specific sessions targeting exact flows or situations)
- * At least two sessions per physical interface
- * Filtering on ingress port
- * Filtering on action
- * Filtering on direction of traffic

3.4. Client selects a Proposal

Upon either a notification or detection that the 'status' field has been updated, the Client then may then set the 'proposal-selected' entry to the value of the desired ID offered in 'proposals', and then set 'status' to 'client-proposal selected'. At this point, the Replicator:

- * MAY remove unnecessary branches in the 'proposals' list, but MUST retain the selected one.

- * MUST either install the requested sampling stream if possible, then MUST set 'status' to 'replicator-install-success'. If it cannot, it MAY set 'install-error' to a human-readable error message and MUST set 'status' to 'replicator-install-error'.
- * If the Proposal selected includes any of the 'dropped-' action-types as a filter, or does not specify an action-type filter at all, a Replicator MUST install the requested sampling before any filtering actions occur to the stream, as the sampling session is explicitly interested in pre-drop traffic.
- * If the Proposal selected does not include any of the 'dropped-' action-types as a filter, a Replicator MUST install the requested sampling after any filtering actions occur to the stream, to ensure the sampling ratio remains correct.

3.5. Ending sampling and cleanup

When a Client is finished with a sampling session, it deletes its entry in the 'samplers' tree to terminate a sampling session. Otherwise, a Client MUST refresh its entry by setting 'client-heartbeat' to 0 at least every 3600 seconds. The 'client-heartbeat' is then incremented by the Replicator. If 'client-heartbeat' exceeds 3600, the Replicator SHOULD consider the sampling configuration and any associated sampling session no longer necessary, terminate the sampling, and delete the entry. A Replicator MAY allow configuration to increase this timeout.

4. Data Stream Format

After the stream setup has been completed, the Receiver MUST use the stream-format data that the Replicator has calculated in its proposal. The Client and Receiver MUST NOT assume that the stream-format data is consistent between one stream setup and any other (there may be different versions of ASICs, different capabilities, different versions of operating systems, or different filters may yield a different format), or that the payload is always at the end (it could appear at the beginning or in the middle, and sufficient data is provided by the other fields to extract the data correctly). The stream-format data provides the Client with what information is provided at what location in the resulting packet. The Replicator MUST follow the expectation that is provided in these fields.

There is one captured packet per encapsulated packet, and thus the outer encapsulation length can be used to deduce the length of one variable-length field (designated by a field length of 0) contained within. If there is more than one variable-length field, a matching "-size"; field type MUST be provided for all but one of the variable-

length fields (as a single variable length can be deduced from the wrapper length).

This means there is no normative packet format or data layout - a large point of this specification is to allow that packet format to be negotiated and decided between the Client and Replicator, with the information passed back via the stream-format data.

One example of what the resulting packet may look like (but not a normative listing of what it is - the actual format can be any combination of fields, of any size, in any order), the data inside the resulting data stream after the UDP tunnel header may look like the following:

Example 1: Packet layout

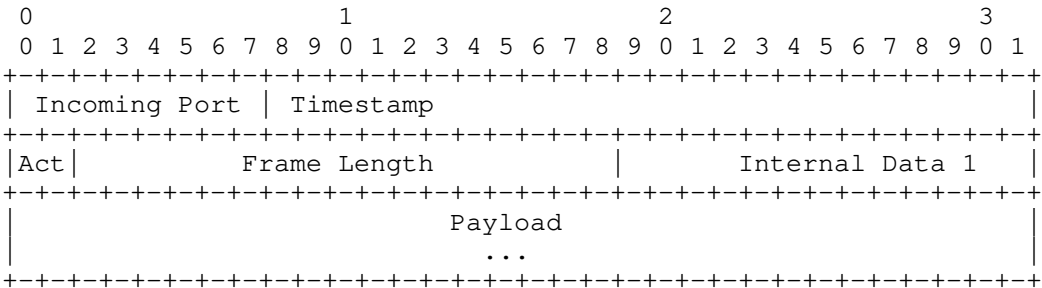


Figure 2

This non-normative example may be associated with a stream-format as per the following table:

Field Name	Field Size	Field Type	Field Type-Data
Incoming port	8	port-ingress	A listing of values that may be seen in this field, mapped to interface-refs from [RFC8343].
Timestamp	24	timestamp-nsec-ingress	Two 32-bit numbers giving when the "0" of this field

			is based off of, using the PTP Truncated Timestamp format.
Act	2	action	A listing of values that may be seen in this field, mapped to action types (accepted, dropped, etc.)
Frame Length	17	frame-length-ingress	Note that this denotes the original frame length - the payload field MAY not include the entire payload.
Internal Data 1	13	padding	Note that this may be ASIC-internal-only data, or some other information that would be expensive to prune out. 'padding' fields MUST have all content ignored.
Payload	0	frame-payload-ingress	

Table 1: Example 1: Stream-format data

Another non-normative example, which is similar to the [I-D.tuexen-opsawg-pcapng] enhanced packet block (EPB) format (and thus, this Replicator may in fact be a server offering a tcpdump-based backend using this frontend):

Field Name	Field Size	Field Type	Field Type-Data
Interface ID	32	port	A listing of values that may be seen in this field, mapped to interface-refs from [RFC8343].
Timestamp	64	timestamp-msec	Two 32-bit numbers giving when the "0" of this field is based off of, using the PTP Truncated Timestamp format.
Captured Packet Length	32	frame-payload-size	Note: This allows us to have the Options field as our real variable length field.
Original Packet Length	32	frame-length	
Packet Data	0	frame-payload	
Options	0	padding	

Table 2: Packet-format response example 2

To restate the prior note, the above is purely an example of what the format could be - the actual format used is negotiated between the Client and Replicator, and can have practically any layout, with any additional fields.

A Client SHOULD take efforts to be notified when a change has occurred on the Replicator (e.g., port or line card changes, device reboot, etc.), and re-verify and re-apply as needed its sampled streaming configurations when such a change is detected.

5. IANA Considerations

This document defines a new UDP port number, entitled "Sampled Streaming", and assigns a value of TBD1 from the Service Name and Transport Protocol Port Number Registry <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>:

Tag	Description
TBD1	Sampled Streaming

Table 3

This document requests registration of a URI in the "IETF XML Registry" RFC 3688 [RFC3688]. Following the format in RFC 3688, the following registration is suggested:

URI: urn:ietf:params:xml:ns:yang:ietf-sampled-streaming

Registrant Contact: The IESG.

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

This document registers a YANG module in the "YANG Module Names" registry RFC 6020 [RFC6020]:

name: ietf-sampled-streaming

namespace: urn:ietf:params:xml:ns:yang:ietf-sampled-streaming

prefix: ss

reference: This document

6. Security Considerations

Vendors and deployments must take into consideration that this functionality allows a mirroring of traffic, with configurable destinations and filters. Similar functionality already exists in various remote packet mirroring systems, and similar considerations should be taken. Filters utilizing the source port of TBD1 SHOULD be applied at the edges of a provider's network to provide an additional layer of security.

A Replicator SHOULD ensure that Clients can only see their own entries in the 'samplers', and MUST ensure that once a Client has created an entry in the samplers list, only that same Client may re-query or make changes to it.

7. Acknowledgments

The authors would like to thank Joe Clarke, Marek Hajduczenia, Brian Harber, Paolo Lucente, Jim Rampley, and Dmytro Shytiy for their reviews and providing helpful suggestions and feedback of this draft.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5476] Claise, B., Ed., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, DOI 10.17487/RFC5476, March 2009, <<https://www.rfc-editor.org/info/rfc5476>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols", RFC 6728, DOI 10.17487/RFC6728, October 2012, <<https://www.rfc-editor.org/info/rfc6728>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

8.2. Informative References

- [I-D.tuexen-opsawg-pcapng] Tuexen, M., Risso, F., Bongertz, J., Combs, G., Harris, G., and M. Richardson, "PCAP Next Generation (pcapng) Capture File Format", Work in Progress, Internet-Draft, draft-tuexen-opsawg-pcapng-01, 27 March 2020, <<http://www.ietf.org/internet-drafts/draft-tuexen-opsawg-pcapng-01.txt>>.

Appendix A. Yang Model Tree Reference

```

module: ietf-sampled-streaming
  +--rw points* [name]
    +--rw name                               psamp:nameType
    +--rw observationPoints* [name]
      +--rw name                             psamp:nameType
      +--ro observationPointId?              uint32
      +--rw observationDomainId              uint32
      +--rw ifName*                          ifNameType
      +--rw ifIndex*                         uint32
      +--rw entPhysicalName*                 string
      +--rw entPhysicalIndex*                uint32
      +--rw direction?                      direction
    +--rw selectors* [name]
      +--rw name                             psamp:nameType
      +--rw (Method)
        +--:(selectAll)
          +--rw selectAll?                   empty
        +--:(sampCountBased)
          +--rw sampCountBased {psampSampCountBased}?
            +--rw packetInterval             uint32
            +--rw packetSpace                uint32
        +--:(sampTimeBased)
          +--rw sampTimeBased {psampSampTimeBased}?
            +--rw timeInterval              uint32
            +--rw timeSpace                 uint32
        +--:(sampRandOutOfN)
          +--rw sampRandOutOfN {psampSampRandOutOfN}?
            +--rw size                      uint32
            +--rw population                uint32
        +--:(sampUniProb)
          +--rw sampUniProb {psampSampUniProb}?
            +--rw probability               decimal64
        +--:(filterMatch)
          +--rw filterMatch {psampFilterMatch}?
            +--rw (nameOrId)
              +--:(ieName)
                +--rw ieName?              ieNameType
              +--:(ieId)
                +--rw ieId?                 ieIdType
            +--rw ieEnterpriseNumber?        uint32
            +--rw value                      string
        +--:(filterHash)
          +--rw filterHash {psampFilterHash}?
            +--rw hashFunction?              identityref
            +--rw initializerValue?          uint64
            +--rw ipPayloadOffset?           uint64

```

```

    +--rw ipPayloadSize?      uint64
    +--rw digestOutput?       boolean
    +--ro outputRangeMin?     uint64
    +--ro outputRangeMax?     uint64
    +--rw selectedRange* [name]
        +--rw name           nameType
        +--rw min?           uint64
        +--rw max?           uint64
    +--ro packetsObserved?     yang:counter64
    +--ro packetsDropped?     yang:counter64
    +--ro selectorDiscontinuityTime? yang:date-and-time
+--ro filters* []
| +--ro filter           filter-type
+--ro max-samplers?      uint32
+--ro max-filters?       uint32
+--ro max-frame-length-copy? uint16
+--ro max-frame-depth-inspect? uint16
+--rw samplers* [name]
    +--rw name            string
    +--rw status           status-type
    +--rw client-heartbeat uint32
    +--rw destination
        +--rw type           destination-type
        +--rw udp-parameters
            +--rw destination-ip      inet:ip-address-no-zone
            +--rw destination-port    inet:port-number
+--rw request
    +--rw filters
        +--rw name?           string
        +--rw interfaces* [int]
            | +--rw int        if:interface-ref
        +--rw actions* [action]
            | +--rw action     action-type
        +--rw direction?      psamp:direction
        +--rw type            filter-type
        +--rw ipv4-address?    inet:ipv4-address-no-zone
        +--rw ipv6-address?    inet:ipv6-address-no-zone
        +--rw version?         inet:ip-version
        +--rw frame-payload
            +--rw offset?      uint16
            +--rw match?       binary
        +--rw frame-length?    uint16
    +--rw selector
        +--rw (Method)
            | +--:(selectAll)
            | | +--rw selectAll?           empty
            | +--:(sampCountBased)
            | | +--rw sampCountBased {psampSampCountBased}?

```

```

+---rw packetInterval    uint32
+---rw packetSpace       uint32
+---:(sampTimeBased)
+---rw sampTimeBased {psampSampTimeBased}?
+---rw timeInterval      uint32
+---rw timeSpace         uint32
+---:(sampRandOutOfN)
+---rw sampRandOutOfN {psampSampRandOutOfN}?
+---rw size              uint32
+---rw population        uint32
+---:(sampUniProb)
+---rw sampUniProb {psampSampUniProb}?
+---rw probability       decimal64
+---:(filterMatch)
+---rw filterMatch {psampFilterMatch}?
+---rw (nameOrId)
+---:(ieName)
+---rw ieName?           ieNameType
+---:(ieId)
+---rw ieId?             ieIdType
+---rw ieEnterpriseNumber? uint32
+---rw value             string
+---:(filterHash)
+---rw filterHash {psampFilterHash}?
+---rw hashFunction?     identityref
+---rw initializerValue? uint64
+---rw ipPayloadOffset?  uint64
+---rw ipPayloadSize?    uint64
+---rw digestOutput?     boolean
+---ro outputRangeMin?   uint64
+---ro outputRangeMax?   uint64
+---rw selectedRange* [name]
+---rw name              nameType
+---rw min?              uint64
+---rw max?              uint64
+---ro packetsObserved?  yang:counter64
+---ro packetsDropped?   yang:counter64
+---ro selectorDiscontinuityTime? yang:date-and-time
+---rw ratio            uint32
+---rw min-ratio?       uint32
+---rw max-ratio?       uint32
+---ro proposals* [id]
+---ro id                uint32
+---ro selector
+---ro (Method)
+---:(selectAll)
+---ro selectAll?        empty
+---:(sampCountBased)

```

```

+--ro sampCountBased {psampSampCountBased}?
+--ro packetInterval    uint32
+--ro packetSpace       uint32
+--:(sampTimeBased)
+--ro sampTimeBased {psampSampTimeBased}?
+--ro timeInterval      uint32
+--ro timeSpace         uint32
+--:(sampRandOutOfN)
+--ro sampRandOutOfN {psampSampRandOutOfN}?
+--ro size               uint32
+--ro population         uint32
+--:(sampUniProb)
+--ro sampUniProb {psampSampUniProb}?
+--ro probability        decimal64
+--:(filterMatch)
+--ro filterMatch {psampFilterMatch}?
+--ro (nameOrId)
+--:(ieName)
+--ro ieName?            ieNameType
+--:(ieId)
+--ro ieId?              ieIdType
+--ro ieEnterpriseNumber? uint32
+--ro value              string
+--:(filterHash)
+--ro filterHash {psampFilterHash}?
+--ro hashFunction?      identityref
+--ro initializerValue?  uint64
+--ro ipPayloadOffset?   uint64
+--ro ipPayloadSize?     uint64
+--ro digestOutput?      boolean
+--ro outputRangeMin?    uint64
+--ro outputRangeMax?    uint64
+--ro selectedRange* [name]
+--ro name               nameType
+--ro min?               uint64
+--ro max?               uint64
+--ro packetsObserved?    yang:counter64
+--ro packetsDropped?     yang:counter64
+--ro selectorDiscontinuityTime? yang:date-and-time
+--ro performance-penalty? boolean
+--ro performance-penalty-amount? uint16
+--ro stream-format
+--ro fields* [name]
+--ro name               string
+--ro size?              uint32
+--ro type?              field-type
+--ro action-mappings* [value]
+--ro value              binary

```

```

| | | | | |--ro meaning?      action-type
| | | | | +--ro port-mappings* [value]
| | | | | |   |--ro value      binary
| | | | | |   |--ro port?     if:interface-ref
| | | | | +--ro direction-mappings* [value]
| | | | | |   |--ro value      binary
| | | | | |   |--ro direction? psamp:direction
| | | | | +--ro timestamp
| | | | | |   |--ro seconds?    uint32
| | | | | |   |--ro nanoseconds? uint32
| | | | | +--ro payload-contents? frame-headers
+--ro filters* [name]
|   |--ro name          string
|   |--ro interfaces* [int]
|   |   |--ro int       if:interface-ref
|   +--ro actions* [action]
|   |   |--ro action    action-type
|   +--ro direction?    psamp:direction
+--ro type              filter-type
+--ro ipv4-address?     inet:ipv4-address-no-zone
+--ro ipv6-address?     inet:ipv6-address-no-zone
+--ro version?          inet:ip-version
+--ro frame-payload
|   |--ro offset?      uint16
|   |--ro match?       binary
+--ro frame-length?     uint16
+--rw proposal-error?   string
+--rw proposal-selected? uint32
+--rw install-error?    string

```

```

module ietf-sampled-streaming {
  namespace "urn:ietf:params:xml:ns:yang:ietf-sampled-streaming";
  prefix ss;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-ipfix-psamp {
    prefix psamp;
    revision-date 2012-09-05;
  }

  organization

```

```
"IETF Working Group";
contact
  "Editor:      Andrew Gray
    <mailto:Andrew.Gray@charter.com>";
description
  "This module contains a collection of YANG definitions for
    managing sampled streaming subscriptions.

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

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

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

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

revision 2019-12-27 {
  description
    "Clarifications based on feedback for -03 draft.  Utilize parts
    of RFC 6728 to avoid redundancy, where possible.";
  reference
    "draft-gray-sampled-streaming-03";
}
revision 2019-10-22 {
  description
    "Updates based on feedback for -02 draft: Adding more forwarded
    action-types.  frame-payload changed to be explicit about
    direction.  Added -size types explicitly for frame-payload and
    padding to allow using more than one zero-length field.";
  reference
    "draft-gray-sampled-streaming-02";
}
revision 2019-08-06 {
  description
    "Updates based on feedback for -01 draft.";
  reference
```



```
    "draft-gray-sampled-streaming-01";
}
revision 2019-06-25 {
  description
    "Initial version.";
  reference
    "draft-gray-sampled-streaming-00";
}

typedef filter-type {
  type enumeration {
    enum interfaces {
      description
        "List of interfaces to filter against.";
    }
    enum action {
      description
        "Filter against a list of actions that the Point took (i.e.
        only consider packets that were actually forwarded).";
    }
    enum direction {
      description
        "Direction to sample traffic in.";
    }
    enum ip-version {
      description
        "The version number in the IP header.";
    }
    enum ip-v4-srcip {
      description
        "The IPv4 header's source IPv4 address.";
    }
    enum ip-v4-dstip {
      description
        "The IPv4 header's destination IPv4 address.";
    }
    enum ip-v4-ttl {
      description
        "The IPv4 header's Time to Live.";
    }
    enum ip-v4-prot {
      description
        "The IPv4 header's protocol number.";
    }
    enum ip-v6-srcip {
      description
        "The IPv6 header's source IPv4 address.";
    }
  }
}
```

```
enum ip-v6-dstip {
    description
        "The IPv6 header's destination IPv4 address.";
}
enum frame-size {
    description
        "The total size of the frame.";
}
enum frame-payload {
    description
        "Specific payload octets.";
}
enum frame-length {
    description
        "Specific frame length.";
}
}
description
    "The filtering abilities available.";
}

typedef field-type {
    type enumeration {
        enum padding {
            description
                "Padding bits that MUST be ignored.";
        }
        enum padding-size {
            description
                "This packet's length of a variable-length padding field.";
        }
        enum port {
            description
                "An indication of the port the traffic was sampled from.";
        }
        enum direction {
            description
                "Which direction the traffic went.";
        }
        enum port-ingress {
            description
                "What port the traffic was received from (may be different
                than 'port')";
        }
        enum port-egress {
            description
                "What port the traffic is leaving on (may be different than
                'port')";
        }
    }
}
```

```
}
enum timestamp-msec-ingress {
  description
    "The timestamp the packet was received at, in integer
    milliseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum timestamp-usec-ingress {
  description
    "The timestamp the packet was received at, in integer
    microseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum timestamp-nsec-ingress {
  description
    "The timestamp the packet was received at, in integer
    nanoseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum timestamp-msec-egress {
  description
    "The timestamp the packet left the point at, in integer
    milliseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum timestamp-usec-egress {
  description
    "The timestamp the packet left the point at, in integer
    microseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum timestamp-nsec-egress {
  description
    "The timestamp the packet left the point at, in integer
    nanoseconds. The epoch of this number is provided in the
    timestamp container of the returned field information.";
}
enum frame-length {
  description
    "The generic frame length. Note that due to chipset
    capabilities, this MAY not be the same as the captured
    packet length.";
}
enum frame-length-ingress {
  description
    "The frame length as received by the point. Note that due
    to chipset capabilities, this MAY not be the same as the
    captured packet length.";
```

```
    }
    enum frame-length-egress {
        description
            "The frame length after local processing, as it leaves the
            point. Note that due to chipset capabilities, this MAY
            not be the same as the captured packet length.";
    }
    enum frame-payload-size {
        description
            "The length of the payload that has actually been copied
            into this stream.";
    }
    enum frame-payload-ingress {
        description
            "The payload of the frame, as received the point.";
    }
    enum frame-payload-egress {
        description
            "The payload of the frame, as it leaves the point.";
    }
    enum action {
        description
            "The action that was taken on this frame. Values are
            mapped as according to action-type.";
    }
}
description
    "Types of data included in the data stream provided back to
    the receiver. Note that all fields MAY not be provided.";
}

typedef action-type {
    type enumeration {
        enum forwarded {
            description
                "Generically forwarded normally through the system. A more
                specific action type code SHOULD be used.";
        }
        enum forwarded-label-change {
            description
                "Forwarded, with a generic MPLS label change having
                occurred.";
        }
        enum forwarded-label-swap {
            description
                "Forwarded, with a MPLS label swap.";
        }
        enum forwarded-label-pop {
```

```
    description
        "Forwarded, with a MPLS label pop.";
}
enum forwarded-label-push {
    description
        "Forwarded, with a MPLS label push.";
}
enum forwarded-cpu-punt {
    description
        "Forwarded after a CPU punt.";
}
enum forwarded-tunnel {
    description
        "Forwarded with additional outer wrapper for tunneling.";
}
enum forwarded-tunnel-frr {
    description
        "Forwarded with additional outer wrapper due to fast
        reroute.";
}
enum dropped {
    description
        "Generically dropped.  A more specific action type code
        SHOULD be used.";
}
enum dropped-rate-limit {
    description
        "Dropped due to a rate limiter applied.";
}
enum dropped-buffer {
    description
        "Dropped due to no buffer space.";
}
enum dropped-security {
    description
        "Dropped due to a security policy.";
}
enum dropped-error {
    description
        "Dropped due to the frame being in error.";
}
enum dropped-cpu-punt {
    description
        "Dropped after a CPU punt.";
}
enum passed-to-cpu {
    description
        "Passed on to the CPU, but what the CPU did with it is
```

```
        unknown.";
    }
}
description
    "Possible actions taken on a packet.";
}

typedef destination-type {
    type enumeration {
        enum udp {
            description
                "Sent with a UDP header.";
        }
    }
    description
        "Different possible destination types.";
}

typedef status-type {
    type enumeration {
        enum client-request-complete {
            description
                "The Client has completed its request setup.";
        }
        enum replicator-proposals-available {
            description
                "The Replicator has finished processing the request, and
                has proposals available in the 'proposals' branch.";
        }
        enum replicator-proposal-error {
            description
                "The Replicator encountered an error attempting to come up
                with a proposal. 'proposal-error' MAY contain an
                explanation.";
        }
        enum client-proposal-selected {
            description
                "The Client has updated 'proposal-selected' and is ready
                for the Replicator to install the requested sampling.";
        }
        enum replicator-install-success {
            description
                "The Replicator has successfully activated the sampling,
                and it is operating.";
        }
        enum replicator-install-error {
            description
                "The Replicator encountered an error installing the
```

```
        sampling. 'install-error' MAY contain an explanation.";
    }
}
description
    "The status of a sampler entry.";
}

typedef frame-headers {
    type bits {
        bit eth-l1-preamble {
            position 0;
            description
                "Will include the Ethernet preamble.";
        }
        bit eth-l1-sof {
            position 1;
            description
                "Will include the Ethernet start of frame
                delimiter";
        }
        bit eth-l2-dmac {
            position 2;
            description
                "Will include the outer Ethernet destination MAC.";
        }
        bit eth-l2-smac {
            position 3;
            description
                "Will include the outer Ethernet source MAC.";
        }
        bit eth-l2-vlan {
            position 4;
            description
                "Will include any 802.1Q-2018 VLAN tags.";
        }
        bit eth-l2-type {
            position 5;
            description
                "Will include the Ethertype or size.";
        }
        bit eth-l2-fcs {
            position 6;
            description
                "Will include the Frame Check Sequence after the
                payload.";
        }
        bit eth-l1-ipg {
            position 7;
```

```
        description
        "Will include the inter-packet gap.  Be aware that
        different Ethernet speeds may have different lengths.";
    }
    bit mpls-tags {
        position 8;
        description
        "Will include MPLS tags.";
    }
}
description
"Listing of fields to be provided in a frame capture.";
}

grouping filters {
    description
    "Filter definition.  Multiple filters are ANDed.";
    leaf name {
        type string {
            length "1..255";
        }
        description
        "A name for this filter.";
    }
    list interfaces {
        when "../type = 'interfaces'";
        key "int";
        description
        "Filter down to only this list of interfaces.";
        leaf int {
            type if:interface-ref;
            description
            "A specific interface to filter against.";
        }
    }
    list actions {
        when "../type = 'action'";
        key "action";
        description
        "Filter down to only this list of actions.";
        leaf action {
            type action-type;
            description
            "One specific action code.";
        }
    }
}
leaf direction {
    when "../type = 'direction'";
```



```
    type psamp:direction;
    description
        "Which direction(s) to sample traffic in.";
}
leaf type {
    type filter-type;
    mandatory true;
    description
        "The type of filter associated.";
}
leaf ipv4-address {
    when "../type = 'ip-v4-srcip' | ../type = 'ip-v4-dstip'";
    type inet:ipv4-address-no-zone;
    description
        "The IPv4 address to filter on.";
}
leaf ipv6-address {
    when "../type = 'ip-v6-srcip' | ../type = 'ip-v6-dstip'";
    type inet:ipv6-address-no-zone;
    description
        "The IPv6 address to filter on.";
}
leaf version {
    when "../type = 'ip-version'";
    type inet:ip-version;
    description
        "The value of the IP version number to match on.";
}
container frame-payload {
    when "../type = 'frame-payload'";
    description
        "Frame payload fragment to match on.";
    leaf offset {
        type uint16;
        description
            "Offset in octets from the start of the frame to begin the
            match on.";
    }
    leaf match {
        type binary;
        description
            "The bytes to match on.";
    }
}
leaf frame-length {
    when "../type = 'frame-length'";
    type uint16;
    description
```

```
        "Frame length to match on.";
    }
}

grouping stream-format {
    description
        "This contains the packet format data that this sampling stream
        is sending. This is only valid after configuration. The
        length fields are given in bits, and are consecutive. Needed
        gaps should use a 'padding' element.";
    list fields {
        key "name";
        description
            "The listing of the fields that will be encapsulated and sent
            to the receiver.";
        leaf name {
            type string {
                length "1..255";
            }
            description
                "Human readable name of what this field contains.";
        }
        leaf size {
            type uint32 {
                range "0..524280";
            }
            description
                "The size of this field, in bits. The value of '0' denotes
                a variable-sized field.";
        }
        leaf type {
            type field-type;
            description
                "The type of this data.";
        }
        list action-mappings {
            when "../type='action'";
            key "value";
            description
                "The mapping of values to action-type codes, valid for
                type=action.";
            leaf value {
                type binary;
                description
                    "The value that will appear in the header.";
            }
            leaf meaning {
                type action-type;
            }
        }
    }
}
```

```
        description
            "What this value indicates.";
    }
}
list port-mappings {
    when "../type='ingress-port' | ../type='egress-port'";
    key "value";
    description
        "The mapping of values to interfaces, valid for
        type=ingress-port or type=egress-port";
    leaf value {
        type binary;
        description
            "The value that will appear in the header.";
    }
    leaf port {
        type if:interface-ref;
        description
            "The port the value maps to.";
    }
}
list direction-mappings {
    when "../type='direction'";
    key "value";
    description
        "The mapping of values to direction codes, valid for
        type=direction.";
    leaf value {
        type binary;
        description
            "The value that will appear in the header.";
    }
    leaf direction {
        type psamp:direction;
        description
            "The direction the traffic in respect to the port. The
            value 'both' MUST NOT be used here.";
    }
}
container timestamp {
    when "../type='timestamp-nsec' | ../type='timestamp-usec' |
        ../type='timestamp-msec'";
    description
        "Supplemental data for type=timestamp*, in PTP Truncated
        Timestamp Format. Provides the time used as the epoch for
        the number in the data stream.";
    leaf seconds {
        type uint32;
    }
}
```

```
        description
            "Specifies the integer portion of the number of seconds
             since the epoch.";
    }
    leaf nanoseconds {
        type uint32;
        description
            "Specifies the fractional portion of the number of
             seconds since the epoch, in integer number of
             nanoseconds.";
    }
}
leaf payload-contents {
    when "../type='frame-payload-ingress' |
        ../type='frame-payload-egress'";
    type frame-headers;
    description
        "Details about what parts of the frame this payload field
         SHOULD contain. Note carefully the 'SHOULD' - for a
         variety of reasons (different forwarding paths, exception
         handling, etc.), the actual headers of any one frame MAY
         be different than this.";
}
}
}

list points {
    key "name";
    description
        "A listing of the observation points available on this device, what
         ports they provide for, and what filtering is available at
         those points.";
    leaf name {
        type psamp:nameType;
        description
            "A unique name for this point.";
    }
    list observationPoints {
        key "name";
        description
            "A list of the observation points (i.e. interfaces) able to be
             monitored at this point.";
        leaf name {
            type psamp:nameType;
            description
                "Name of this observationPoint";
        }
    }
    uses psamp:observationPointParameters;
}
```

```
}
list selectors {
  key "name";
  description
    "List of packet selector options available at this point.";
  leaf name {
    type psamp:nameType;
    description
      "A unique name for this selector option.";
  }
  uses psamp:selectorParameters;
}
list filters {
  config false;
  description
    "List of filtering options available at this point.";
  leaf filter {
    type filter-type;
    mandatory true;
    description
      "One specific filter available at this point.";
  }
}
leaf max-samplers {
  type uint32;
  config false;
  description
    "The maximum number of additional samplers that can be
    installed at this point.";
}
leaf max-filters {
  type uint32;
  config false;
  description
    "The maximum number of filtering rules permitted at this
    location. Note this is an absolute maximum, and fewer rules
    that are complex may still be rejected by the device.";
}
leaf max-frame-length-copy {
  type uint16;
  config false;
  description
    "The maximum size that the point can replicate and copy into
    the header.";
}
leaf max-frame-depth-inspect {
  type uint16;
  config false;
```

```
    description
      "The offset of the last octet in a frame the point can
       perform filtering against.";
  }
  list samplers {
    key "name";
    description
      "A list of all the samplers attached to this point.";
    leaf name {
      type string;
      mandatory true;
      description
        "A unique name given to this sampler.";
    }
    leaf status {
      type status-type;
      mandatory true;
      description
        "The current status of this sampler.";
    }
    leaf client-heartbeat {
      type uint32;
      mandatory true;
      description
        "The number of seconds since the Client has refreshed this
         request. The Client MUST only be able to set this value
         to 0, the Replicator MUST keep track of it, and SHOULD
         delete this entry when it reaches 3600.";
    }
  }
  container destination {
    description
      "The destination of where to send the UDP stream to.";
    leaf type {
      type destination-type;
      mandatory true;
      description
        "The type of encoding for the destination.";
    }
  }
  container udp-parameters {
    when "../type='udp'";
    description
      "Parameters for destination-type=udp. Source port is
       always the port number assigned by IANA.";
    leaf destination-ip {
      type inet:ip-address-no-zone;
      mandatory true;
      description
        "The destination IP to send the stream to.";
```

```
    }
    leaf destination-port {
      type inet:port-number;
      mandatory true;
      description
        "The destination UDP port number to send the stream
        to.";
    }
  }
}
container request {
  description
    "The request as sent in by a Client.";
  container filters {
    description
      "Requested filters to apply to the stream.";
    uses filters;
  }
  container selector {
    description
      "Requested packet Selector.";
    uses psamp:selectorParameters;
  }
  leaf ratio {
    type uint32 {
      range "1..max";
    }
    mandatory true;
    description
      "The requested sampling ratio (1:N, with N being this
      value).";
  }
  leaf min-ratio {
    type uint32 {
      range "1..max";
    }
    description
      "The minimum value of N the client will accept.";
  }
  leaf max-ratio {
    type uint32 {
      range "1..max";
    }
    description
      "The maximum value of N the client will accept.";
  }
}
list proposals {
```

```
key "id";
config false;
description
  "The proposals as offered by the Replicator.";
leaf id {
  type uint32 {
    range "1..max";
  }
  description
    "An id-number representing this proposal for selection.";
}
container selector {
  description
    "Provided packet Selector, plus stores statistics when
    this proposal is active.";
  uses psamp:selectorParameters;
}
leaf performance-penalty {
  type boolean;
  description
    "Selecting this offer will result in a forwarding performance
    penalty on the device (usually due to ASIC recirculation)";
}
leaf performance-penalty-amount {
  type uint16 {
    range "0..10000";
  }
  description
    "The forwarding performance penalty amount, in hundredths
    of a percent. This value is not required even if
    performance-penalty is true. If present, it MUST be
    treated as an estimate.";
}
container stream-format {
  description
    "The stream format that would be generated if this
    proposal is selected.";
  uses stream-format;
}
list filters {
  key "name";
  description
    "The filters the Replicator can actually apply in this
    proposal. These MAY not match the request.";
  uses filters;
}
}
leaf proposal-error {
```



```
        type string {
            length "1..1023";
        }
        description
            "The Replicator was unable to generate any Proposals.";
    }
    leaf proposal-selected {
        type uint32 {
            range "1..max";
        }
        description
            "The ID of the proposal above the Client wants
            to use.";
    }
    leaf install-error {
        type string {
            length "1..1023";
        }
        description
            "The Replicator was unable to install the requested
            Proposal for this reason.";
    }
}
}
```

Authors' Addresses

Andrew Gray
Charter Communications
8560 Upland Drive, Suite B
Englewood, CO 80112
United States of America

Phone: +1 720 699 5125
Email: Andrew.Gray@charter.com

Lawrence J Wobker
Cisco Systems
170 W Tasman Drive
San Jose, CA 95134
United States of America

Phone: +1 984 216 1860
Email: lwobker@cisco.com

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

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

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

Abstract

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

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 11 April 2022.

Copyright Notice

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

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

Table of Contents

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

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

1. Introduction

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

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

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

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

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

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

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

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

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

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

2. Terminology

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

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

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

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

This document makes use of the following terms:

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

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

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

Network orchestrator: A functional entity that is hierarchically

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

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

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

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

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

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

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

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

3. Acronyms

The following acronyms are used in the document:

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

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

4. L3NM Reference Architecture

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

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

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

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

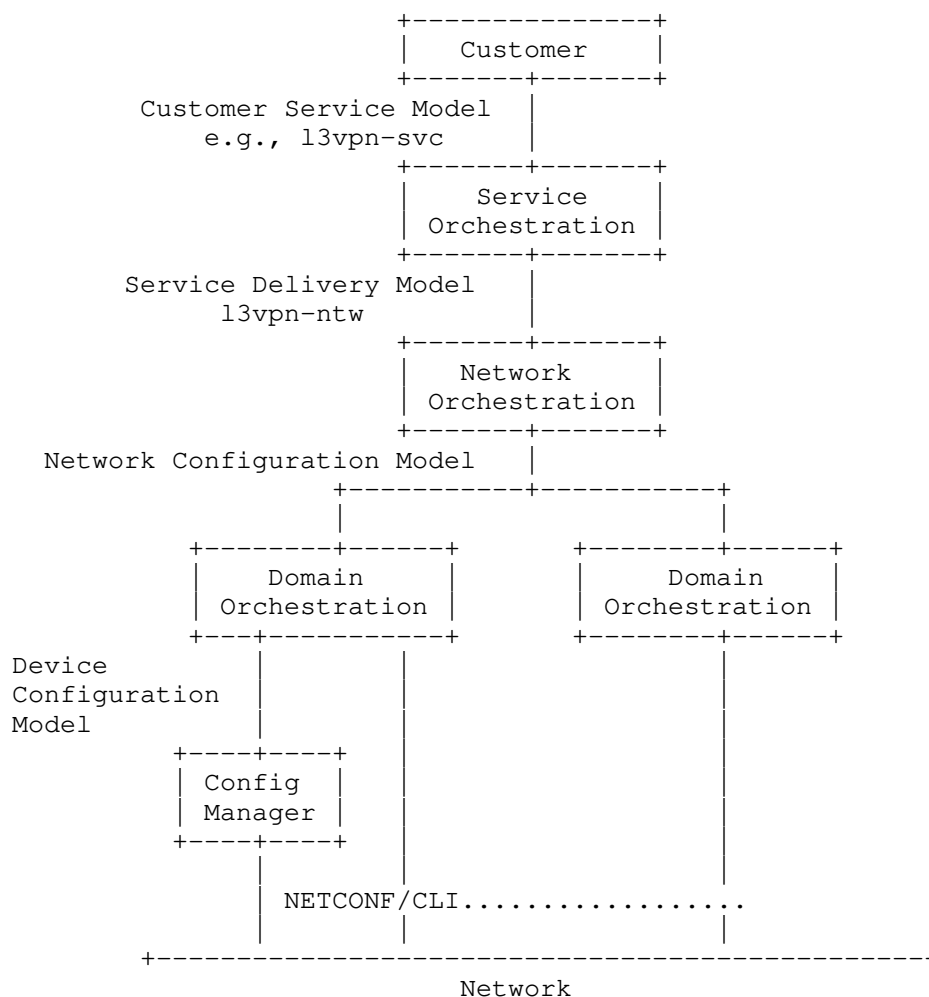


Figure 1: L3NM Reference Architecture

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

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

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

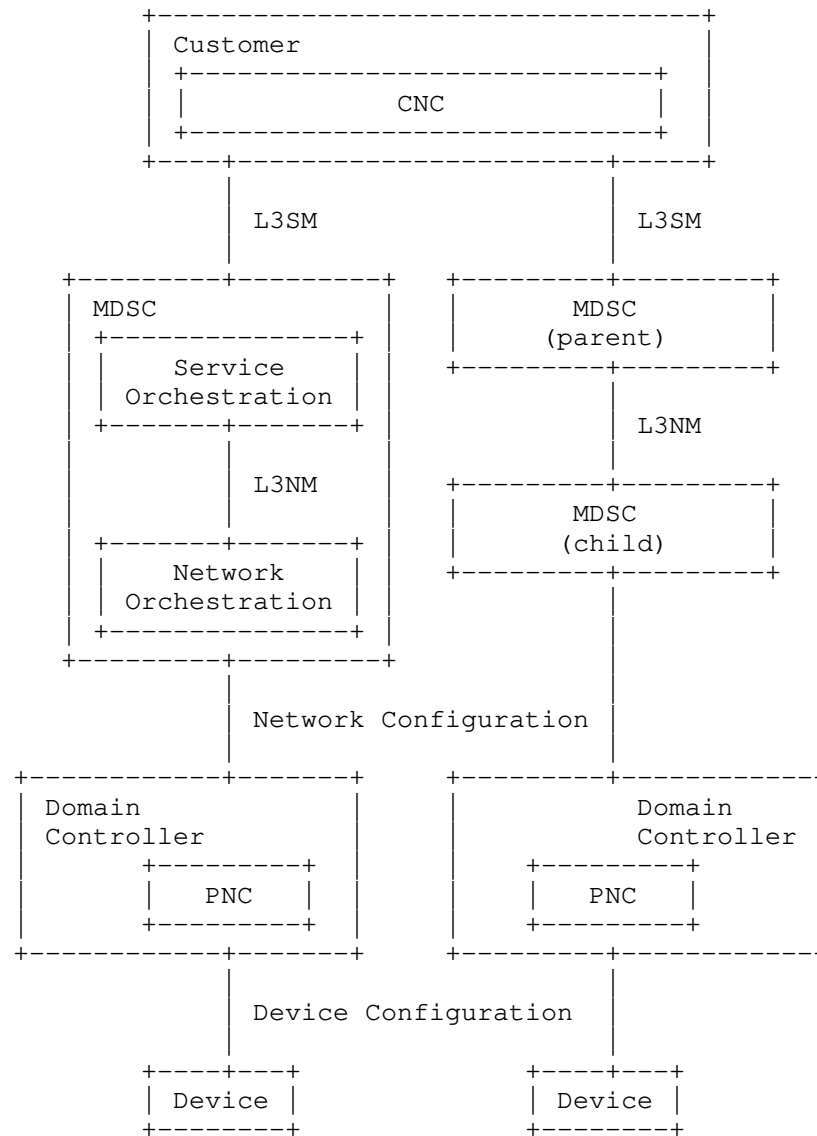


Figure 2: L3SM and L3NM in the Context of ACTN

5. Relation with other YANG Models

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

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

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

L3SM: L3NM is not a customer service model.

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

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

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

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

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

Device Modules: L3NM is not a device model.

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

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

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

6. Sample Uses of the L3NM Data Model

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

6.1. Enterprise Layer 3 VPN Services

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

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

6.2. Multi-Domain Resource Management

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

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

6.3. Management of Multicast Services

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

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

7. Description of the L3NM YANG Module

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

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

7.1. Overall Structure of the Module

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

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

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

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

Figure 3: Overall L3NM Tree Structure

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

7.2. VPN Profiles

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

```

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

```

Figure 4: VPN Profiles Subtree Structure

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

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

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

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

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

7.3. VPN Services

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

```

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

```

Figure 5: VPN Services Subtree Structure

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

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

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

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

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

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

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

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

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

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

More details are provided in Section 7.4.

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

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

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

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

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

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

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

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

More details are provided in Section 7.5.

7.4. VPN Instance Profiles

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

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

```

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

```

```

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

```

Figure 6: Subtree Structure of VPN Instance Profiles

The description of the listed data nodes is as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

7.5. VPN Nodes

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

```

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

```

```

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

```


Figure 7: VPN Node Subtree Structure

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See Section 7.6 for more details.

7.6. VPN Network Accesses

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

```

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

```

Figure 8: VPN Network Access Subtree Structure

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

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

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

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

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

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

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

7.6.1. Connection

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

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

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

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

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

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

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

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

```

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

```

Figure 9: Connection Subtree Structure

7.6.2. IP Connection

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

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

```

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

```

Figure 10: IP Connection Subtree Structure

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

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

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

```

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

```

Figure 11: IP Connection Subtree Structure (IPv4)

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


```

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

```

Figure 12: IP Connection Subtree Structure (IPv6)

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

```

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

```

Figure 13: IP Connection Subtree Structure (Static Mode)

7.6.3. CE-PE Routing Protocols

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

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

```

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

```

Figure 14: Routing Subtree Structure

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

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

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

* A combination thereof.

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

7.6.3.1. Static Routing

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

```

...
+--rw routing-protocols
+--rw routing-protocol* [id]
...
+--rw static
+--rw cascaded-lan-prefixes
+--rw ipv4-lan-prefixes*
|   [lan next-hop]
|   {vpn-common:ipv4}?
+--rw lan                inet:ipv4-prefix
+--rw lan-tag?           string
+--rw next-hop           union
+--rw bfd-enable?       boolean
+--rw metric?            uint32
+--rw preference?       uint32
+--rw status
|   +--rw admin-status
|   |   +--rw status?            identityref
|   |   +--rw last-change?      yang:date-and-time
|   +--ro oper-status
|   |   +--ro status?            identityref
|   |   +--ro last-change?      yang:date-and-time
+--rw ipv6-lan-prefixes*
|   [lan next-hop]
|   {vpn-common:ipv6}?
+--rw lan                inet:ipv6-prefix
+--rw lan-tag?           string
+--rw next-hop           union
+--rw bfd-enable?       boolean
+--rw metric?            uint32
+--rw preference?       uint32
+--rw status
|   +--rw admin-status
|   |   +--rw status?            identityref
|   |   +--rw last-change?      yang:date-and-time
|   +--ro oper-status
|   |   +--ro status?            identityref
|   |   +--ro last-change?      yang:date-and-time
...

```

Figure 15: Static Routing Subtree Structure

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.

```

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

```

```

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

```

Figure 16: BGP Routing Subtree Structure

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

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

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

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

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

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

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

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

7.6.3.3. OSPF

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

```

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

```

Figure 17: OPSF Routing Subtree Structure

The following data nodes are captured in Figure 17:

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

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

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

'metric': Associates a metric with OSPF routes.

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

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

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

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

7.6.3.4. IS-IS

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

```

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

```

Figure 18: IS-IS Routing Subtree Structure

The following IS-IS data nodes are supported:

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

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

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

7.6.3.5. RIP

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

```

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

```

Figure 19: RIP Subtree Structure

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

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

```

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

```

Figure 20: VRRP Subtree Structure

The following data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. Note that VRRP version 3 [RFC5798] supports both IPv4 and IPv6.

'vrrp-group': Is used to identify the VRRP group.

'backup-peer': Carries the IP address of the peer.

'virtual-ip-address': Includes virtual IP addresses for a single VRRP group.

'priority': Assigns the VRRP election priority for the backup virtual router.

'ping-reply': Controls whether ping requests can be replied to.

'status': Indicates the status of the VRRP instance.

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

7.6.4. OAM

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

```

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

```

Figure 21: IP Connection Subtree Structure (OAM)

The following OAM data nodes can be specified:

'session-type': Indicates which BFD flavor is used to set up the session (e.g., classic BFD [RFC5880], Seamless BFD [RFC7880]). By default, the BFD session is assumed to follow the behavior specified in [RFC5880].

'desired-min-tx-interval': Is the minimum interval, in microseconds, that a PE would like to use when transmitting BFD Control packets less any jitter applied.

'required-min-rx-interval': Is the minimum interval, in microseconds, between received BFD Control packets that a PE is capable of supporting, less any jitter applied by the sender.

'local-multiplier': The negotiated transmit interval, multiplied by this value, provides the detection time for the peer.

'holdtime': Is used to indicate the expected BFD holddown time, in

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

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

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

'status': Indicates the status of BFD.

7.6.5. Security

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

```

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

```

Figure 22: Security Subtree Structure

7.6.6. Services

7.6.6.1. Overview

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

```

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

```

Figure 23: Services Subtree Structure

The following data nodes are defined:

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

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

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

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

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

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

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

7.6.6.2. QoS

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

```

...
+--rw qos {vpn-common:qos}?
|   +--rw qos-classification-policy
|   |   +--rw rule* [id]
|   |   |   +--rw id                string
|   |   |   +--rw (match-type)?
|   |   |   |   +--:(match-flow)
|   |   |   |   |   +--rw (l3)?
|   |   |   |   |   |   +--:(ipv4)
|   |   |   |   |   |   ...
|   |   |   |   |   |   +--:(ipv6)
|   |   |   |   |   |   ...
|   |   |   |   |   +--rw (l4)?
|   |   |   |   |   |   +--:(tcp)
|   |   |   |   |   |   ...
|   |   |   |   |   |   +--:(udp)
|   |   |   |   |   |   ...
|   |   |   |   +--:(match-application)
|   |   |   |   |   +--rw match-application?
|   |   |   |   |   |   identityref
|   |   |   +--rw target-class-id?
|   |   |   |   string
|   +--rw qos-action
|   |   +--rw rule* [id]
|   |   |   +--rw id                string
|   |   |   +--rw target-class-id?  string
|   |   |   +--rw inbound-rate-limit? decimal64
|   |   |   +--rw outbound-rate-limit? decimal64
|   +--rw qos-profile
|   |   +--rw qos-profile* [profile]
|   |   |   +--rw profile            leafref
|   |   |   +--rw direction?        identityref
...

```

Figure 24: Services Subtree Structure

QoS classification can be based on many criteria such as:

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

```

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

```

Figure 25: QoS Subtree Structure (L3)

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

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

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

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

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

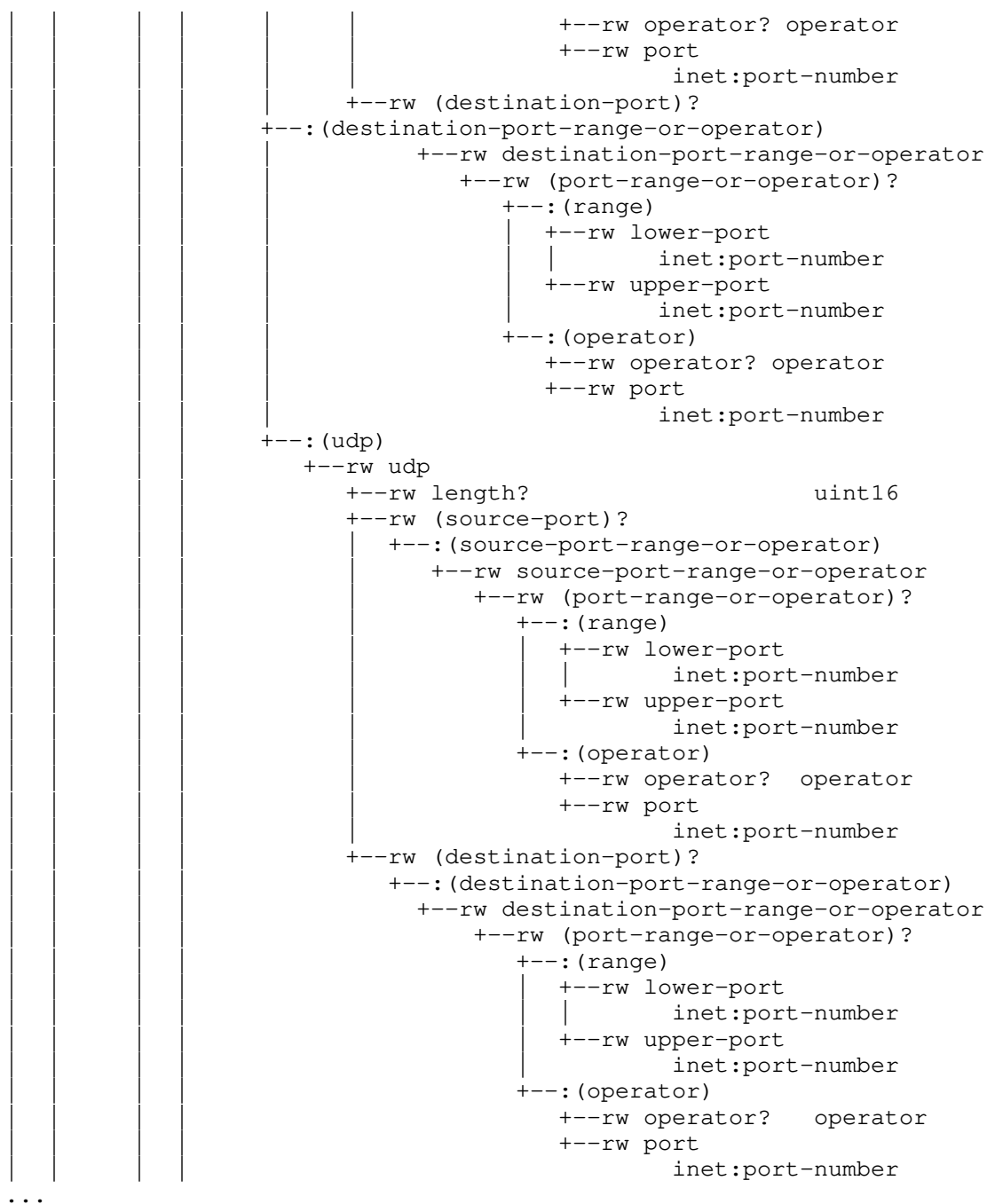


Figure 26: QoS Subtree Structure (L4)

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

```



```

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

```

...

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

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

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

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

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

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

```

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

```

Figure 29: Multicast Subtree Structure (VPN Node Level)

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

```

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

```

```

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

```

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

8. L3NM YANG Module

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

```

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

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

```

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

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

    Author:   Samier Barguil
              <mailto:samier.barguilgiraldo.ext@telefonica.com>
    Editor:   Oscar Gonzalez de Dios
              <mailto:oscar.gonzalezdedios@telefonica.com>
    Editor:   Mohamed Boucadair
              <mailto:mohamed.boucadair@orange.com>
    Author:   Luis Angel Munoz
              <mailto:luis-angel.munoz@vodafone.com>
    Author:   Alejandro Aguado
              <mailto:alejandro.aguado\_martin@nokia.com>;

  description
    "This YANG module defines a generic network-oriented model
     for the configuration of Layer 3 Virtual Private Networks."
```

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

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

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

```
revision 2021-09-28 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A Layer 3 VPN Network YANG Model";
}

/* Features */

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

/* Identities */

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

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

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

```
        customer.";
    }

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

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

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

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

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

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

```
}

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

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

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

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

/* Typedefs */

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

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

/* Groupings */
```



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

```

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

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



```
    default "vpn-common:static-rp";
    description
      "Type of RP discovery used.";
  }
  container bsr-candidates {
    when "derived-from-or-self(../rp-discovery-type, "
      + "'vpn-common:bsr-rp')" {
      description
        "Only applicable if discovery type is BSR-RP.";
    }
    description
      "Container for the customer Bootstrap Router (BSR)
        candidate's addresses.";
    leaf-list bsr-candidate-address {
      type inet:ip-address;
      description
        "Specifies the address of candidate BSR.";
    }
  }
}

container igmp {
  if-feature "vpn-common:igmp and vpn-common:ipv4";
  description
    "Includes IGMP-related parameters.";
  list static-group {
    key "group-addr";
    description
      "Multicast static source/group associated to the
        IGMP session.";
    leaf group-addr {
      type rt-types:ipv4-multicast-group-address;
      description
        "Multicast group IPv4 address.";
    }
    leaf source-addr {
      type rt-types:ipv4-multicast-source-address;
      description
        "Multicast source IPv4 address.";
    }
  }
  leaf max-groups {
    type uint32;
    description
      "Indicates the maximum number of groups.";
  }
  leaf max-entries {
    type uint32;
```

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

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

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

container l3vpn-ntw {
    description
```

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

```



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

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

```

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

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

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

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

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

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



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

```
        type string;
        description
            "Member link name.";
    }
}
}
}
}
container ip-connection {
    description
        "Defines IP connection parameters.";
    leaf l3-termination-point {
        type string;
        description
            "Specifies a reference to a local layer 3
            termination point such as a bridge domain
            interface.";
    }
    container ipv4 {
        if-feature "vpn-common:ipv4";
        description
            "IPv4-specific parameters.";
        leaf local-address {
            type inet:ipv4-address;
            description
                "The IP address used at the provider's
                interface.";
        }
        leaf prefix-length {
            type uint8 {
                range "0..32";
            }
            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;
            }
            must "not(derived-from-or-self(current(), "
                + "'slaac') or derived-from-or-self(current(), "
                + "'provider-dhcp-slaac'))" {
                error-message
                    "SLAAC is only applicable to IPv6.";
            }
            description

```

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

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

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

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

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

```
        description
            "Indicates the first address
             in the pool.";
    }
    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-alloc-
                + "cation-type, 'provider-dhcp') "
                + "or derived-from-or-self(..../address-alloc-";
        }
    }
}
```

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

```

        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.";
  }
}
}
case static-addresses {
  description
    "IPv6-specific parameters for static
    allocation.";
  leaf primary-address {
    type leafref {
      path "../address/address-id";
    }
    description
      "Principal address of the connection";
  }
  list address {
    key "address-id";
    description
      "Describes IPv6 addresses that are used.";
    leaf address-id {
      type string;
      description
        "An identifier of an IPv6 address.";
    }
    leaf customer-address {
      type inet:ipv6-address;
      description
        "An IPv6 address of the customer side.";
    }
  }
}
}
}
}
container routing-protocols {
  description
    "Defines routing protocols.";
  list routing-protocol {

```

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

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

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

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

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

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

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

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

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

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



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

messages' frequency between a PE and a BGP peer.

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

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

reference

"RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.4";

```
}
leaf hold-time {
  type uint16 {
    range "0 | 3..65535";
  }
  units "seconds";
  default "90";
  description
    "It indicates the maximum number of
    seconds that may elapse between the
    receipt of successive KEEPALIVE
    and/or UPDATE messages from the peer.

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

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

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

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

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

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

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



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

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

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

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

      The detection interval for the receiving
```

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

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

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

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

```
    leaf meticulous {
      type boolean;
      description
        "Enables meticulous mode.";
      reference
        "RFC 5880: Bidirectional Forwarding
          Detection (BFD), Section 6.7";
    }
  }
  uses vpn-common:service-status;
}

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



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

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

```
    description
      "Identification of the class of service.
       This identifier is internal to the
       administration.";
  }
  leaf inbound-rate-limit {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    units "percent";
    description
      "Specifies whether/how to rate-limit the
       inbound traffic matching this QoS policy.
       It is expressed as a percent of the value
       that is indicated in 'input-bandwidth'.";
  }
  leaf outbound-rate-limit {
    type decimal64 {
      fraction-digits 5;
      range "0..100";
    }
    units "percent";
    description
      "Specifies whether/how to rate-limit the
       outbound traffic matching this QoS policy.
       It is expressed as a percent of the value
       that is indicated in 'output-bandwidth'.";
  }
}

container qos-profile {
  description
    "QoS profile configuration.";
  list qos-profile {
    key "profile";
    description
      "QoS profile.
       Can be standard profile or customized
       profile.";
    leaf profile {
      type leafref {
        path "/l3vpn-ntw/vpn-profiles"
          + "/valid-provider-identifiers"
          + "/qos-profile-identifier/id";
      }
      description
        "QoS profile to be used.";
    }
  }
}
```

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

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

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

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

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

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

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



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

```
<CODE ENDS>
```

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]:

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

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

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

11. References

11.1. Normative References

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

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

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

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

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

11.2. Informative References

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

Progress, Internet-Draft, draft-ietf-idr-bgp-model-11, 11 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgp-model-11.txt>>.

[I-D.ietf-pim-yang]

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

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

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

[I-D.ietf-teas-enhanced-vpn]

Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-08, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-08.txt>>.

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

Farrel, A., Gray, E., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slices-04, 23 August 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slices-04.txt>>.

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

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

[IEEE802.1AX]

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

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

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

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

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

Appendix A. L3VPN Examples

A.1. 4G VPN Provisioning Example

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

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

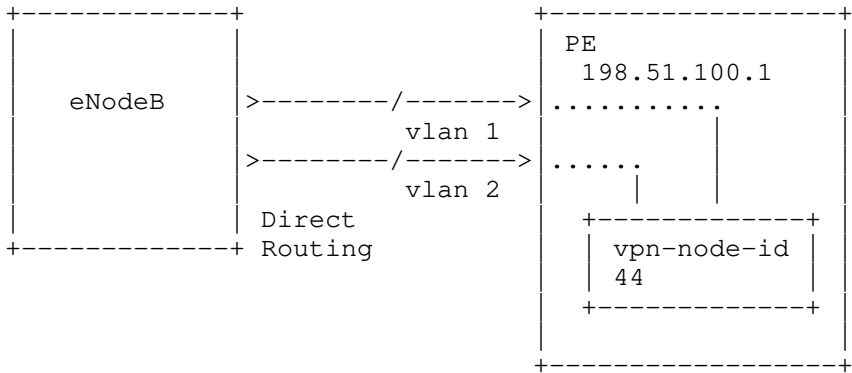


Figure 31: Mobile Backhaul Example

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

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

```

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

{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "4G",
        "customer-name": "mycustomer",
        "vpn-service-topology": "custom",
        "vpn-description": "VPN to deploy 4G services",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "simple-profile",
              "local-as": 65550,
              "rd": "0:65550:1",
              "address-family": [
                {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "vpn-target": [
                    {
                      "id": 1,
                      "route-targets": [
                        {
                          "route-target": "0:65550:1"
                        }
                      ],
                      "route-target-type": "both"
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  }
}

```

Figure 32: Create VPN Service

Second: Create a VPN node as depicted in Figure 33. In this type of service, the VPN node is equivalent to the VRF configured in the physical device ('ne-id'=198.51.100.1).

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

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

```
{
  "ietf-l3vpn-ntw:vpn-nodes": {
    "vpn-node": [
      {
        "vpn-node-id": "44",
        "ne-id": "198.51.100.1",
        "active-vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "simple-profile"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 33: Create VPN Node

Finally, two VPN network accesses are created using the same physical port ('interface-id'=1/1/1). Each 'vpn-network-access' has a particular VLAN (1,2) to differentiate the traffic between: Sync and data (Figure 34).

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

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

```
{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1",
        "description": "Interface SYNC to eNODE-B",
        "vpn-network-access-type": "ietf-vpn-common:point-to-point",
        "vpn-instance-profile": "simple-profile",
        "status": {
```

```
    "admin-status": {
      "status": "ietf-vpn-common:admin-up"
    }
  },
  "connection": {
    "encapsulation": {
      "type": "ietf-vpn-common:dot1q",
      "dot1q": {
        "cvlan-id": 1
      }
    }
  },
  "ip-connection": {
    "ipv4": {
      "local-address": "192.0.2.1",
      "prefix-length": 30,
      "address-allocation-type": "static-address",
      "static-addresses": {
        "primary-address": "1",
        "address": [
          {
            "address-id": "1",
            "customer-address": "192.0.2.2"
          }
        ]
      }
    },
    "ipv6": {
      "local-address": "2001:db8::1",
      "prefix-length": 64,
      "address-allocation-type": "static-address",
      "primary-address": "1",
      "address": [
        {
          "address-id": "1",
          "customer-address": "2001:db8::2"
        }
      ]
    }
  },
  "routing-protocols": {
    "routing-protocol": [
      {
        "id": "1",
        "type": "ietf-vpn-common:direct"
      }
    ]
  }
}
```



```
    },
    {
      "id": "1/1/1.2",
      "interface-id": "1/1/1",
      "description": "Interface DATA to eNODE-B",
      "vpn-network-access-type": "ietf-vpn-common:point-to-point",
      "vpn-instance-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "encapsulation": {
          "type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 2
          }
        }
      },
      "ip-connection": {
        "ipv4": {
          "local-address": "192.0.2.1",
          "prefix-length": 30,
          "address-allocation-type": "static-address",
          "static-addresses": {
            "primary-address": "1",
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          }
        },
        "ipv6": {
          "local-address": "2001:db8::1",
          "prefix-length": 64,
          "address-allocation-type": "static-address",
          "primary-address": "1",
          "address": [
            {
              "address-id": "1",
              "customer-address": "2001:db8::2"
            }
          ]
        }
      }
    },
  ],
},
```

```

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

```

Figure 34: Create VPN Network Access

A.2. Loopback Interface

An example of loopback interface is depicted in Figure 35.

```

{
  "ietf-l3vpn-ntw:vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "vpn-access-loopback",
        "interface-id": "Loopback1",
        "description": "An example of loopback interface.",
        "vpn-network-access-type": "ietf-vpn-common:loopback",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        },
        "ip-connection": {
          "ipv6": {
            "local-address": "2001:db8::4",
            "prefix-length": 128
          }
        }
      }
    ]
  }
}

```

Figure 35: VPN Network Access with a Loopback Interface (Message Body)

A.3. Overriding VPN Instance Profile Parameters

Figure 36 shows a simplified example to illustrate how some information that is provided at the VPN service level (particularly as part of the 'vpn-instance-profiles') can be overridden by the one configured at the VPN node level. In this example, PE3 and PE4 inherit the 'vpn-instance-profiles' parameters that are specified at the VPN service level, but PE1 and PE2 are provided with "maximum-routes" values at the VPN node level that override the ones that are specified at the VPN service level.

```
{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "override-example",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "HUB",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 64510,
              "rd-suffix": 1001,
              "address-family": [
                {
                  "address-family": "ietf-vpn-common:dual-stack",
                  "maximum-routes": [
                    {
                      "protocol": "ietf-vpn-common:any",
                      "maximum-routes": 100
                    }
                  ]
                }
              ]
            }
          ]
        },
        {
          "profile-id": "SPOKE",
          "role": "ietf-vpn-common:spoke-role",
          "local-as": 64510,
          "address-family": [
            {
              "address-family": "ietf-vpn-common:dual-stack",
              "maximum-routes": [
                {
                  "protocol": "ietf-vpn-common:any",
                  "maximum-routes": 1000
                }
              ]
            }
          ]
        }
      ]
    }
  }
}
```

```

    ]
  }
]
},
"vpn-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
              }
            ]
          }
        ]
      }
    ]
  }
}

```

```

    ]
  }
]
}
},
{
  "vpn-node-id": "PE3",
  "ne-id": "pe3",
  "router-id": "198.51.100.3",
  "active-vpn-instance-profiles": {
    "vpn-instance-profile": [
      {
        "profile-id": "SPOKE"
      }
    ]
  }
},
{
  "vpn-node-id": "PE4",
  "ne-id": "pe4",
  "router-id": "198.51.100.4",
  "active-vpn-instance-profiles": {
    "vpn-instance-profile": [
      {
        "profile-id": "SPOKE"
      }
    ]
  }
}
]
}
}
]
}
}
}

```

Figure 36: VPN Instance Profile Example (Message Body)

A.4. Multicast VPN Provisioning Example

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

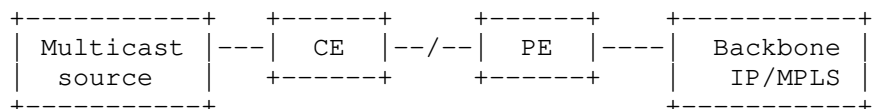


Figure 37: Multicast L3VPN Service Example

An example is provided below to illustrate how to configure a multicast L3VPN service using the L3NM.

First, the multicast service is created together with a generic VPN instance profile (see the excerpt of the request message body shown in Figure 38)

```

{
  "ietf-l3vpn-ntw:vpn-services": {
    "vpn-service": [
      {
        "vpn-id": "Multicast-IPTV",
        "vpn-description": "Multicast IPTV VPN service",
        "customer-name": "a-name",
        "vpn-service-topology": "ietf-vpn-common:hub-spoke",
        "vpn-instance-profiles": {
          "vpn-instance-profile": [
            {
              "profile-id": "multicast",
              "role": "ietf-vpn-common:hub-role",
              "local-as": 65536,
              "multicast": {
                "rp": {
                  "rp-group-mappings": {
                    "rp-group-mapping": [
                      {
                        "id": 1,
                        "rp-address": "203.0.113.17",
                        "groups": {
                          "group": [
                            {
                              "id": 1,
                              "group-address": "239.130.0.0/15"
                            }
                          ]
                        }
                      ]
                    }
                  },
                  "rp-discovery": {
                    "rp-discovery-type": "ietf-vpn-common:static-rp"
                  }
                }
              }
            ]
          }
        ]
      }
    ]
  }
}

```

Figure 38: Create Multicast VPN Service (Excerpt of the Message Request Body)

Then, the VPN nodes are created (see the excerpt of the request message body shown in Figure 39). In this example, the VPN node will represent VRF configured in the physical device.

```
{
  "ietf-l3vpn-ntw:vpn-node": [
    {
      "vpn-node-id": "500003105",
      "description": "VRF-IPTV-MULTICAST",
      "ne-id": "198.51.100.10",
      "router-id": "198.51.100.10",
      "active-vpn-instance-profiles": {
        "vpn-instance-profile": [
          {
            "profile-id": "multicast",
            "rd": "65536:31050202"
          }
        ]
      }
    }
  ]
}
```

Figure 39: Create Multicast VPN Node (Excerpt of the Message Request Body)

Finally, create the VPN network access with multicast enabled (see the excerpt of the request message body shown in Figure 40).

```
{
  "ietf-l3vpn-ntw:vpn-network-access": {
    "id": "1/1/1",
    "description": "Connected-to-source",
    "vpn-network-access-type": "ietf-vpn-common:point-to-point",
    "vpn-instance-profile": "multicast",
    "status": {
      "admin-status": {
        "status": "vpn-common:admin-up"
      },
      "ip-connection": {
        "ipv4": {
          "local-address": "203.0.113.1",
          "prefix-length": 30,
          "address-allocation-type": "static-address",
          "static-addresses": {
            "primary-address": "1",
            "address": [
              {

```



```

        "address-id": "1",
        "customer-address": "203.0.113.2"
    }
}
},
"routing-protocols": {
    "routing-protocol": [
        {
            "id": "1",
            "type": "ietf-vpn-common:bgp-routing",
            "bgp": {
                "description": "Connected to CE",
                "peer-as": "65537",
                "address-family": "ietf-vpn-common:ipv4",
                "neighbor": "203.0.113.2"
            }
        }
    ]
},
"service": {
    "inbound-bandwidth": "1000000000",
    "outbound-bandwidth": "1000000000",
    "mtu": 1500,
    "multicast": {
        "access-type": "source-only",
        "address-family": "ietf-vpn-common:ipv4",
        "protocol-type": "router",
        "pim": {
            "hello-interval": 30,
            "status": {
                "admin-status": {
                    "status": "ietf-vpn-common:admin-up"
                }
            }
        }
    }
}
}
}
}

```

Figure 40: Create VPN Network Access (Excerpt of the Message Request Body)

Appendix B. Implementation Status

This section records the status of known implementations of the YANG module defined by this specification at the time of posting of this document and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note to the RFC Editor: As per [RFC7942] guidelines, please remove this Implementation Status appendix prior publication.

B.1. Nokia Implementation

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

B.2. Huawei Implementation

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

B.3. Infinera Implementation

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

B.4. Ribbon-ECI Implementation

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

B.5. Juniper Implementation

<https://github.com/IETF-OPSAWG-WG/lxnm/blob/master/Implementattion/Juniper>

Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from (listed alphabetically): Raul Arco, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Greg Mirsky, and Tom Petch. Many thanks to them. Thanks to Philip Eardly for the review of an early version of the document.

Daniel King, Daniel Voyer, Luay Jalil, and Stephane Litkowski contributed to early version of the individual submission. Many thanks to Robert Wilton for the AD review. Thanks to Andrew Malis for the routing directorate review, Rifaat Shekh-Yusef for the security directorate review, Qin Wu for the opsdire review, and Pete Resnick for the genart directorate review. Thanks to Michael Scharf for the discussion on TCP-AO. Thanks to Martin Duke, Lars Eagert, Zaheduzzaman Sarker, Roman Danyliw, Erik Kline, Benjamin Kaduk, Francesca Palombini, and Eric Vyncke for the IESG review.

This work was supported in part by the European Commission funded H2020-ICT-2016-2 METRO-HAUL project (G.A. 761727) and Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (G.A. 101015857).

Contributors

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

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

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

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

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

Paul Sherratt
Gamma Telecom
Email: paul.sherratt@gamma.co.uk

Authors' Addresses

Samier Barguil
Telefonica
Madrid
Spain

Email: samier.barguilgiraldo.ext@telefonica.com

Oscar Gonzalez de Dios (editor)
Telefonica
Madrid
Spain

Email: oscar.gonzalezdedios@telefonica.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Luis Angel Munoz
Vodafone
Spain

Email: luis-angel.munoz@vodafone.com

Alejandro Aguado
Nokia
Madrid
Spain

Email: alejandro.aguado_martin@nokia.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: April 28, 2021

Q. Wu, Ed.
Huawei
M. Boucadair, Ed.
Orange
D. Lopez
Telefonica I+D
C. Xie
China Telecom
L. Geng
China Mobile
October 25, 2020

A Framework for Automating Service and Network Management with YANG
draft-ietf-opsawg-model-automation-framework-10

Abstract

Data models provide a programmatic approach to represent services and networks. Concretely, they can be used to derive configuration information for network and service components, and state information that will be monitored and tracked. Data models can be used during the service and network management life cycle, such as service instantiation, provisioning, optimization, monitoring, diagnostic, and assurance. Data models are also instrumental in the automation of network management, and they can provide closed-loop control for adaptive and deterministic service creation, delivery, and maintenance.

This document describes a framework for service and network management automation that takes advantage of YANG modeling technologies. This framework is drawn from a network operator perspective irrespective of the origin of a data model; it can thus accommodate YANG modules that are developed outside the IETF.

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 28, 2021.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology and Acronyms	5
2.1. Terminology	5
2.2. Acronyms	7
3. Architectural Concepts and Goals	7
3.1. Data Models: Layering and Representation	7
3.2. Automation of Service Delivery Procedures	12
3.3. Service Fulfillment Automation	13
3.4. YANG Modules Integration	13
4. Functional Blocks and Interactions	14
4.1. Service Lifecycle Management Procedure	15
4.1.1. Service Exposure	15
4.1.2. Service Creation/Modification	15
4.1.3. Service Assurance	16
4.1.4. Service Optimization	16
4.1.5. Service Diagnosis	16
4.1.6. Service Decommission	17
4.2. Service Fullfillment Management Procedure	17
4.2.1. Intended Configuration Provision	17
4.2.2. Configuration Validation	18
4.2.3. Performance Monitoring	18
4.2.4. Fault Diagnostic	19
4.3. Multi-Layer/Multi-Domain Service Mapping	19
4.4. Service Decomposition	19
5. YANG Data Model Integration Examples	20
5.1. L2VPN/L3VPN Service Delivery	20

5.2.	VN Lifecycle Management	22
5.3.	Event-based Telemetry in the Device Self Management	23
6.	Security Considerations	24
6.1.	Service Level	25
6.2.	Network Level	26
6.3.	Device Level	26
7.	IANA Considerations	26
8.	Acknowledgements	26
9.	Contributors	27
10.	References	27
10.1.	Normative References	27
10.2.	Informative References	28
Appendix A.	Layered YANG Modules Examples Overview	37
A.1.	Service Models: Definition and Samples	37
A.2.	Schema Mount	38
A.3.	Network Models: Samples	38
A.4.	Device Models: Samples	41
A.4.1.	Model Composition	43
A.4.2.	Device Management	43
A.4.3.	Interface Management	43
A.4.4.	Some Device Model Examples	43
Authors' Addresses	46

1. Introduction

Service management systems usually comprise service activation/provision and service operation. Current service delivery procedures, from the processing of customer requirements and orders to service delivery and operation, typically assume the manipulation of data sequentially into multiple Operations Support System (OSS) or Business Support System (BSS) applications that may be managed by different departments within the service provider's organization (e.g., billing factory, design factory, network operation center). Many of these applications have been developed in-house over the years and operate in a silo mode:

- o The lack of standard data input/output (i.e., data model) raises many challenges in system integration and often results in manual configuration tasks.
- o Service fulfillment systems might have a limited visibility on the network state and therefore have slow response to network changes.

Software Defined Networking (SDN) becomes crucial to address these challenges. SDN techniques are meant to automate the overall service delivery procedures and typically rely upon standard data models. These models are used to not only reflect service providers' savoir-faire, but also to dynamically instantiate and enforce a set of

service-inferred policies that best accommodate what has been defined and possibly negotiated with the customer. [RFC7149] provides a first tentative attempt to rationalize that service provider's view on the SDN space by identifying concrete technical domains that need to be considered and for which solutions can be provided:

- o Techniques for the dynamic discovery of topology, devices, and capabilities, along with relevant information and data models that are meant to precisely document such topology, devices, and their capabilities.
- o Techniques for exposing network services [RFC8309] and their characteristics.
- o Techniques used by service-derived dynamic resource allocation and policy enforcement schemes, so that networks can be programmed accordingly.
- o Dynamic feedback mechanisms that are meant to assess how efficiently a given policy (or a set thereof) is enforced from a service fulfillment and assurance perspectives.

Models are key for each of the aforementioned four technical items. Service and network management automation is an important step to improve the agility of network operations. Models are also important to ease integrating multi-vendor solutions.

YANG [RFC7950] module developers have taken both top-down and bottom-up approaches to develop modules [RFC8199] and to establish a mapping between a network technology and customer requirements at the top or abstracting common constructs from various network technologies at the bottom. At the time of writing this document (2020), there are many YANG data models including configuration and service models that have been specified or are being specified by the IETF. They cover many of the networking protocols and techniques. However, how these models work together to configure a function, manage a set of devices involved in a service, or provide a service is something that is not currently documented either within the IETF or other Standards Development Organizations (SDOs).

Many of the YANG modules listed in this document are used to exchange data between NETCONF/RESTCONF clients and servers [RFC6241][RFC8040]. Nevertheless, YANG is a transport-independent data modeling language. It can thus be used independently of NETCONF/RESTCONF. For example, YANG can be used to define abstract data structures [RFC8791] that can be manipulated by other protocols (e.g., [I-D.ietf-dots-rfc8782-bis]).

This document describes an architectural framework for service and network management automation (Section 3) that takes advantage of YANG modeling technologies and investigates how YANG data models at different layers interact with each other (e.g., service mapping, model composition) in the context of service delivery and fulfillment (Section 4). Concretely, the following benefits can be provided:

- o Allow for vendor-agnostic interfaces to manage a service and the underlying network.
- o Move from deployment schemes where vendor-specific network managers are required to a scheme where the entities that are responsible for orchestrating and controlling services and network resources provided by multi-vendor devices are unified.
- o Ease data inheritance and reusability among the various architecture layers thus promoting a network-wise provisioning instead of device-specific configuration.
- o Dynamically feed a decision-making process (e.g., Controllers, Orchestrators) with notifications that will trigger appropriate actions, allowing that decision-making process to continuously adjust a network (and thus, the involved resources) to deliver the service that conforms to the intended parameters (service objectives).

This framework is drawn from a network operator perspective irrespective of the origin of a data model; it can also accommodate YANG modules that are developed outside the IETF. The document covers service models that are used by an operator to expose its services and capture service requirements from the customers (including other operators). Nevertheless, the document does not elaborate on the communication protocol(s) that makes use of these service models in order to request and deliver a service. Such considerations are out of scope.

The document identifies a list of use cases to exemplify the proposed approach (Section 5), but it does not claim nor aim to be exhaustive. Appendix A lists some examples to illustrate the layered YANG modules view.

2. Terminology and Acronyms

2.1. Terminology

The following terms are defined in [RFC8309][RFC8199] and are not redefined here:

- o Network Operator
- o Customer
- o Service
- o Data Model
- o Service Model
- o Network Element Module

In addition, the document makes use of the following terms:

Network Model: Describes a network level abstraction (or a subset of aspects of a network infrastructure), including devices and their subsystems, and relevant protocols operating at the link and network layers across multiple devices. This model corresponds to the network configuration model discussed in [RFC8309].

It can be used by a network operator to allocate resources (e.g., tunnel resource, topology resource) for the service or schedule resources to meet the service requirements defined in a service model.

Network Domain: Refers to a network partitioning that is usually followed by network operators to delimit parts of their network. "access network" and "core network" are examples of network domains.

Device Model: Refers to the Network Element YANG data model described in [RFC8199] or the device configuration model discussed in [RFC8309].

Device models are also used to refer to model a function embedded in a device (e.g., Network Address Translation (NAT) [RFC8512], Access Control Lists (ACLs) [RFC8519]).

Pipe: Refers to a communication scope where only one-to-one (1:1) communications are allowed. The scope can be identified between ingress and egress nodes, two service sites, etc.

Hose: Refers to a communication scope where one-to-many (1:N) communications are allowed (e.g., one site to multiple sites).

Funnel: Refers to a communication scope where many-to-one (N:1) communications are allowed.

2.2. Acronyms

The following acronyms are used in the document:

ACL	Access Control List
AS	Autonomous System
AP	Access Point
CE	Customer Edge
DBE	Data Border Element
E2E	End-to-End
ECA	Event Condition Action
L2VPN	Layer 2 Virtual Private Network
L3VPN	Layer 3 Virtual Private Network
L3SM	L3VPN Service Model
L3NM	L3VPN Network Model
NAT	Network Address Translation
OAM	Operations, Administration, and Maintenance
OWD	One-Way Delay
PE	Provider Edge
PM	Performance Monitoring
QoS	Quality of Service
RD	Route Distinguisher
RT	Route Target
SBE	Session Border Element
SDN	Software Defined Networking
SP	Service Provider
TE	Traffic Engineering
VN	Virtual Network
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding

3. Architectural Concepts and Goals

3.1. Data Models: Layering and Representation

As described in Section 2 of [RFC8199], layering of modules allows for better reusability of lower-layer modules by higher-level modules while limiting duplication of features across layers.

Data models in the context of network management can be classified into service, network, and device models. Different service models may rely on the same set of network and/or device models.

Service models traditionally follow a top-down approach and are mostly customer-facing YANG modules providing a common model construct for higher level network services (e.g., Layer 3 Virtual Private Network (L3VPN)). Such modules can be mapped to network technology-specific modules at lower layers (e.g., tunnel, routing,

Quality of Service (QoS), security). For example, service models can be used to characterise the network service(s) to be ensured between service nodes (ingress/egress) such as:

- o the communication scope (pipe, hose, funnel, ...),
- o the directionality (inbound/outbound),
- o the traffic performance guarantees expressed using metrics such as One-Way Delay (OWD) [RFC7679] or One-Way Loss [RFC7680]; a summary of performance metrics maintained by IANA can be found in [IPPM],
- o link capacity [RFC5136] [I-D.ietf-ippm-capacity-metric-method],
- o etc.

Figure 1 depicts the example of a VoIP service that relies upon connectivity services offered by a network operator. In this example, the VoIP service is offered to the network operator's customers by Service Provider (SP1). In order to provide global VoIP reachability, SP1 service site interconnects with other Service Providers service sites typically by interconnecting Session Border Elements (SBEs) and Data Border Elements (DBEs) [RFC5486][RFC6406]. For other VoIP destinations, sessions are forwarded over the Internet. These connectivity services can be captured in a YANG service model that reflects the service attributes that are shown in Figure 2. This example follows the IP Connectivity Provisioning Profile template defined in [RFC7297].

In reference to Figure 2, "Full traffic performance guarantees class" refers to a service class where all traffic performance metrics included in the service model (OWD, loss, delay variation) are guaranteed, while "Delay traffic performance guarantees class" refers to a service class where only OWD is guaranteed.

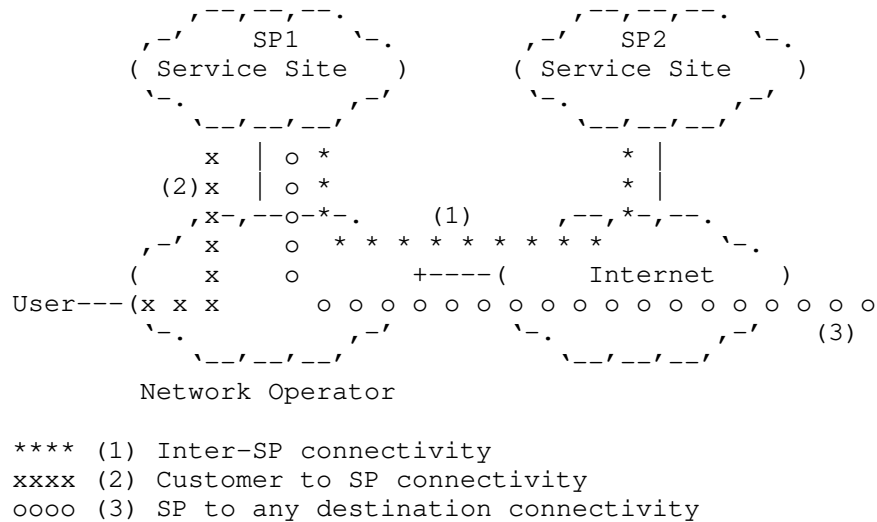


Figure 1: An Example of Service Connectivity Components

Connectivity: Scope and Guarantees

- (1) Inter-SP connectivity
 - Pipe scope from the local to the remote SBE/DBE
 - Full traffic performance guarantees class
- (2) Customer to SP connectivity
 - Hose/Funnel scope connecting the local SBE/DBE to the customer access points
 - Full traffic performance guarantees class
- (3) SP to any destination connectivity
 - Hose/Funnel scope from the local SBE/DBE to the Internet gateway
 - Delay traffic performance guarantees class

Flow Identification

- * Destination IP address (SBE, DBE)
- * DSCP marking

Traffic Isolation

- * VPN

Routing & Forwarding

- * Routing rule to exclude some ASes from the inter-domain paths

Notifications (including feedback)

- * Statistics on aggregate traffic to adjust capacity
- * Failures
- * Planned maintenance operations
- * Triggered by thresholds

Figure 2: Sample Attributes Captured in a Service Model

Network models are mainly network resource-facing modules; they describe various aspects of a network infrastructure, including devices and their subsystems, and relevant protocols operating at the link and network layers across multiple devices (e.g., network topology and traffic-engineering tunnel modules).

Device (and function) models usually follow a bottom-up approach and are mostly technology-specific modules used to realize a service (e.g., BGP, ACL).

Each level maintains a view of the supported YANG modules provided by lower levels (see for example, Appendix A). Mechanisms such as YANG library [RFC8525] can be used to expose which YANG modules are supported by nodes in lower levels.

Figure 3 illustrates the overall layering model. The reader may refer to Section 4 of [RFC8309] for an overview of "Orchestrator" and "Controller" elements. All these elements (i.e., Orchestrator(s), Controller(s), device(s)) are under the responsibility of the same operator.

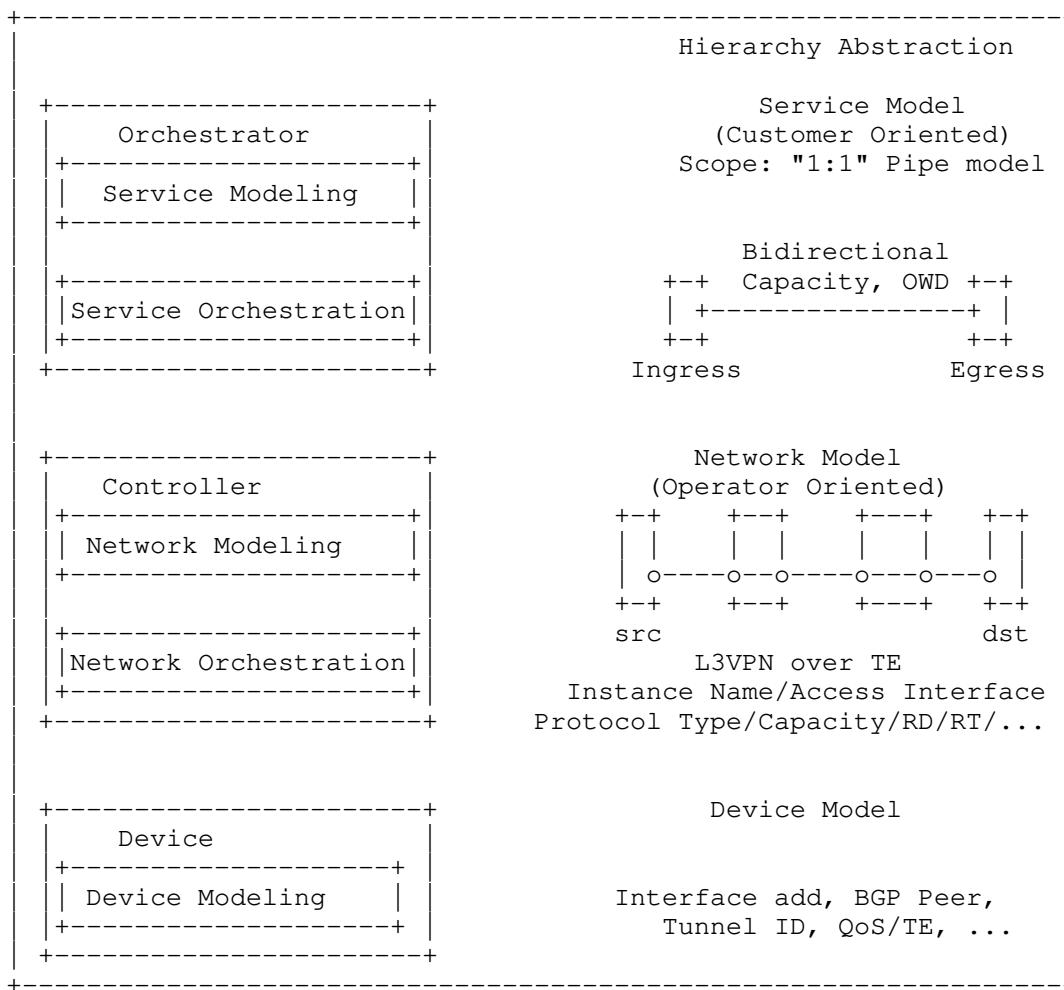


Figure 3: Layering and Representation Within a Network Operator

A composite service offered by a network operator may rely on services from other operators. In such case, the network operator acts as a customer to request services from other networks. The operators providing these services will then follow the layering depicted in Figure 3. The mapping between a composite service and a third-party service is maintained at the orchestration level. From a data plane perspective, appropriate traffic steering policies (e.g., Service Function Chaining [RFC7665]) are managed by the network controllers to guide how/when a third party service is invoked for flows bound to a composite service.

The layering model depicted in Figure 3 does not make any assumption about the location of the various entities (e.g., controller, orchestrator) within the network. As such, the architecture does not preclude deployments where, for example, the controller is embedded on a device that hosts other functions that are controlled via YANG modules.

In order to ease the mapping between layers and data reuse, this document focuses on service models that are modelled using YANG. Nevertheless, fully compliant with Section 3 of [RFC8309], Figure 3 does not preclude service models to be modelled using other data modelling languages than YANG.

3.2. Automation of Service Delivery Procedures

Service models can be used by a network operator to expose its services to its customers. Exposing such models allows to automate the activation of service orders and thus the service delivery. One or more monolithic service models can be used in the context of a composite service activation request (e.g., delivery of a caching infrastructure over a VPN). Such models are used to feed a decision-making intelligence to adequately accommodate customer's needs.

Also, such models may be used jointly with services that require dynamic invocation. An example is provided by the service modules defined by the DOTS WG to dynamically trigger requests to handle Distributed Denial-of-Service (DDoS) attacks [RFC8783]. The service filtering request modelled using [RFC8783] will be translated into device-specific filtering (e.g., ACLs defined in [RFC8519]) that fulfils the service request.

Network models can be derived from service models and used to provision, monitor, instantiate the service, and provide lifecycle management of network resources. Doing so is meant to:

- o expose network resources to customers (including other network operators) to provide service fulfillment and assurance.
- o allow customers (or network operators) to dynamically adjust the network resources based on service requirements as described in service models (e.g., Figure 2) and the current network performance information described in the telemetry modules.

Note that it is out of the scope of this document to elaborate on the communication protocols that are used to implement the interface between the service ordering (customer) and service order handling (provider).

3.3. Service Fulfillment Automation

To operate a service, the settings of the parameters in the device models are derived from service models and/or network models and are used to:

- o Provision each involved network function/device with the proper configuration information.
- o Operate the network based on service requirements as described in the service model(s) and local operational guidelines.

In addition, the operational state including configuration that is in effect together with statistics should be exposed to upper layers to provide better network visibility and assess to what extent the derived low level modules are consistent with the upper level inputs.

Filters are enforced on the notifications that are communicated to Service layers. The type and frequency of notifications may be agreed in the service model.

Note that it is important to correlate telemetry data with configuration data to be used for closed loops at the different stages of service delivery, from resource allocation to service operation, in particular.

3.4. YANG Modules Integration

To support top-down service delivery, YANG modules at different levels or at the same level need to be integrated together for proper service delivery (including, proper network setup). For example, the service parameters captured in service models need to be decomposed into a set of configuration/notification parameters that may be specific to one or more technologies; these technology-specific parameters are grouped together to define technology-specific device level models or network level models.

In addition, these technology-specific device or network models can be further integrated with each other using the schema mount mechanism [RFC8528] to provision each involved network function/device or each involved network domain to support newly added modules or features. A collection of device models integrated together can be loaded and validated during implementation.

High-level policies can be defined at service or network models (e.g., "Autonomous System Number (ASN) Exclude" in the example depicted in Figure 2). Device models will be tweaked accordingly to provide policy-based management. Policies can also be used for

telemetry automation, e.g., policies that contain conditions to trigger the generation and pushing of new telemetry data.

4. Functional Blocks and Interactions

The architectural considerations described in Section 3 lead to the lifecycle management architecture illustrated in Figure 4 and described in the following subsections.

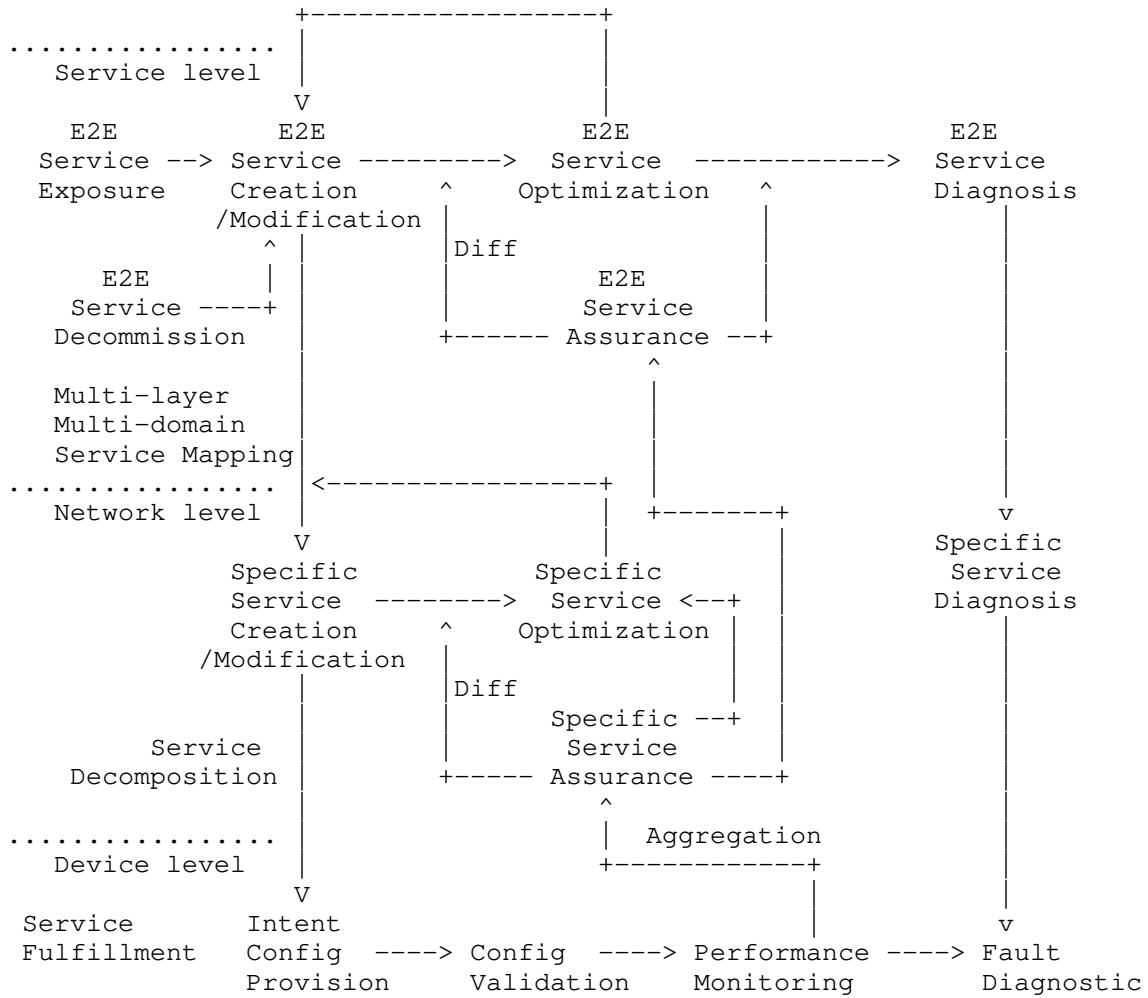


Figure 4: Service and Network Lifecycle Management

4.1. Service Lifecycle Management Procedure

Service lifecycle management includes end-to-end service lifecycle management at the service level and technology specific network lifecycle management at the network level.

The end-to-end service lifecycle management is technology-independent service management and spans across multiple network domains and/or multiple layers while technology specific service lifecycle management is technology domain specific or layer specific service lifecycle management.

4.1.1. Service Exposure

A service in the context of this document (sometimes called, Network Service) is some form of connectivity between customer sites and the Internet or between customer sites across the operator's network and across the Internet.

Service exposure is used to capture services offered to customers (ordering and order handling). One example is that a customer can use a L3VPN Service Model (L3SM) to request L3VPN service by providing the abstract technical characterization of the intended service between customer sites.

Service model catalogs can be created along to expose the various services and the information needed to invoke/order a given service.

4.1.2. Service Creation/Modification

A customer is usually unaware of the technology that the network operator has available to deliver the service, so the customer does not make requests specific to the underlying technology but is limited to making requests specific to the service that is to be delivered. This service request can be filled using a service model.

Upon receiving a service request, and assuming that appropriate authentication and authorization checks have been made with success, the service orchestrator/management system should verify whether the service requirements in the service request can be met (i.e., whether there are sufficient resources that can be allocated with the requested guarantees).

If the request is accepted, the service orchestrator/management system maps such service request to its view. This view can be described as a technology specific network model or a set of technology specific device models and this mapping may include a

choice of which networks and technologies to use depending on which service features have been requested.

In addition, a customer may require to change the underlying network infrastructure to adapt to new customer's needs and service requirements (e.g., service a new customer site, add a new access link, provide disjoint paths). This service modification can be issued following the same service model used by the service request.

Withdrawing a service is discussed in Section 4.1.6.

4.1.3. Service Assurance

The performance measurement telemetry (Section 4.2) can be used to provide service assurance at Service and/or Network levels. Performance measurement telemetry model can tie with service or network models to monitor network performance or Service Level Agreement.

4.1.4. Service Optimization

Service optimization is a technique that gets the configuration of the network updated due to network changes, incident mitigation, or new service requirements. One example is once a tunnel or a VPN is setup, Performance monitoring information or telemetry information per tunnel (or per VPN) can be collected and fed into the management system. If the network performance doesn't meet the service requirements, the management system can create new VPN policies capturing network service requirements and populate them into the network.

Both network performance information and policies can be modelled using YANG. With Policy-based management, self-configuration and self-optimization behavior can be specified and implemented.

The overall service optimization is managed at the service level, while the network level is responsible for the optimization of the specific network services it provides.

4.1.5. Service Diagnosis

Operations, Administration, and Maintenance (OAM) are important networking functions for service diagnosis that allow network operators to:

- o monitor network communications (i.e., reachability verification and Continuity Check)

- o troubleshoot failures (i.e., fault verification and localization)
- o monitor service-level agreements and performance (i.e., performance management)

When the network is down, service diagnosis should be in place to pinpoint the problem and provide recommendations (or instructions) for the network recovery.

The service diagnosis information can be modelled as technology-independent Remote Procedure Call (RPC) operations for OAM protocols and technology-independent abstraction of key OAM constructs for OAM protocols [RFC8531][RFC8533]. These models can be used to provide consistent configuration, reporting, and presentation for the OAM mechanisms used to manage the network.

Refer to Section 4.2.4 for the device-specific side.

4.1.6. Service Decommission

Service decommission allows a customer to stop the service by removing the service from active status and thus releasing the network resources that were allocated to the service. Customers can also use the service model to withdraw the subscription to a service.

4.2. Service Fullfillment Management Procedure

4.2.1. Intended Configuration Provision

Intended configuration at the device level is derived from network models at the network level or service model at the service level and represents the configuration that the system attempts to apply. Take L3SM as a service model example to deliver a L3VPN service, there is a need to map the L3VPN service view defined in the service model into a detailed intended configuration view defined by specific configuration models for network elements; the configuration information includes:

- o Virtual Routing and Forwarding (VRF) definition, including VPN policy expression
- o Physical Interface(s)
- o IP layer (IPv4, IPv6)
- o QoS features such as classification, profiles, etc.

- o Routing protocols: support of configuration of all protocols listed in a service request, as well as routing policies associated with those protocols.
- o Multicast support
- o Address sharing
- o Security (e.g., access control, authentication, encryption)

These specific configuration models can be used to configure Provider Edge (PE) and Customer Edge (CE) devices within a site, e.g., a BGP policy model can be used to establish VPN membership between sites and VPN Service Topology.

Note that in networks with legacy devices (that support proprietary modules or do not support YANG at all), an adaptation layer is likely to be required at the network level so that these devices can be involved in the delivery of the network services.

This interface is also used to handle service withdrawal (Section 4.1.6).

4.2.2. Configuration Validation

Configuration validation is used to validate intended configuration and ensure the configuration take effect.

For example, if a customer creates an interface "eth-0/0/0" but the interface does not physically exist at this point, then configuration data appears in the <intended> status but does not appear in the <operational> datastore. More details about <intended> and <operational> datastores can be found in Section 5.1 of [RFC8342].

4.2.3. Performance Monitoring

When a configuration is in effect in a device, <operational> datastore holds the complete operational state of the device including learned, system, default configuration, and system state. However, the configurations and state of a particular device does not have the visibility on the whole network or how packets are going to be forwarded through the entire network. Therefore, it becomes more difficult to operate the entire network without understanding the current status of the network.

The management system should subscribe to updates of a YANG datastore in all the network devices for performance monitoring purposes and

build a full topological visibility of the network by aggregating (and filtering) these operational state from different sources.

4.2.4. Fault Diagnostic

When configuration is in effect in a device, some devices may be mis-configured (e.g., device links are not consistent in both sides of the network connection) or network resources might be mis-allocated. Therefore, services may be negatively affected without knowing the root cause in the network.

Technology-dependent nodes and RPC commands are defined in technology-specific YANG data models which can use and extend the base model described in Section 4.1.5 to deal with these issues.

These RPC commands received in the technology-dependent node can be used to trigger technology-specific OAM message exchanges for fault verification and fault isolation. For example, TRILL Multicast Tree Verification (MTV) RPC command [I-D.ietf-trill-yang-oam] can be used to trigger Multi-Destination Tree Verification Message defined in [RFC7455] to verify TRILL distribution tree integrity.

4.3. Multi-Layer/Multi-Domain Service Mapping

Multi-layer/Multi-domain Service Mapping allows to map an end-to-end abstract view of the service segmented at different layers and/or different network domains into domain-specific views.

One example is to map service parameters in the L3SM into configuration parameters such as Route Distinguisher (RD), Route Target (RT), and VRF in the L3VPN Network Model (L3NM).

Another example is to map service parameters in the L3SM into Traffic Engineered (TE) tunnel parameters (e.g., Tunnel ID) in TE model and Virtual Network (VN) parameters (e.g., Access Point (AP) list, VN members) in the YANG data model for VN operation [I-D.ietf-teas-actn-vn-yang].

4.4. Service Decomposition

Service Decomposition allows to decompose service models at the service level or network models at the network level into a set of device models at the device level. These device models may be tied to specific device types or classified into a collection of related YANG modules based on service types and features offered, and load at the implementation time before configuration is loaded and validated.

5. YANG Data Model Integration Examples

The following subsections provide some YANG data models integration examples.

5.1. L2VPN/L3VPN Service Delivery

In reference to Figure 5, the following steps are performed to deliver the L3VPN service within the network management automation architecture defined in Section 4:

1. The Customer requests to create two sites (as per Service Creation in Section 4.2.1) relying upon L3SM with each site having one network access connectivity, for example:
 - * Site A: network-access A, link-capacity = 20 Mbps, class "foo", guaranteed-capacity-percent = 10, average-one-way-delay = 70 ms.
 - * Site B: network-access B, link-capacity = 30 Mbps, class "foo1", guaranteed-capacity-percent = 15, average-one-way-delay = 60 ms.
2. The Orchestrator extracts the service parameters from the L3SM. Then, it uses them as input to the Service Mapping in Section 4.3 to translate them into an orchestrated configuration parameters (e.g., RD, RT, VRF) that are part of the L3NM specified in [I-D.ietf-opsawg-l3sm-l3nm].
3. The Controller takes the orchestrated configuration parameters in the L3NM and translates them into orchestrated (Service Decomposition in Section 4.4) configuration of network elements that are part of, e.g., BGP, QoS, Network Instance, IP management, and interface models.

[I-D.ogondio-opsawg-uni-topology] can be used for representing, managing, and controlling the User Network Interface (UNI) topology.

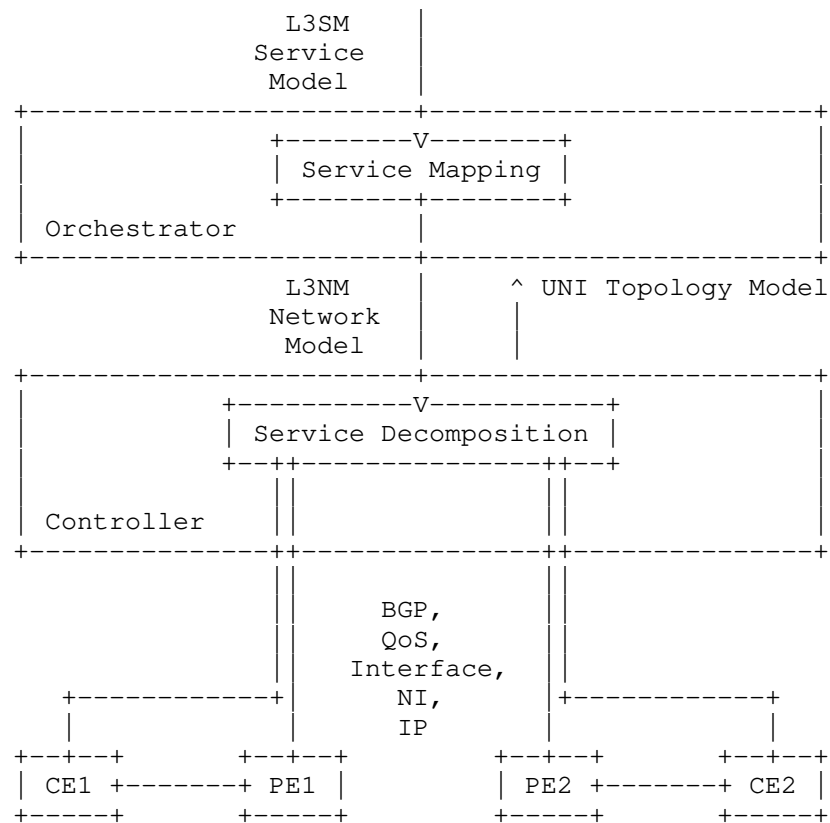


Figure 5: L3VPN Service Delivery Example (Current)

L3NM inherits some of data elements from the L3SM. Nevertheless, the L3NM as currently designed in [I-D.ietf-opsawg-l3sm-l3nm] does not expose some information to the above layer such as the capabilities of an underlying network (which can be used to drive service order handling) or notifications (to notify subscribers about specific events or degradations as per agreed SLAs). Some of this information can be provided using, e.g., [I-D.www-opsawg-yang-vpn-service-pm]. A target overall model is depicted in Figure 6.

allow the NETCONF server to send updates only when the value exceeds a certain threshold for the first time, but not again until the threshold is cleared), which constitute an Event/Condition/Action (ECA) policy or an event-driven policy control logic that can be executed on the device (e.g., [I-D.www-netmod-event-yang]).

2. To provide rapid autonomic response that can exhibit self-management properties, the Controller pushes the ECA policy to the network device and delegates the network control logic to the network device.
3. The network device uses the ECA model to subscribe to the event source, e.g., an event stream or datastore state data conveyed to the server via YANG Push subscription [RFC8641], monitors state parameters, and takes simple and instant actions when an associated event condition on state parameters is met. ECA notifications can be generated as the result of actions based on event stream subscription or datastore subscription (model-driven telemetry operation discussed in Section 4.2.3).

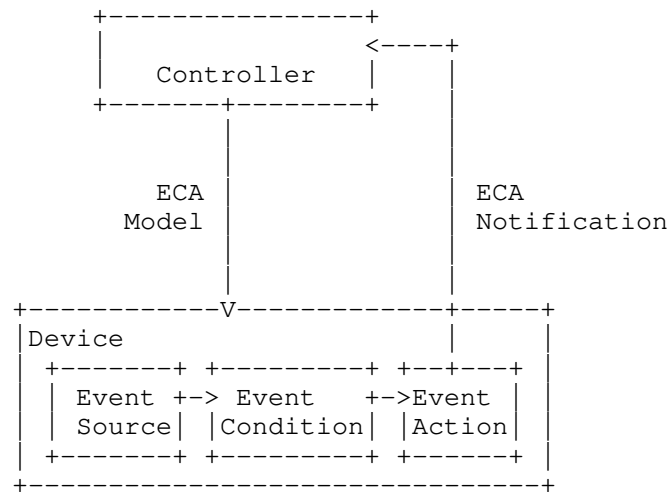


Figure 8: Event-based Telemetry

6. Security Considerations

Many of the YANG modules cited in this document define schema 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 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.

Security considerations specific to each of the technologies and protocols listed in the document are discussed in the specification documents of each of these protocols.

In order to prevent leaking sensitive information and the "confused deputy" problem [Hardy] in general, special care should be considered when translating between the various layers in Section 4 or when aggregating data retrieved from various sources. Authorization and authentication checks should be performed to ensure that a data is available to an authorized entity. The network operator must enforce means to protect privacy-related information included in customer-facing models.

To detect misalignment between layers that might be induced by misbehaving nodes, upper layers should continuously monitor the perceived service (Section 4.1.4) and should proceed with checks to assess that the provided service complies with the expected service and that the data reported by an underlying layer is matching the perceived service by the above layer. Such checks are the responsibility of the service diagnosis (Section 4.1.5).

When a YANG module includes security-related parameters, it is recommended to include the relevant information as part of the service assurance to track the correct functioning of the security mechanisms.

Additional considerations are discussed in the following subsections.

6.1. Service Level

A provider may rely on services offered by other providers to build composite services. Appropriate mechanisms should be enabled by the provider to monitor and detect a service disruption from these providers. The characterization of a service disruption (including, mean time between failures, mean time to repair), the escalation procedure, and penalties are usually documented in contractual agreements (e.g., as described in Section 2.1 of [RFC4176]). Misbehaving peer providers will thus be identified and appropriate countermeasures will be applied.

The communication protocols that make use of a service model between a customer and an operator are out of scope. Relevant security considerations should be discussed in the specification documents of these protocols.

6.2. Network Level

Security considerations specific to the network level are listed below:

- o A controller may create forwarding loops by mis-configuring the underlying network nodes. It is recommended to proceed with tests to check the status of forwarding paths regularly or whenever changes are made to routing or forwarding processes. Such checks may be triggered from the service level owing to the means discussed in Section 4.1.5.
- o Some service models may include a traffic isolation clause that is passed down to the network level so that appropriate technology-specific actions must be enforced at the underlying network (and thus involved network devices) to avoid that such traffic is accessible to non-authorized parties. In particular, network models may indicate whether encryption is enabled and if so, expose a list of supported encryption schemes and parameters. Refer for example to the encryption feature defined in [I-D.ietf-opsawg-vpn-common] and its use in [I-D.ietf-opsawg-l3sm-l3nm].

6.3. Device Level

Network operators should monitor and audit their networks to detect misbehaving nodes and abnormal behaviors. For example, OAM discussed in Section 4.1.5 can be used for that purpose.

Access to some data requires specific access privilege levels. Devices must check that a required access privilege is provided before granting access to specific data or performing specific actions.

7. IANA Considerations

There are no IANA requests or assignments included in this document.

8. Acknowledgements

Thanks to Joe Clark, Greg Mirsky, Shunsuke Homma, Brian Carpenter, Adrian Farrel, Christian Huitema, Tommy Pauly, Ines Robles, and Olivier Augizeau for the review.

Many thanks to Robert Wilton for the detailed AD review.

Thanks to Eric Vyncke, Roman Danyliw, Erik Kline, and Benjamin Kaduk for the IESG review.

9. Contributors

Christian Jacquenet
Orange
Rennes, 35000
France
Email: Christian.jacquenet@orange.com

Luis Miguel Contreras Murillo
Telefonica

Email: luismiguel.contrerasmurillo@telefonica.com

Oscar Gonzalez de Dios
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Weiqiang Cheng
China Mobile

Email: chengweiqiang@chinamobile.com

Young Lee
Sung Kyun Kwan University

Email: younglee.tx@gmail.com

10. References

10.1. Normative References

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

10.2. Informative References

- [Hardy] Hardy, N., "The Confused Deputy: (or why capabilities might have been invented)", October 1988, <<https://dl.acm.org/doi/10.1145/54289.871709>>.
- [I-D.clacla-netmod-model-catalog] Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-clacla-netmod-model-catalog-03 (work in progress), April 2018.
- [I-D.ietf-bess-evpn-yang] Brissette, P., Shah, H., Hussain, I., Tiruveedhula, K., and J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-07 (work in progress), March 2019.
- [I-D.ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.
- [I-D.ietf-bess-l3vpn-yang] Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-04 (work in progress), October 2018.

- [I-D.ietf-bess-mvpn-yang]
Liu, Y., Guo, F., Litkowski, S., Liu, X., Kebler, R., and M. Sivakumar, "Yang Data Model for Multicast in MPLS/BGP IP VPNs", draft-ietf-bess-mvpn-yang-04 (work in progress), June 2020.
- [I-D.ietf-bfd-yang]
Rahman, R., Zheng, L., Jethanandani, M., Pallagatti, S., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-17 (work in progress), August 2018.
- [I-D.ietf-dots-rfc8782-bis]
Boucadair, M., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-rfc8782-bis-01 (work in progress), September 2020.
- [I-D.ietf-i2rs-yang-l2-network-topology]
Dong, J., Wei, X., WU, Q., Boucadair, M., and A. Liu, "A YANG Data Model for Layer 2 Network Topologies", draft-ietf-i2rs-yang-l2-network-topology-18 (work in progress), September 2020.
- [I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-09 (work in progress), June 2020.
- [I-D.ietf-ippm-capacity-metric-method]
Morton, A., Geib, R., and L. Ciavattone, "Metrics and Methods for One-way IP Capacity", draft-ietf-ippm-capacity-metric-method-04 (work in progress), September 2020.
- [I-D.ietf-ippm-stamp-yang]
Mirsky, G., Min, X., and W. Luo, "Simple Two-way Active Measurement Protocol (STAMP) Data Model", draft-ietf-ippm-stamp-yang-06 (work in progress), October 2020.
- [I-D.ietf-ippm-twamp-yang]
Civil, R., Morton, A., Rahman, R., Jethanandani, M., and K. Pentikousis, "Two-Way Active Measurement Protocol (TWAMP) Data Model", draft-ietf-ippm-twamp-yang-13 (work in progress), July 2018.

- [I-D.ietf-mpls-base-yang]
Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A YANG Data Model for MPLS Base", draft-ietf-mpls-base-yang-16 (work in progress), October 2020.
- [I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-10 (work in progress), February 2020.
- [I-D.ietf-opsawg-l2nm]
barguil, s., Dios, O., Boucadair, M., Munoz, L., Jalil, L., and J. Ma, "A Layer 2 VPN Network YANG Model", draft-ietf-opsawg-l2nm-00 (work in progress), July 2020.
- [I-D.ietf-opsawg-l3sm-l3nm]
barguil, s., Dios, O., Boucadair, M., Munoz, L., and A. Aguado, "A Layer 3 VPN Network YANG Model", draft-ietf-opsawg-l3sm-l3nm-05 (work in progress), October 2020.
- [I-D.ietf-opsawg-vpn-common]
barguil, s., Dios, O., Boucadair, M., and Q. WU, "A Layer 2/3 VPN Common YANG Model", draft-ietf-opsawg-vpn-common-01 (work in progress), September 2020.
- [I-D.ietf-pim-igmp-mld-snooping-yang]
Zhao, H., Liu, X., Liu, Y., Sivakumar, M., and A. Peter, "A Yang Data Model for IGMP and MLD Snooping", draft-ietf-pim-igmp-mld-snooping-yang-18 (work in progress), August 2020.
- [I-D.ietf-pim-yang]
Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG Data Model for Protocol Independent Multicast (PIM)", draft-ietf-pim-yang-17 (work in progress), May 2018.
- [I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", draft-ietf-rtgwg-policy-model-26 (work in progress), October 2020.
- [I-D.ietf-rtgwg-qos-model]
Choudhary, A., Jethanandani, M., Strahle, N., Aries, E., and I. Chen, "YANG Model for QoS", draft-ietf-rtgwg-qos-model-02 (work in progress), July 2020.

[I-D.ietf-spring-sr-yang]

Litkowski, S., Qu, Y., Lindem, A., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-22 (work in progress), August 2020.

[I-D.ietf-teas-actn-pm-telemetry-autonomics]

Lee, Y., Dhody, D., Karunanithi, S., Vilata, R., King, D., and D. Ceccarelli, "YANG models for VN/TE Performance Monitoring Telemetry and Scaling Intent Autonomics", draft-ietf-teas-actn-pm-telemetry-autonomics-03 (work in progress), July 2020.

[I-D.ietf-teas-actn-vn-yang]

Lee, Y., Dhody, D., Ceccarelli, D., Bryskin, I., and B. Yoon, "A YANG Data Model for VN Operation", draft-ietf-teas-actn-vn-yang-09 (work in progress), July 2020.

[I-D.ietf-teas-yang-path-computation]

Busi, I., Belotti, S., Lopez, V., Sharma, A., and Y. Shi, "Yang model for requesting Path Computation", draft-ietf-teas-yang-path-computation-10 (work in progress), July 2020.

[I-D.ietf-teas-yang-rsvp-te]

Beeram, V., Saad, T., Gandhi, R., Liu, X., Bryskin, I., and H. Shah, "A YANG Data Model for RSVP-TE Protocol", draft-ietf-teas-yang-rsvp-te-08 (work in progress), March 2020.

[I-D.ietf-teas-yang-te]

Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths and Interfaces", draft-ietf-teas-yang-te-25 (work in progress), July 2020.

[I-D.ietf-trill-yang-oam]

Kumar, D., Senevirathne, T., Finn, N., Salam, S., Xia, L., and H. Weiguo, "YANG Data Model for TRILL Operations, Administration, and Maintenance (OAM)", draft-ietf-trill-yang-oam-05 (work in progress), March 2017.

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

Dios, O., barguil, s., WU, Q., and M. Boucadair, "A YANG Model for User-Network Interface (UNI) Topologies", draft-ogondio-opsawg-uni-topology-01 (work in progress), April 2020.

- [I-D.www-opsawg-yang-vpn-service-pm]
Bo, W., WU, Q., Boucadair, M., Dios, O., Wen, B., Liu, C.,
and H. Xu, "A YANG Model for Network and VPN Service
Performance Monitoring", draft-www-opsawg-yang-vpn-
service-pm-01 (work in progress), July 2020.
- [I-D.www-netmod-event-yang]
Bierman, A., WU, Q., Bryskin, I., Birkholz, H., Liu, X.,
and B. Claise, "A YANG Data model for ECA Policy
Management", draft-www-netmod-event-yang-09 (work in
progress), July 2020.
- [IPPM] IANA, "Performance Metrics", March 2020,
<[https://www.iana.org/assignments/performance-metrics/
performance-metrics.xhtml](https://www.iana.org/assignments/performance-metrics/performance-metrics.xhtml)>.
- [RFC4176] El Mghazli, Y., Ed., Nadeau, T., Boucadair, M., Chan, K.,
and A. Gonguet, "Framework for Layer 3 Virtual Private
Networks (L3VPN) Operations and Management", RFC 4176,
DOI 10.17487/RFC4176, October 2005,
<<https://www.rfc-editor.org/info/rfc4176>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February
2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer
2 Virtual Private Networks (L2VPNs)", RFC 4664,
DOI 10.17487/RFC4664, September 2006,
<<https://www.rfc-editor.org/info/rfc4664>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private
LAN Service (VPLS) Using BGP for Auto-Discovery and
Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007,
<<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private
LAN Service (VPLS) Using Label Distribution Protocol (LDP)
Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007,
<<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC5136] Chimento, P. and J. Ishac, "Defining Network Capacity",
RFC 5136, DOI 10.17487/RFC5136, February 2008,
<<https://www.rfc-editor.org/info/rfc5136>>.

- [RFC5486] Malas, D., Ed. and D. Meyer, Ed., "Session Peering for Multimedia Interconnect (SPEERMINT) Terminology", RFC 5486, DOI 10.17487/RFC5486, March 2009, <<https://www.rfc-editor.org/info/rfc5486>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC6406] Malas, D., Ed. and J. Livingood, Ed., "Session PEERing for Multimedia INTERconnect (SPEERMINT) Architecture", RFC 6406, DOI 10.17487/RFC6406, November 2011, <<https://www.rfc-editor.org/info/rfc6406>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <<https://www.rfc-editor.org/info/rfc7276>>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <<https://www.rfc-editor.org/info/rfc7297>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<https://www.rfc-editor.org/info/rfc7317>>.
- [RFC7455] Senevirathne, T., Finn, N., Salam, S., Kumar, D., Eastlake 3rd, D., Aldrin, S., and Y. Li, "Transparent Interconnection of Lots of Links (TRILL): Fault Management", RFC 7455, DOI 10.17487/RFC7455, March 2015, <<https://www.rfc-editor.org/info/rfc7455>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.

- [RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <<https://www.rfc-editor.org/info/rfc7679>>.
- [RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <<https://www.rfc-editor.org/info/rfc7680>>.
- [RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <<https://www.rfc-editor.org/info/rfc8077>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.

- [RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <<https://www.rfc-editor.org/info/rfc8346>>.
- [RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <<https://www.rfc-editor.org/info/rfc8348>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.
- [RFC8513] Boucadair, M., Jacquenet, C., and S. Sivakumar, "A YANG Data Model for Dual-Stack Lite (DS-Lite)", RFC 8513, DOI 10.17487/RFC8513, January 2019, <<https://www.rfc-editor.org/info/rfc8513>>.
- [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <<https://www.rfc-editor.org/info/rfc8519>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.
- [RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <<https://www.rfc-editor.org/info/rfc8528>>.

- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.
- [RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <<https://www.rfc-editor.org/info/rfc8530>>.
- [RFC8531] Kumar, D., Wu, Q., and Z. Wang, "Generic YANG Data Model for Connection-Oriented Operations, Administration, and Maintenance (OAM) Protocols", RFC 8531, DOI 10.17487/RFC8531, April 2019, <<https://www.rfc-editor.org/info/rfc8531>>.
- [RFC8532] Kumar, D., Wang, Z., Wu, Q., Ed., Rahman, R., and S. Raghavan, "Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8532, DOI 10.17487/RFC8532, April 2019, <<https://www.rfc-editor.org/info/rfc8532>>.
- [RFC8533] Kumar, D., Wang, M., Wu, Q., Ed., Rahman, R., and S. Raghavan, "A YANG Data Model for Retrieval Methods for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8533, DOI 10.17487/RFC8533, April 2019, <<https://www.rfc-editor.org/info/rfc8533>>.
- [RFC8632] Vallin, S. and M. Bjorklund, "A YANG Data Model for Alarm Management", RFC 8632, DOI 10.17487/RFC8632, September 2019, <<https://www.rfc-editor.org/info/rfc8632>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8652] Liu, X., Guo, F., Sivakumar, M., McAllister, P., and A. Peter, "A YANG Data Model for the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", RFC 8652, DOI 10.17487/RFC8652, November 2019, <<https://www.rfc-editor.org/info/rfc8652>>.
- [RFC8675] Boucadair, M., Farrer, I., and R. Asati, "A YANG Data Model for Tunnel Interface Types", RFC 8675, DOI 10.17487/RFC8675, November 2019, <<https://www.rfc-editor.org/info/rfc8675>>.

- [RFC8676] Farrer, I., Ed. and M. Boucadair, Ed., "YANG Modules for IPv4-in-IPv6 Address plus Port (A+P) Softwires", RFC 8676, DOI 10.17487/RFC8676, November 2019, <<https://www.rfc-editor.org/info/rfc8676>>.
- [RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", RFC 8783, DOI 10.17487/RFC8783, May 2020, <<https://www.rfc-editor.org/info/rfc8783>>.
- [RFC8791] Bierman, A., Bjoerklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <<https://www.rfc-editor.org/info/rfc8791>>.
- [RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Gonzalez de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", RFC 8795, DOI 10.17487/RFC8795, August 2020, <<https://www.rfc-editor.org/info/rfc8795>>.

Appendix A. Layered YANG Modules Examples Overview

This appendix lists a set of YANG data models that can be used for the delivery of connectivity services. These models can be classified as service, network, or device models.

It is not the intent of this appendix to provide an inventory of tools and mechanisms used in specific network and service management domains; such inventory can be found in documents such as [RFC7276].

The reader may refer to the YANG Catalog (<<https://www.yangcatalog.org>>) or the public Github YANG repository (<<https://github.com/YangModels/yang>>) to query existing YANG models. The YANG Catalog includes some metadata to indicate the module type ('module-classification') [I-D.clacla-netmod-model-catalog]. Note that the mechanism defined in [I-D.ietf-netmod-module-tags] allows to associate tags with YANG modules in order to help classifying the modules.

A.1. Service Models: Definition and Samples

As described in [RFC8309], the service is "some form of connectivity between customer sites and the Internet and/or between customer sites across the network operator's network and across the Internet". More concretely, an IP connectivity service can be defined as the IP transfer capability characterized by a (Source Nets, Destination Nets, Guarantees, Scope) tuple where "Source Nets" is a group of unicast IP addresses, "Destination Nets" is a group of IP unicast

and/or multicast addresses, and "Guarantees" reflects the guarantees (expressed in terms of QoS, performance, and availability, for example) to properly forward traffic to the said "Destination" [RFC7297].

For example:

- o The L3SM [RFC8299] defines the L3VPN service ordered by a customer from a network operator.
- o The L2SM [RFC8466] defines the L2VPN service ordered by a customer from a network operator.
- o The Virtual Network (VN) model [I-D.ietf-teas-actn-vn-yang] provides a YANG data model applicable to any mode of VN operation.

L2SM and L3SM are customer service models as per [RFC8309].

A.2. Schema Mount

Modularity and extensibility were among the leading design principles of the YANG data modeling language. As a result, the same YANG module can be combined with various sets of other modules and thus form a data model that is tailored to meet the requirements of a specific use case. [RFC8528] defines a mechanism, denoted schema mount, that allows for mounting one data model consisting of any number of YANG modules at a specified location of another (parent) schema.

A.3. Network Models: Samples

L2NM [I-D.ietf-opsawg-l2nm] and L3NM [I-D.ietf-opsawg-l3sm-l3nm] are examples of YANG network models.

Figure 9 depicts a set of additional network models such as topology and tunnel models:

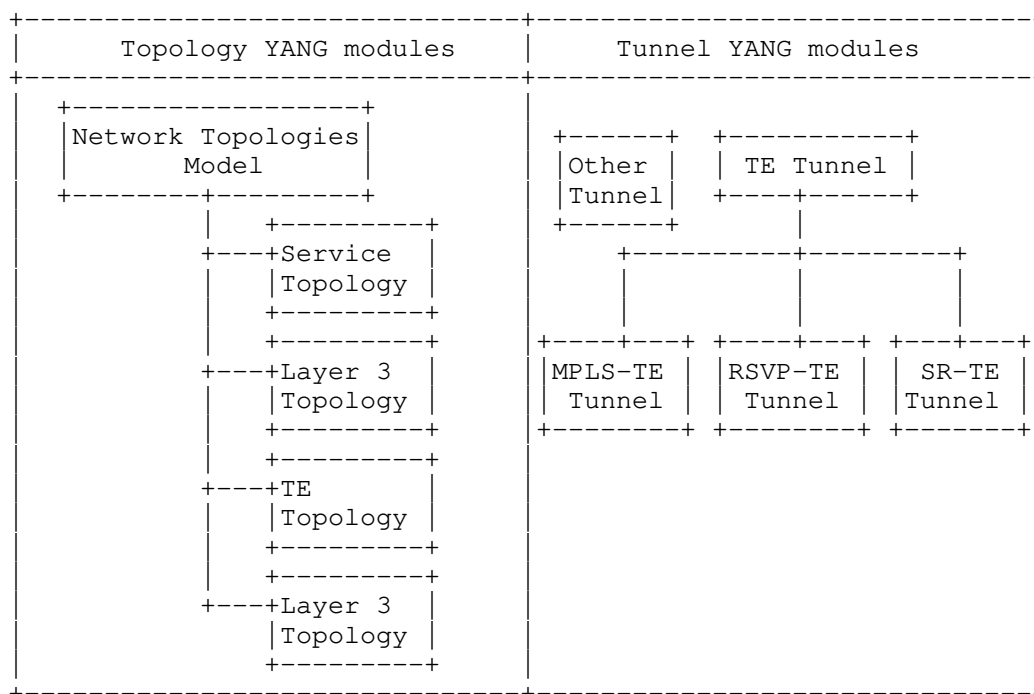


Figure 9: Sample Resource Facing Network Models

Examples of topology YANG modules are listed below:

- o Network Topologies Model: [RFC8345] defines a base model for network topology and inventories. Network topology data include link, node, and terminate-point resources.
- o TE Topology Model: [RFC8795] defines a YANG data model for representing and manipulating TE topologies.

This module is extended from network topology model defined in [RFC8345] with TE topologies related content. This model contains technology-agnostic TE Topology building blocks that can be augmented and used by other technology-specific TE topology models.

- o Layer 3 Topology Model:

[RFC8346] defines a YANG data model for representing and manipulating Layer 3 topologies. This model is extended from the network topology model defined in [RFC8345] with Layer 3 topologies specifics.

- o Layer 2 Topology Model:

[I-D.ietf-i2rs-yang-l2-network-topology] defines a YANG data model for representing and manipulating Layer 2 topologies. This model is extended from the network topology model defined in [RFC8345] with Layer 2 topology specifics.

Examples of tunnel YANG modules are provided below:

- o Tunnel identities: [RFC8675] defines a collection of YANG identities used as interface types for tunnel interfaces.

- o TE Tunnel Model:

[I-D.ietf-teas-yang-te] defines a YANG module for the configuration and management of TE interfaces, tunnels, and LSPs.

- o Segment Routing (SR) Traffic Engineering (TE) Tunnel Model:

[I-D.ietf-teas-yang-te] augments the TE generic and MPLS-TE model(s) and defines a YANG module for SR-TE specific data.

- o MPLS-TE Model:

[I-D.ietf-teas-yang-te] augments the TE generic and MPLS-TE model(s) and defines a YANG module for MPLS-TE configurations, state, RPC and notifications.

- o RSVP-TE MPLS Model:

[I-D.ietf-teas-yang-rsvp-te] augments the RSVP-TE generic module with parameters to configure and manage signaling of MPLS RSVP-TE LSPs.

Other sample network models are listed hereafter:

- o Path Computation API Model:

[I-D.ietf-teas-yang-path-computation] YANG module for a stateless RPC which complements the stateful solution defined in [I-D.ietf-teas-yang-te].

- o OAM Models (including Fault Management (FM) and Performance Monitoring):

[RFC8532] defines a base YANG module for the management of OAM protocols that use Connectionless Communications. [RFC8533] defines a retrieval method YANG module for connectionless OAM

protocols. [RFC8531] defines a base YANG module for connection oriented OAM protocols. These three models are intended to provide consistent reporting, configuration, and representation for connection-less OAM and Connection oriented OAM separately.

Alarm monitoring is a fundamental part of monitoring the network. Raw alarms from devices do not always tell the status of the network services or necessarily point to the root cause. [RFC8632] defines a YANG module for alarm management.

A.4. Device Models: Samples

Network Element models (listed in Figure 10) are used to describe how a service can be implemented by activating and tweaking a set of functions (enabled in one or multiple devices, or hosted in cloud infrastructures) that are involved in the service delivery. For example, the L3VPN service will involve many PEs and require manipulating the following modules:

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

Figure 10 uses IETF-defined data models as an example.

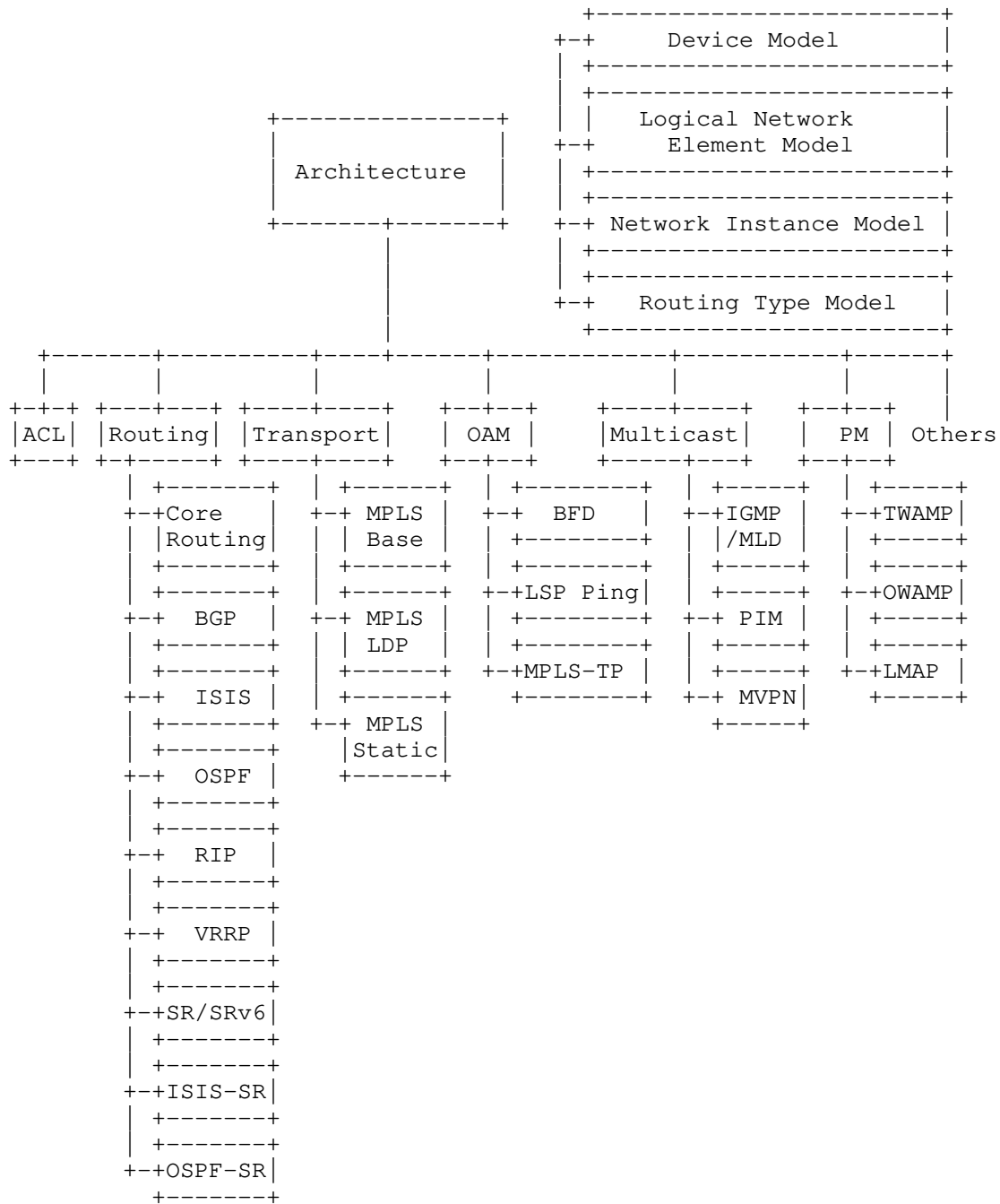


Figure 10: Network Element Modules Overview

A.4.1. Model Composition

- o Logical Network Element Model

[RFC8530] defines a logical network element module which can be used to manage the logical resource partitioning that may be present on a network device. Examples of common industry terms for logical resource partitioning are Logical Systems or Logical Routers.

- o Network Instance Model

[RFC8529] defines a network instance module. This module can be used to manage the virtual resource partitioning that may be present on a network device. Examples of common industry terms for virtual resource partitioning are VRF instances and Virtual Switch Instances (VSIs).

A.4.2. Device Management

The following list enumerates some YANG modules that can be used for device management:

- o [RFC8348]: defines a YANG module for the management of hardware.
- o [RFC7317]: defines the "ietf-system" YANG module that provides many features such as the configuration and the monitoring of system or system control operations (e.g., shutdown, restart, setting time) identification.
- o [RFC8341]: defines a network configuration access control YANG module.

A.4.3. Interface Management

The following provides some YANG modules that can be used for interface management:

- o [RFC7224]: defines a YANG module for interface type definitions.
- o [RFC8343]: defines a YANG module for the management of network interfaces.

A.4.4. Some Device Model Examples

The following provides an overview of some device models that can be used within a network. This list is not comprehensive.

- L2VPN:** [I-D.ietf-bess-l2vpn-yang] defines a YANG module for MPLS based Layer 2 VPN services (L2VPN) [RFC4664] and includes switching between the local attachment circuits. The L2VPN model covers point-to-point VPWS and Multipoint VPLS services. These services use signaling of Pseudowires across MPLS networks using LDP [RFC8077][RFC4762] or BGP [RFC4761].
- EVPN:** [I-D.ietf-bess-evpn-yang] defines a YANG module for Ethernet VPN services. The model is agnostic of the underlay. It applies to MPLS as well as to VxLAN encapsulation. The module is also agnostic to the services, including E-LAN, E-LINE, and E-TREE services.
- L3VPN:** [I-D.ietf-bess-l3vpn-yang] defines a YANG module that can be used to configure and manage BGP L3VPNs [RFC4364]. It contains VRF specific parameters as well as BGP specific parameters applicable for L3VPNs.
- Core Routing:** [RFC8349] defines the core routing YANG data model, which is intended as a basis for future data model development covering more-sophisticated routing systems. It is expected that other Routing technology YANG modules (e.g., VRRP, RIP, ISIS, OSPF models) will augment the Core Routing base YANG module.
- MPLS:** [I-D.ietf-mpls-base-yang] defines a base model for MPLS which serves as a base framework for configuring and managing an MPLS switching subsystem. It is expected that other MPLS technology YANG modules (e.g., MPLS LSP Static, LDP, or RSVP-TE models) will augment the MPLS base YANG module.
- BGP:** [I-D.ietf-idr-bgp-model] defines a YANG module for configuring and managing BGP, including protocol, policy, and operational aspects based on data center, carrier, and content provider operational requirements.
- Routing Policy:** [I-D.ietf-rtgwg-policy-model] defines a YANG module for configuring and managing routing policies based on operational practice. The module provides a generic policy framework which can be augmented with protocol-specific policy configuration.
- SR/SRv6:** [I-D.ietf-spring-sr-yang] a YANG module for segment routing configuration and operation.

- BFD:** Bidirectional Forwarding Detection (BFD) [RFC5880] is a network protocol which is used for liveness detection of arbitrary paths between systems. [I-D.ietf-bfd-yang] defines a YANG module that can be used to configure and manage BFD.
- Multicast:** [I-D.ietf-pim-yang] defines a YANG module that can be used to configure and manage Protocol Independent Multicast (PIM) devices.
- [RFC8652] defines a YANG module that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices.
- [I-D.ietf-pim-igmp-mld-snooping-yang] defines a YANG module that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping devices.
- [I-D.ietf-bess-mvpn-yang] defines a YANG data model to configure and manage Multicast in MPLS/BGP IP VPNs (MVPNs).
- PM:** [I-D.ietf-ippm-twamp-yang] defines a YANG data model for client and server implementations of the Two-Way Active Measurement Protocol (TWAMP).
- [I-D.ietf-ippm-stamp-yang] defines the data model for implementations of Session-Sender and Session-Reflector for Simple Two-way Active Measurement Protocol (STAMP) mode using YANG.
- [RFC8194] defines a YANG data model for Large-Scale Measurement Platforms (LMAPs).
- ACL:** Access Control List (ACL) is one of the basic elements used to configure device forwarding behavior. It is used in many networking technologies such as Policy Based Routing, firewalls, etc. [RFC8519] describes a YANG data model of ACL basic building blocks.
- QoS:** [I-D.ietf-rtgwg-qos-model] describes a YANG module of Differentiated Services for configuration and operations.
- NAT:** For the sake of network automation and the need for programming Network Address Translation (NAT) function in particular, a YANG data model for configuring and managing the NAT is essential.

[RFC8512] defines a YANG module for the NAT function covering a variety of NAT flavors such as Network Address Translation from IPv4 to IPv4 (NAT44), Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers (NAT64), customer-side translator (CLAT), Stateless IP/ICMP Translation (SIIT), Explicit Address Mappings (EAM) for SIIT, IPv6-to-IPv6 Network Prefix Translation (NPTv6), and Destination NAT.

[RFC8513] specifies a DS-Lite YANG module.

Stateless Address Sharing: [RFC8676] specifies a YANG module for A+P address sharing, including Lightweight 4over6, Mapping of Address and Port with Encapsulation (MAP-E), and Mapping of Address and Port using Translation (MAP-T) software mechanisms.

Authors' Addresses

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

Email: bill.wu@huawei.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Diego R. Lopez
Telefonica I+D
Spain

Email: diego.r.lopez@telefonica.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@chinatelecom.cn

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Opsawg
Internet-Draft
Intended status: Standards Track
Expires: November 14, 2021

B. Wu, Ed.
G. Zheng
M. Wang, Ed.
Huawei
May 13, 2021

A YANG Module for TACACS+
draft-ietf-opsawg-tacacs-yang-12

Abstract

This document defines a Terminal Access Controller Access-Control System Plus (TACACS+) client YANG module, that augments the System Management data model, defined in RFC 7317, to allow devices to make use of TACACS+ servers for centralized Authentication, Authorization and Accounting (AAA). Though being a standard module, this module does not endorse the security mechanisms of the TACACS+ protocol (RFC 8907) and TACACS+ MUST be used within a secure deployment.

The YANG module in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 14, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions used in this document	3
2.1. Tree Diagrams	3
3. Design of the TACACS+ Data Model	3
4. TACACS+ Client Module	5
5. Security Considerations	12
6. IANA Considerations	13
7. Acknowledgments	13
8. References	13
8.1. Normative References	13
8.2. Informative References	15
Appendix A. Example TACACS+ Authentication Configuration	15
Authors' Addresses	16

1. Introduction

This document defines a YANG module that augments the System Management data model defined in the [RFC7317] to support the configuration and management of TACACS+ clients.

TACACS+ [RFC8907] provides device administration for routers, network access servers and other networked devices via one or more centralized servers.

The System Management Model [RFC7317] defines separate functionality to support local and RADIUS authentication:

- o User Authentication Model: Defines a list of usernames with associated passwords and a configuration leaf to decide the order in which local or RADIUS authentication is used.
- o RADIUS Client Model: Defines a list of RADIUS servers used by a device for centralized user authentication.

The System Management Model is augmented with the TACACS+ YANG module defined in this document to allow the use of TACACS+ servers as an alternative to RADIUS servers.

The YANG module can be used with network management protocols such as NETCONF[RFC6241].

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

2. Conventions used in this document

The keywords "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 BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

- o configuration data
- o state data

The following terms are defined in [RFC7950] and are used in this specification:

- o augment
- o data model
- o data node

The terminology for describing YANG data models is found in [RFC7950].

2.1. Tree Diagrams

The tree diagram used in this document follows the notation defined in [RFC8340].

3. Design of the TACACS+ Data Model

This module is used to configure a TACACS+ client on a device to support deployment scenarios with centralized authentication, authorization, and accounting servers. Authentication is used to validate a user's username and password, authorization allows the user to access and execute commands at various privilege levels assigned to the user, and accounting keeps track of the activity of a user who has accessed the device.

The ietf-system-tacacs-plus module augments the `"/sys:system"` path defined in the ietf-system module with the contents of the `"tacacs-plus"` grouping. Therefore, a device can use local, RADIUS, or TACACS+ to validate users who attempt to access the router by several mechanisms, e.g., a command line interface or a web-based user interface.

The `"server"` list is directly under the `"tacacs-plus"` container, which holds a list of TACACS+ servers and uses `server-type` to distinguish between Authentication, Authorization and Accounting (AAA). The list of servers is for redundancy.

Most of the parameters in the `"server"` list are taken directly from the TACACS+ protocol [RFC8907], and some are derived from the various implementations by network equipment manufacturers. For example, when there are multiple interfaces connected to the TACACS+ client or server, the source address of outgoing TACACS+ packets could be specified, or the source address could be specified through the interface IP address setting, or derived from the outbound interface from the local Forwarding Information Base (FIB). For the TACACS+ server located in a Virtual Private Network (VPN), a VPN Routing and Forwarding (VRF) instance needs to be specified.

The `"statistics"` container under the `"server list"` is a collection of read-only counters for sent and received messages from a configured server.

The YANG module for TACACS+ client has the following structure:


```

module: ietf-system-tacacs-plus
augment /sys:system:
  +--rw tacacs-plus
    +--rw server* [name]
      +--rw name string
      +--rw server-type tacacs-plus-server-type
      +--rw address inet:host
      +--rw port? inet:port-number
      +--rw (security)
        | +--:(obfuscation)
        |   +--rw shared-secret? string
      +--rw (source-type)?
        | +--:(source-ip)
        |   | +--rw source-ip? inet:ip-address
        |   +--:(source-interface)
        |     +--rw source-interface? if:interface-ref
      +--rw vrf-instance?
        |   -> /ni:network-instances/network-instance/name
      +--rw single-connection? boolean
      +--rw timeout? uint16
      +--ro statistics
        +--ro connection-opens? yang:counter64
        +--ro connection-closes? yang:counter64
        +--ro connection-aborts? yang:counter64
        +--ro connection-failures? yang:counter64
        +--ro connection-timeouts? yang:counter64
        +--ro messages-sent? yang:counter64
        +--ro messages-received? yang:counter64
        +--ro errors-received? yang:counter64
        +--ro sessions? yang:counter64

```

4. TACACS+ Client Module

This YANG module imports typedefs from [RFC6991]. This module also uses the interface typedef from [RFC8343], the leafref to VRF instance from [RFC8529], and the "default-deny-all" extension statement from [RFC8341].

<CODE BEGINS> file "ietf-system-tacacs-plus@2021-05-13.yang"

```

module ietf-system-tacacs-plus {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-system-tacacs-plus";
  prefix sys-tcs-plus;

  import ietf-inet-types {
    prefix inet;

```

```
reference
  "RFC 6991: Common YANG Data Types";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types";
}
import ietf-network-instance {
  prefix ni;
  reference
    "RFC 8529: YANG Data Model for Network Instances";
}
import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343: A YANG Data Model for Interface Management";
}
import ietf-system {
  prefix sys;
  reference
    "RFC 7317: A YANG Data Model for System Management";
}
import ietf-netconf-acm {
  prefix nacm;
  reference
    "RFC 8341: Network Configuration Access Control Model";
}

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

   Editor: Bo Wu <lane.wubo@huawei.com>
   Editor: Guangying Zheng <zhengguangying@huawei.com>";
description
  "This module provides configuration of TACACS+ client.

  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.

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

```
// RFC Ed.: update the date below with the date of RFC
// publication and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note.
```

```
revision 2021-05-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Module for TACACS+";
}

typedef tacacs-plus-server-type {
  type bits {
    bit authentication {
      description
        "Indicates that the TACACS+ server is providing authentication
        services.";
    }
    bit authorization {
      description
        "Indicates that the TACACS+ server is providing authorization
        services.";
    }
    bit accounting {
      description
        "Indicates that the TACACS+ server is providing accounting
        services.";
    }
  }
  description
    "tacacs-plus-server-type can be set to
    authentication/authorization/accounting
    or any combination of the three types.";
}

identity tacacs-plus {
```

```
base sys:authentication-method;
description
  "Indicates AAA operation using TACACS+.";
reference
  "RFC 8907: The TACACS+ Protocol";
}

grouping statistics {
  description
    "Grouping for TACACS+ statistics attributes";
  container statistics {
    config false;
    description
      "A collection of server-related statistics objects";
    leaf connection-opens {
      type yang:counter64;
      description
        "Number of new connection requests sent to the server, e.g.,
        socket open";
    }
    leaf connection-closes {
      type yang:counter64;
      description
        "Number of connection close requests sent to the server, e.g.,
        socket close";
    }
    leaf connection-aborts {
      type yang:counter64;
      description
        "Number of aborted connections to the server. These do
        not include connections that are closed gracefully.";
    }
    leaf connection-failures {
      type yang:counter64;
      description
        "Number of connection failures to the server";
    }
    leaf connection-timeouts {
      type yang:counter64;
      description
        "Number of connection timeouts to the server";
    }
    leaf messages-sent {
      type yang:counter64;
      description
        "Number of messages sent to the server";
    }
    leaf messages-received {
```

```
    type yang:counter64;
    description
        "Number of messages received from the server";
}
leaf errors-received {
    type yang:counter64;
    description
        "Number of error messages received from the server";
}
leaf sessions {
    type yang:counter64;
    description
        "Number of TACACS+ sessions completed with the server.
        If the Single Connection Mode was NOT enabled, the number of
        sessions is the same as the number of 'connection-closes'.
        If the Mode was enabled, a single TCP connection may contain
        multiple TACACS+ sessions.";
}
}
}

grouping tacacs-plus {
    description
        "Grouping for TACACS+ attributes";
    container tacacs-plus {
        must "not (derived-from-or-self(..sys:authentication"
            + "/sys:user-authentication-order, 'tacacs-plus'))"
            + " or bit-is-set(server/server-type, 'authentication') " {
            error-message "When 'tacacs-plus' is used as a system"
                + " authentication method, a TACACS+ authentication"
                + " server must be configured.";
        }
        description
            "When 'tacacs-plus' is used as an authentication method,
            a TACACS+ server must be configured.";
    }
    description
        "Container for TACACS+ configurations and operations.";
    list server {
        key "name";
        ordered-by user;
        description
            "List of TACACS+ servers used by the device.";
        leaf name {
            type string;
            description
                "An arbitrary name for the TACACS+ server.";
        }
        leaf server-type {
```

```
type tacacs-plus-server-type;
mandatory true;
description
  "Server type: authentication/authorization/accounting and
  various combinations.";
}
leaf address {
  type inet:host;
  mandatory true;
  description
    "The address of the TACACS+ server.";
}
leaf port {
  type inet:port-number;
  default "49";
  description
    "The port number of TACACS+ Server port.";
}
choice security {
  mandatory true;
  description
    "Security mechanism between TACACS+ client and server.
    This is modelled as a YANG 'choice' so that it can be
    augmented by a YANG module in a backwards compatible
    manner.";
  case obfuscation {
    leaf shared-secret {
      type string {
        length "1..max";
      }
      nacm:default-deny-all;
      description
        "The shared secret, which is known to both the
        TACACS+ client and server. TACACS+ server
        administrators SHOULD configure a shared secret of
        minimum 16 characters length.
        It is highly recommended that this shared secret is
        at least 32 characters long and sufficiently complex
        with a mix of different character types
        i.e. upper case, lower case, numeric, punctuation.
        Note that this security mechanism is best described as
        'obfuscation' and not 'encryption' as it does not
        provide any meaningful integrity, privacy, or replay
        protection.";
      reference
        "RFC 8907: The TACACS+ Protocol";
    }
  }
}
```

```
}
choice source-type {
  description
    "The source address type for outbound TACACS+ packets.";
  case source-ip {
    leaf source-ip {
      type inet:ip-address;
      description
        "Specifies source IP address for TACACS+ outbound
        packets.";
    }
  }
  case source-interface {
    leaf source-interface {
      type if:interface-ref;
      description
        "Specifies the interface from which the IP address is
        derived for use as the source for the outbound TACACS+
        packet";
    }
  }
}
leaf vrf-instance {
  type leafref {
    path "/ni:network-instances/ni:network-instance/ni:name";
  }
  description
    "Specifies the VPN Routing and Forwarding (VRF) instance to
    use to communicate with the TACACS+ server.";
  reference
    "RFC 8529: YANG Data Model for Network Instances";
}
leaf single-connection {
  type boolean;
  default "false";
  description
    "Whether the single connection mode is enabled for the
    server. By default, the single connection mode is
    disabled.";
}
leaf timeout {
  type uint16 {
    range "1..max";
  }
  units "seconds";
  default "5";
  description
    "The number of seconds the device will wait for a
```

```
        response from each TACACS+ server before trying with a
        different server.";
    }
    uses statistics;
}
}

augment "/sys:system" {
    description
        "Augment the system model with the tacacs-plus model";
    uses tacacs-plus;
}
}
```

<CODE ENDS>

5. Security Considerations

The YANG module defined in this document 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 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:

/system/tacacsplus/server: This list contains the data nodes used to control the TACACS+ servers used by the device. Unauthorized access to this list could enable an attacker to assume complete control over the device by pointing to a compromised TACACS+ server, or to modify the counters to hide attacks against the device.

/system/tacacsplus/server/shared-secret: This leaf controls the key known to both the TACACS+ client and server. Unauthorized access to this leaf could make the device vulnerable to attacks, therefore it has been restricted using the "default-deny-all" access control defined in [RFC8341]. When setting, it is highly recommended that the leaf is at least 32 characters long and sufficiently complex with a mix of different character types i.e. upper case, lower case, numeric, punctuation.

This document describes the use of TACACS+ for purposes of authentication, authorization and accounting, it is vulnerable to all of the threats that are present in TACACS+ applications. For a discussion of such threats, see Section 10 of the TACACS+ Protocol [RFC8907].

6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-system-tacacs-plus

Registrant Contact: The IESG.

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

This document registers a YANG module in the YANG Module Names registry [RFC7950].

Name: ietf-system-tacacs-plus

Namespace: urn:ietf:params:xml:ns:yang:ietf-system-tacacs-plus

Prefix: sys-tcs-plus

Reference: RFC XXXX (RFC Ed.: replace XXXX with actual RFC number and remove this note.)

7. Acknowledgments

The authors wish to thank Alex Campbell, John Heasley, Ebben Aries, Alan DeKok, Joe Clarke, Joe Clarke, Tom Petch, Robert Wilton, and many others for their helpful comments and suggestions.

8. References

8.1. Normative References

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

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<https://www.rfc-editor.org/info/rfc7317>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.
- [RFC8907] Dahm, T., Ota, A., Medway Gash, D., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <<https://www.rfc-editor.org/info/rfc8907>>.

8.2. Informative References

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

Appendix A. Example TACACS+ Authentication Configuration

The following shows an example where a TACACS+ authentication server instance is configured.

```
{
  "ietf-system:system": {
    "authentication": {
      "user-authentication-order": [tacacs-plus, local-users]
    }
    "tacacs-plus": {
      "server": [
        {
          "name": "tac_plus1",
          "server-type": "authentication",
          "address": "192.0.2.2",
          "shared-secret": "QaEfThUkO198010075460923+h3TbE8n",
          "source-ip": "192.0.2.12",
          "timeout": "10"
        }
      ]
    }
  }
}
```

Authors' Addresses

Bo Wu (editor)
Huawei Technologies, Co.,
Ltd
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: lana.wubo@huawei.com

Guangying Zheng
Huawei Technologies, Co.,
Ltd
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Michael Wang (editor)
Huawei Technologies, Co.,
Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

OPSWG WG
Internet-Draft
Intended status: Standards Track
Expires: March 3, 2021

T. Reddy
McAfee
D. Wing
Citrix
B. Anderson
Cisco
August 30, 2020

MUD (D)TLS profiles for IoT devices
draft-reddy-opsawg-mud-tls-05

Abstract

This memo extends Manufacturer Usage Description (MUD) to incorporate (D)TLS profile parameters. This allows a network element to identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software or malware on an endpoint.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 3, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Overview of MUD (D)TLS profiles for IoT devices	5
4. (D)TLS 1.3 handshake	5
4.1. Full (D)TLS 1.3 handshake inspection	6
4.2. Encrypted SNI	6
5. (D)TLS profile YANG module	7
5.1. Tree Structure	9
5.2. YANG Module	10
6. MUD File Example	15
7. Security Considerations	17
8. Privacy Considerations	17
9. IANA Considerations	17
10. Acknowledgments	17
11. References	17
11.1. Normative References	18
11.2. Informative References	19
Authors' Addresses	20

1. Introduction

Encryption is necessary to protect the privacy of end users using IoT devices. In a network setting, TLS [RFC8446] and DTLS [I-D.ietf-tls-dtls13] are the dominant protocols providing encryption for IoT device traffic. Unfortunately, in conjunction with IoT applications' rise of encryption, malware is also using encryption which thwarts network-based analysis such as deep packet inspection (DPI). Other mechanisms are needed to notice malware is running on the IoT device.

Malware frequently uses its own libraries for its activities, and those libraries are re-used much like any other software engineering project. Research [malware] indicates there are observable differences in how malware uses encryption compared with how non-malware uses encryption. There are several interesting findings specific to DTLS and TLS which were found common to malware:

- o Older and weaker cryptographic parameters (e.g., TLS_RSA_WITH_RC4_128_SHA).
- o TLS SNI and server certificates are composed of subjects with characteristics of a domain generation algorithm (DGA) (e.g., www.33mhw2j.net).

- o Higher use of self-signed certificates compared with typical legitimate software.
- o Discrepancies in the server name indication (SNI) TLS extension in the ClientHello message and the DNS names in the SubjectAltName(SAN) X.509 extension in the server certificate message.
- o Discrepancies in the key exchange algorithm and the client public key length in comparison with legitimate flows. As a reminder, Client Key Exchange message has been removed from TLS 1.3.
- o Lower diversity in TLS client advertised TLS extensions compared to legitimate clients.
- o Malware using privacy enhancing technologies like Tor, Psiphon and Ultrasurf (see [malware-tls]) and, evasion techniques such as ClientHello randomization to evade detection in order to continue exploiting the end user.
- o Malware using DNS-over-HTTPS [RFC8484] to avoid detection by malware DNS filtering service ([malware-doh]). Malware agent may not use the DNS-over-HTTPS server provided by the local network.

If observable (D)TLS profile parameters are used, the following functions are possible which have a favorable impact on network security:

- o Permit intended DTLS or TLS use and block malicious DTLS or TLS use. This is superior to the layer 3 and layer 4 ACLs of Manufacturer Usage Description Specification (MUD) [RFC8520] which are not suitable for broad communication patterns.
- o 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, however in TLS 1.3, the Certificate message is encrypted thereby hiding the server identity from any intermediary. In TLS 1.3, the middle-box needs to act as a TLS proxy to validate the server certificate and to detect TLS SNI mismatch with the server certificate.

- o 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. In such a case, observable (D)TLS profile parameters can be used to permit intended use and to block malicious behaviour from the IoT device.

This document extends MUD [RFC8520] to model observable (D)TLS profile parameters. Using these (D)TLS profile parameters, an active MUD-enforcing 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 on 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. Enterprise firewalls 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 agents have deep visibility on the devices where they are installed, whereas the network has broader visibility. Installing host agents may not be a viable option on IoT devices, and network-based security is an efficient means to protect such IoT devices. (D)TLS profile parameters of IoT device can be used by middle-boxes to detect and block malware communication, while at the same time preserving the privacy of legitimate uses of encryption. Middle-boxes need not proxy (D)TLS but can passively observe the parameters of (D)TLS handshakes from IoT devices and gain good visibility into TLS 1.0 to 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. Further, 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.

The compromised IoT devices are typically used for launching DDoS attacks (Section 3 of [RFC8576]). Some of the DDoS attacks like Slowloris and Transport Layer Security (TLS) re-negotiation can be detected by observing the (D)TLS profile parameters. For example, the victim's server certificate need 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 devices (such as a firewall) are incapable deciphering the handshake, 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 middle-box 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 middle-box can act as a (D)TLS 1.3 proxy. A middle-box 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 middle-box acting as a (D)TLS proxy is restricted to Enterprise network owning and managing the IoT devices. The middle-box MUST follow the behaviour explained in Section 9.3 of [RFC8446] to act as a compliant (D)TLS 1.3 proxy.

To function as a (D)TLS proxy the middle-box creates a signed certificate using itself as a certificate authority. That certificate authority has to be trusted by the (D)TLS client. The IoT device needs to be configured with the middle-box's CA certificate as Explicit Trust Anchor database entry to validate the server certificate. The mechanism to configure the IoT device with the middle-box's CA certificate is out of scope. The middle-box uses the "supported_versions" TLS extension (defined in TLS 1.3 to negotiate the supported TLS versions between client and server) to determine the TLS version. During the (D)TLS handshake, If (D)TLS version 1.3 is used, the middle-box ((D)TLS proxy) modifies the certificate provided by the server and signs it with the private key from the local CA certificate. The middle-box has visibility into further exchanges between the IoT device and server which enables it to inspect the (D)TLS 1.3 handshake, enforce the MUD (D)TLS profile and can inspect subsequent network traffic. The IoT device uses the Explicit Trust Anchor database to validate the server certificate.

4.2. Encrypted SNI

To increase privacy, encrypted SNI (ESNI, [I-D.ietf-tls-sni-encryption]) prevents passive observation of the TLS Server Name Indication extension. To effectively provide that privacy protection, SNI encryption needs to be used in conjunction with DNS encryption (e.g., DNS-over-TLS (DoT) [RFC7858] or DNS-over-HTTPS (DoH) [RFC8484]). A middle-box (e.g., firewall) passively inspecting an encrypted SNI (D)TLS handshake cannot observe the encrypted SNI nor observe the encrypted DNS traffic. If an IoT device is pre-configured to use public DoH/DoT servers, that middle-

box needs to act as a DoH/DoT proxy and replace the ECH configuration in the "echconfig" SvcParamKey (Section 6.3 of [I-D.ietf-dnsop-svcb-https]) with the middle box's ECH configuration. Instead of an unappealing DoH/DoT proxy, the IoT device can be bootstrapped to discover and authenticate DoH/DoT servers provided by a local network by making use of one of the mechanisms described in Section 4 of [I-D.reddy-add-enterprise]. The local DoH/DoT server replaces the ECH configuration in the "echconfig" SvcParamKey with the middle box's ECH configuration.

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 public DoH/DoT server, the MUD policy enforcement point is moved to that public server, 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 public DoH/DoT server is incompatible with MUD in general. A local DoH/DoT server is necessary to allow MUD policy enforcement on the local network.

5. (D)TLS profile YANG module

This document specifies a YANG module for representing (D)TLS profile. The (D)TLS profile YANG module provides a method for firewall 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 common YANG types defined in [RFC6991], rules defined in [RFC8519] and cryptographic types defined in [I-D.ietf-netconf-crypto-types].

The (D)TLS profiles and the parameters in each (D)TLS profile include the following:

- o Profile name
- o (D)TLS version in ClientHello.legacy_version
- o (D)TLS versions supported by the IoT device. As a reminder, "supported_versions" extension defined in (D)TLS 1.3 is used by the client to indicate which versions of (D)TLS it supports and a client is considered to be attempting to negotiate (D)TLS 1.3 if the ClientHello contains a "supported_versions" extension with 0x0304 contained in its body.

- o If GREASE [RFC8701] (Generate Random Extensions And Sustain Extensibility) values are offered by the client or not.
- o List of supported symmetric encryption algorithms. TLS 1.3 defines five cipher suites (Appendix B.4 of [RFC8446]), but most clients are continuing to offer TLS 1.2 compatible cipher suites for backwards compatibility.
- o List of supported compression methods for data compression. In TLS 1.3, only the "null" compression method is allowed (Section 4.1.2 of [RFC8446]).
- o List of supported extension types
- o List of trust anchor certificates used by the IoT device. If the server certificate is signed by one of the trust anchors, the middle-box continues with the connection as normal. Otherwise, the middle-box 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).
- o 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 middle-box 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 middle-box continues with the connection as normal. Otherwise, the middle-box will act on the SPKI validation failure and takes appropriate action.
- o Cryptographic hash algorithm used to generate the SPKI pinsets
- o List of pre-shared key exchange modes
- o List of named groups (DHE or ECDHE) supported by the client
- o List signature algorithms the client can validate in X.509 server certificates

- o List 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.
- o List of supported application protocols (e.g., h3, h2, http/1.1 etc.)
- o List of certificate compression algorithms (defined in [I-D.ietf-tls-certificate-compression])
- o List of the distinguished names [X501] of acceptable certificate authorities, represented in DER-encoded format [X690] (defined in Section 4.2.4 of [RFC8446])
- o List of client key exchange algorithms and the client public key lengths in versions prior to (D)TLS 1.3

The (D)TLS profile parameters include the GREASE values for extension types, named groups, signature algorithms, (D)TLS versions, pre-shared key exchange modes and cipher suites, but normalized to the value 0x0a to preserve ordering information. Note that the GREASE values are random but their positions are deterministic (Section 5 of [RFC8701]) and peers will ignore these values and interoperate.

If the (D)TLS profile parameters are not observed in a (D)TLS session from the IoT device, the default behaviour is to block the (D)TLS session.

Note: The TLS and DTLS IANA registries are available from <https://www.iana.org/assignments/tls-parameters/tls-parameters.txt>.

5.1. Tree Structure

This document augments the "ietf-mud" MUD YANG module defined in [RFC8520] for signaling the IoT device (D)TLS profile. This document defines the YANG module "redy-opsawg-mud-tls-profile", which has the following tree structure:

```

module: reddy-opsawg-mud-tls-profile
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
  +--rw client-profile
    +--rw tls-profiles* [profile-name]
      +--rw profile-name          string
      +--rw protocol-version?     uint16
      +--rw supported_versions*   uint16
      +--rw grease_extension?     boolean
      +--rw encryption-algorithms* encryption-algorithm
      +--rw compression-methods*  compression-method
      +--rw extension-types*      extension-type
      +--rw acceptlist-ta-certs*   ct:trust-anchor-cert-cms
      +--rw SPKI-pin-sets*         SPKI-pin-set
      +--rw SPKI-hash-algorithm?   iha:hash-algorithm-type
      +--rw psk-key-exchange-modes* psk-key-exchange-mode
      +--rw supported-groups*      supported-group
      +--rw signature-algorithms-cert* signature-algorithm
      +--rw signature-algorithms*  signature-algorithm
      +--rw application-protocols* application-protocol
      +--rw cert-compression-algorithms* cert-compression-algorithm
      +--rw certificate_authorities* certificate_authorities
      +--rw client-public-keys
        +--rw key-exchange-algorithms* key-exchange-algorithm
        +--rw client-public-key-lengths* client-public-key-length

```

5.2. YANG Module

```

module reddy-opsawg-mud-tls-profile {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:reddy-opsawg-mud-tls-profile";
  prefix mud-tls-profile;

  import ietf-crypto-types {
    prefix ct;
    reference "draft-ietf-netconf-crypto-types-01:
              Common YANG Data Types for Cryptography";
  }

  import iana-hash-algs {
    prefix iha;
    reference
      "RFC XXXX: 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 Operations and Management Area Working Group Working Group";
  contact
    "Editor: Konda, Tirumaleswar Reddy
      <mailto:TirumaleswarReddy_Konda@McAfee.com>";

  description
    "This module contains YANG definition for the IoT device
      (D)TLS profile.

      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
      (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 2019-06-12 {
    description
      "Initial revision";
  }

  typedef compression-method {
    type uint8;
    description "Compression method";
  }

  typedef extension-type {
    type uint16;
    description "Extension type";
  }

  typedef encryption-algorithm {
    type uint16;
    description "Encryption algorithm";
  }

  typedef supported-group {
```

```
    type uint16;
    description "Named group (DHE or ECDHE)";
}

typedef SPKI-pin-set {
    type binary;
    description "Subject Public Key Info pin set";
}

typedef signature-algorithm {
    type uint16;
    description "Signature algorithm";
}

typedef key-exchange-algorithm {
    type uint8;
    description "key exchange algorithm";
}

typedef psk-key-exchange-mode {
    type uint8;
    description "pre-shared key exchange mode";
}

typedef client-public-key-length {
    type uint8;
    description "client public key length";
}

typedef application-protocol {
    type string;
    description "application protocol";
}

typedef cert-compression-algorithm {
    type uint8;
    description "certificate compression algorithm";
}

typedef certificate_authority {
    type binary;
    description "Distinguished Name of Certificate authority";
}

grouping client-profile {
    description
        "A grouping for (D)TLS profiles.";
    container client-profile {
```



```
list tls-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 protocol-version {
    type uint16;
    description "(D)TLS version in ClientHello.legacy_version";
  }
  leaf-list supported_versions {
    type uint16;
    description
      "TLS versions supported by the client indicated
      in the supported_versions extension in (D)TLS 1.3.";
  }
  leaf grease_extension {
    type boolean;
    description
      "If set to 'true', Grease extension values are offered by
      the client.";
  }
  leaf-list encryption-algorithms {
    type encryption-algorithm;
    description "Encryption algorithms";
  }
  leaf-list compression-methods {
    type compression-method;
    description "Compression methods";
  }
  leaf-list extension-types {
    type extension-type;
    description "Extension Types";
  }
  leaf-list acceptlist-ta-certs {
    type ct:trust-anchor-cert-cms;
    description
      "A list of trust anchor certificates used by the client.";
  }
  leaf-list SPKI-pin-sets {
    type 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 {
        type psk-key-exchange-mode;
        description
            "pre-shared key exchange modes";
    }
    leaf-list supported-groups {
        type supported-group;
        description
            "A list of named groups supported by the client.";
    }
    leaf-list signature-algorithms-cert {
        type signature-algorithm;
        description
            "A list signature algorithms the client can validate
            in X.509 certificates.";
    }
    leaf-list signature-algorithms {
        type signature-algorithm;
        description
            "A list signature algorithms the client can validate
            in the CertificateVerify message.";
    }
    leaf-list application-protocols {
        type application-protocol;
        description
            "A list application protocols supported by the client";
    }
    leaf-list cert-compression-algorithms {
        type cert-compression-algorithm;
        description
            "A list certificate compression algorithms
            supported by the client";
    }
    leaf-list certificate_authorities {
        type certificate_authority;
        description
            "A list of the distinguished names of certificate authorities
            acceptable to the client";
```

```

    }
    container client-public-keys {
      leaf-list key-exchange-algorithms {
        type key-exchange-algorithm;
        description
          "Key exchange algorithms supported by the client";
      }
      leaf-list client-public-key-lengths {
        type client-public-key-length;
        description
          "client public key lengths";
      }
    }
  }
}
}
}
augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
  description
    "MUD (D)TLS specific matches.";
  uses client-profile;
}
}

```

6. MUD File Example

This 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.

```

{
  "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,
    "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
                  }
                }
              },
              "redy-opsawg-mud-tls-profile:client-profile" : {
                "tls-profiles" : [
                  {
                    "protocol-version" : 771,
                    "supported_versions_ext" : "FALSE",
                    "encryption-algorithms" :
                      [31354, 4865, 4866, 4867],
                    "extension-types" : [10],
                    "supported-groups" : [29]
                  }
                ]
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      ]
    }
  ]
}

```

7. Security Considerations

Security considerations in [RFC8520] need to be taken into consideration. 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.

8. Privacy Considerations

The middle-box acting as a (D)TLS proxy must immediately delete the decrypted data upon completing any necessary inspection functions. TLS proxy potentially has access to a user's PII (Personally identifiable information) and PHI (Protected Health Information). The TLS proxy must not store, process or modify PII data. For example, IT administrator can configure firewall to bypass payload inspection for a connection destined to a specific service due to privacy compliance requirements.

9. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:redddy-opsawg-mud-tls-profile
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

10. Acknowledgments

Thanks to Flemming Andreassen, Shashank Jain, Michael Richardson, Piyush Joshi and Harsha Joshi for the discussion and comments.

11. References

11.1. Normative References

- [I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", draft-ietf-netconf-crypto-types-18 (work in progress), August 2020.
- [I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", draft-ietf-tls-certificate-compression-10 (work in progress), January 2020.
- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-38 (work in progress), May 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <<https://www.rfc-editor.org/info/rfc8519>>.

- [RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <<https://www.rfc-editor.org/info/rfc8701>>.
- [X690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2002, 2002.

11.2. Informative References

- [crypto-vulnerability] Perez, B., "Exploiting the Windows CryptoAPI Vulnerability", January 2020, <<https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF>>.
- [I-D.ietf-dnsop-svcb-https] Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", draft-ietf-dnsop-svcb-https-01 (work in progress), July 2020.
- [I-D.ietf-tls-sni-encryption] Huitema, C. and E. Rescorla, "Issues and Requirements for SNI Encryption in TLS", draft-ietf-tls-sni-encryption-09 (work in progress), October 2019.
- [I-D.reddy-add-enterprise] Reddy, K. T. and D. Wing, "DNS-over-HTTPS and DNS-over-TLS Server Deployment Considerations for Enterprise Networks", draft-reddy-add-enterprise-00 (work in progress), June 2020.
- [malware] Anderson, B., Paul, S., and D. McGrew, "Deciphering Malware's use of TLS (without Decryption)", July 2016, <<https://arxiv.org/abs/1607.01639>>.
- [malware-doh] Cimpanu, C., "First-ever malware strain spotted abusing new DoH (DNS over HTTPS) protocol", July 2019, <<https://www.zdnet.com/article/first-ever-malware-strain-spotted-abusing-new-doh-dns-over-https-protocol/>>.

[malware-tls]

Anderson, B. and D. McGrew, "TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior", October 2019, <<https://dl.acm.org/citation.cfm?id=3355601>>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<https://www.rfc-editor.org/info/rfc7469>>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.

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

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <<https://www.rfc-editor.org/info/rfc8576>>.

[X501] "Information Technology - Open Systems Interconnection - The Directory: Models", ITU-T X.501, 1993.

Authors' Addresses

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
USA

Email: danwing@gmail.com

Blake Anderson
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA 95134
USA

Email: blake.anderson@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 13, 2022

D. Shyti
L. Beylier
SFR
L. Iannone
Telecom ParisTech
September 9, 2021

A YANG Module for uCPE management.
draft-shyti-opsawg-vysm-10

Abstract

This document provides a YANG data model for uCPE management (VYSM) and definition of the uCPE equipment. The YANG Model serves as a base framework for managing an universal Customer-Premises Equipment (uCPE) subsystem. The model can be used by a Network Orchestrator.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 13, 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	2
2. Terminology	2
3. Universal CPE	3
3.1. uCPE purpose	4
3.2. uCPE VNF ecosystem example	4
3.3. Internal uCPE service example	5
4. YANG Model for uCPE management	6
5. Components for uCPE Management	7
6. Set of YANG Models	9
7. Diagram overview of YANG Data Model tree for uCPE management	12
8. Security Considerations	16
9. IANA Considerations	16
10. Acknowledgements	16
11. Normative References	17
Appendix A. Example of the uCPE resources management	18
Appendix B. Example of the uCPE resources management (deprecated)	21
Appendix C. Deprecated VNF YANG Model	22
Appendix D. XML example of deprecated YANG model	28
Authors' Addresses	30

1. Introduction

Network Function Virtualization is a technology that allows to virtualize the network services running on dedicated hardware. This technology became a base for universal Customer-Premises Equipment (uCPE). This document defines the uCPE as hardware with x86 capabilities that has a hypervisor. In other words, uCPE is a host that may run multiple Virtual Machines with guest OSs, where each Guest OS may represent a Physical Network Function. This document presents the YANG Model (VYSM) to manage from an Orchestrator the infrastructure inside the uCPE.

2. Terminology

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].

Link - is an entity that enables link layer communication of nodes.

Port - node connector to the link.

NE - Network Element.

NSYM - Network Yang Module.

VYSM - VNF YANG Model.

3. Universal CPE

Firstly, this document defines the platform that is controlled with VYSM - universal CPE (uCPE). The uCPE as hardware with x86 capabilities that is generally running Linux distribution with additional virtualization layer. Virtualization layer provides virtual compute, virtual storage and virtual network resources. Each VNF running in the uCPE requires the amount of virtual resources (for example: 4 vCPUs, 4GB RAM, 40GB storage, 4 vPorts). VNFs MAY be interconnected between each other and physical ports via Virtual Networks. Topology construction and VM life-cycle management is allowed via high level interface (Configuration can be done in the same transaction). The figure below presents the uCPE architecture.

VNF1	VNF2	VNF3	
Virtual Compute	Virtual Storage	Virtual Networks	uCPE software
PHY x86 processor	RAM+PHY storage	PHYsical ports	uCPE Hardware

The next elements can be managed in the uCPE:

o Virtual Network Functions:

- * Number of assigned vCPUs.
- * Size of allocated RAM.
- * VNF day0 config (bootstrap).
- * vLinks that are attached to the VNF.

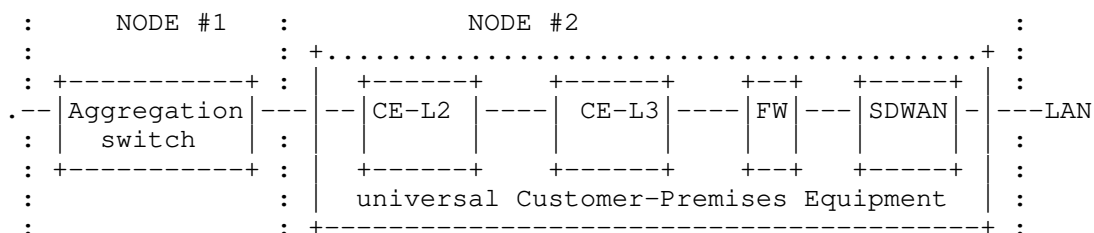
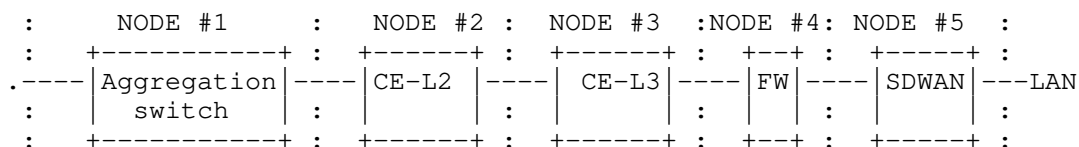
o Virtual Switches:

- * vLinks that are attached to the vSW.

- o Virtual Links(vLinks).
- o Physical Ports of the uCPE.

3.1. uCPE purpose

- o uCPE replaces multiple types of equipment (Node#1 - Node#5) with 1 unit by virtualizing them as Virtual Network Functions on the top of NFVIs:



- o uCPE facilitates the interconnection between the Network Functions (NF) as interconnection between NF is performed via virtual links(that is part of the uCPE management). That means that no need to hire technician to cable the equipment, it could be done via orchestrator.
- o uCPE facilitates the 0day configuration of the VNFs as its 0day configuration can be putted remotely.

3.2. uCPE VNF ecosystem example

uCPE supports a Virtual Network Functions of different type:

- o SD-WAN
- o vRouter
- o vFirewall

- o vLB(vLoad Balancer)
- o vCGNAT(vCarrier Grade NAT)
- o virtual WAN Optimistaion
- o vWireless LAN controller
- o Other...

3.3. Internal uCPE service example

The VNF in the uCPE could be a vRouter or vFirewall or an SD-WAN that is not a default part of virtual network resources of the uCPE. Multiple VNFs MAY be instantiated in the uCPE. With support of links and switches, VNFs MAY participate a service chains. Example of service chains (Note that virtual switch "vs(WAN)" connected to LAN ports and vSW(WAN) is connected to WAN ports):

- o vSW(WAN)-l1-vRouter-l2-vSW(LAN) .
- o vSW(WAN)-l1-vRouter-l2-vSW(Service)-l3-vFirewall-l4-vSW(LAN) .
- o vSW(WAN)-l1-vRouter-l2-vSW(Service1)-l3-vFirewall-l4-vSW(Service2)-l5-SD-WAN-l6-vSW(LAN) .
- o vSW(WAN)-l1-SDWAN-l2-vSW(Service)-l3-vFirewall-l4-vSW(LAN) .
- o

```

vSW(WAN1) --vRouter--+
                        +--vLoadBalance  vFirewall--vSW(LAN)
vSW(WAN2) --vRouter--+      |              |
                        +--vSW(Service1)+

```

o

```

vSW(WAN1) --vRouter(ISP1)--+
                        +--SD-WAN          vFirewall--vSW(LAN)
vSW(WAN2) --vRouter(ISP2)--+      |              |
                        +--vSW(Service1)+

```

4. YANG Model for uCPE management

Secondly, this document defines and classifies the YANG Model for uCPE Management. This Module is modeled representation of the specific network requirements. It provides abstraction of network configuration and operations. The YANG Model for uCPE Management does not describe all configuration to be performed on the devices, but provides the configuration that is required for the "Network to Network Element(s)" decomposition process RFC 8199 [RFC8199]. Example of the decomposition is presented in the figure below.

The Network YANG module exposes the configuration commands via the Northbound interfaces of the orchestrator. Therefore the set of the commands modeled in the VYSM can be inputted via Northbound interfaces (for example CLI). In the example the command "vm VNF1" is passed via Northbound interface to the orchestrator. It defines the virtual machine name. Further the same configuration MAY be transformed to the one or multiple Network Element payloads (for example xml for NETCONF) that carry an equivalent of commands such as "nf nf-name VNF1"

```

      +---+---+---+---+---+---+
      |               |
      |   config t    |
      |   vm VNF1     |
      |               |
      +---+---+---+---+---+---+
          #
          #

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:
: +---+---+---+---+---+---+---+---+---+
: |   Network YANG Module   | <= scope of this document
: +---+---+---+---+---+---+---+---+---+
:
:               #
: #####
: #               #               #
: '-----' / '-----' / '-----'
: 'Module1' / 'Module 2' / 'Module3'
: '-----' / '-----' / '-----'
:
: #               #               #
: #               #               #####
: #####          #####          #
: #               #               #
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      #               #               #
      Network # element 1   Network # element 2   Network # element3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| domains domain VNF1 | |tenants tenant name VNF1| |nf nf-name VNF1|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

5. Components for uCPE Management

This section provides a components overview to manage the uCPE.

There are multiple RFCs and drafts produced by the IETF community, that are referenced in the YANG tree to manage the uCPE. Each document produced by the IETF covers a part of uCPE Management. The list of the documents is provided below:

- o [RFC8530] - logical network elements (VNFs) properties.
- o [RFC8345] - definition of networks, nodes, node-termination-points: network includes the uCPE with uCPE's physical termination points.
- o [I-D.ietf-teas-sf-aware-topo-model] physical ports and service functions (VNFs) interconnection matrices (PhyPort-VNF, VNF-VNF).

- o This document itself provides yang modules that completes the existing documents produced by IETF.

This document introduces yang modules for 'logical network elements properties(VNFs)' part:

- o day0-info: mapping between variables inside of the bootstrap config and required values in the list "day0-info". In the bootstrap config the variable could be putted instead value. The value could be set in the day0-info part (check the YANG model) and after the value in the list will be mapped to the variable in the bootstrap config.
- o vCPU/vRAM/vDisk/VNF-ports leafs and lists.

The minimal list of yang models required for compilation of the YANG tree to manage the uCPE is presented below:

- o ieee-dot1Q-types
- o ietf-interfaces
- o ietf-ip
- o ietf-logical-network-element
- o ietf-network
- o ietf-network-instance
- o ietf-ietf-network-topology
- o ietf-routing-types
- o ietf-te-topology
- o ietf-te-topology-sf
- o ietf-te-types
- o ietf-yang-schema-mount
- o etsi-sol-006-deviation
- o The YANG modules introduced in this document:
- o

- * ietf-ucpe-lne-properties
- * ietf-ucpe-lt-virtual-link-id
- * ietf-ucpe-ni-properties
- * ietf-ucpe-node-type

6. Set of YANG Models

This section provides a YANG models that address uCPE network service resources management oranized according to the ID [I-D.ietf-netmod-yang-packages]

```
<CODE BEGINS> file "ietf-ucpe-network-service-pkg.json"
===== NOTE: '\ ' line wrapping per BCP XX (RFC XXXX) =====

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "ietf-ucpe-network-service-pkg",
    "pkg-schema": {
      package: "ietf-yang-package-defn-pkg@0.1.0.json"
    },
    "description": "YANG package for universal CPE network service",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "ietf-ucpe-network-service-pkg",
        "version": "0.0.1",
        "timestamp": "2021-09-09T17:00:00Z",
        "organization": "IETF OPSAWG Working Group",
        "contact" : "WG Web:  <http://tools.ietf.org/wg/opsawg/>, \
                      WG List: <mailto:opsawg@ietf.org>, \
                      Author:  <mailto:ietf.dmytro@shytyi.net>",
        "description": "IETF uCPE network service YANG package.\
\
This package defines a small sample set of \
YANG modules that could represent the basic set of \
modules that a standard universal CPE device might be \
expected \
to support.",

        "reference": "XXX, draft-shytyi-opsawg-vysm-10.xml",
        "location": [ "https://github.com/dmytroshytyi/ucpe-ietf/\
                      ietf-ucpe-service@v0.0.1.json" ],
        "module": [
          {
            "name": "ieee-dot1Q-types,
```

```
    "revision": "2015-08-18",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/ieee-dot1Q-types.yang" ],
  },
  {
    "name": "ietf-interfaces",
    "revision": "2018-02-20",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-interfaces%402018-02-20.yang" ],
  },
  {
    "name": "ietf-ip",
    "revision": "2018-02-22",
    "location": [ "https://github.com/dmytroshytyi/ucpe-ietf\
/blob/master/ietf-ip%402018-02-22.yang" ],
  },
  {
    "name": "ietf-logical-network-element",
    "revision": "2019-01-25",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-logical-network-element%402019-01-25.yang" ],
  },
  {
    "name": "ietf-network",
    "revision": "2018-02-26",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-network%402018-02-26.yang" ],
  },
  {
    "name": "ietf-network-instance",
    "revision": "2019-01-21",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-network-instance%402019-01-21.yang" ],
  },
  {
    "name": "ietf-network-topology",
    "revision": "2018-02-26",
    "location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-network-topology%402018-02-26.yang" ],
  },
  {
    "name": "ietf-routing-types",
    "revision": "2017-12-04",
```

```
    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-routing-types%402017-12-04.yang" ],\n  },\n  {\n    "name": "ietf-te-topology",\n    "revision": "2019-02-07",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-te-topology%402019-02-07.yang" ],\n  },\n  {\n    "name": "ietf-te-topology-sf",\n    "revision": "2019-11-03",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-te-topology-sf%402019-11-03.yang" ],\n  },\n  {\n    "name": "ietf-te-types",\n    "revision": "2019-11-18",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-te-types%402019-11-18.yang" ],\n  },\n  {\n    "name": "ietf-ucpe-lne-properties",\n    "revision": "2019-11-21",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-ucpe-lne-properties%402019-11-21.yang" ],\n  },\n  {\n    "name": "ietf-ucpe-lt-virtual-link-id",\n    "revision": "2020-02-14",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-ucpe-lt-virtual-link-id%402020-02-14.yang" ],\n  },\n  {\n    "name": "ietf-ucpe-ni-properties",\n    "revision": "2019-11-27",\n    "location": [ "https://github.com/dmytroshytyi/\nucpe-ietf/blob/master/\nietf-ucpe-ni-properties%402019-11-27.yang" ],\n  },\n  {\n    "name": "ietf-ucpe-node-type",
```

```
"revision": "2020-02-14",
"location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-ucpe-node-type%402020-02-14.yang" ],
},
{
"name": "ietf-ucpe-ni-properties",
"revision": "2019-11-27",
"location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-ucpe-ni-properties%402019-11-27.yang" ],
},
{
"name": "ietf-yang-schema-mount",
"revision": "2019-01-14",
"location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
ietf-yang-schema-mount%402019-01-14.yang" ],
},
{
"name": "etsi-nfv-common-deviation",
"revision": "2020-06-10",
"location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
submodules/etsi-nfv-common-deviation.yang" ],
},
{
"name": "etsi-nfv-vnf-deviation",
"revision": "2020-06-10",
"location": [ "https://github.com/dmytroshytyi/\
ucpe-ietf/blob/master/\
submodules/etsi-nfv-vnf-deviation.yang" ],
}
]
}
}
}
}
<CODE ENDS>
```

This section provides an overview of the Data YANG Model that MAY be made with "pyang" utility. The figure below presents the tree diagram.

```

module: ietf-logical-network-element
+--rw logical-network-elements
  +--rw logical-network-element* [name]
    +--rw name string
    +--rw managed? boolean
    +--rw description? string
    +--rw root
    +--rw ietf-ucpe:logical-network-element-properties
      +--rw ietf-ucpe:etsi
        +--rw ietf-ucpe:vnfd? -> /nfv/vnfd/id
        +--rw ietf-ucpe:vdu? -> /nfv/vnfd[id=current()]\
          /../vnfd]/vdu/id
      +--rw ietf-ucpe:supporting-node? -> /nw:networks\
        /network/node/node-id
      +--rw ietf-ucpe:uuid? enumeration
      +--rw ietf-ucpe:uuid-custom-value? string
      +--rw ietf-ucpe:persistence-id? string
      +--rw ietf-ucpe:pci-passthrough
        +--rw ietf-ucpe:device* [device-name]
          +--rw ietf-ucpe:device-name string
          +--rw ietf-ucpe:vendor-id? string
          +--rw ietf-ucpe:device-id? string
          +--rw ietf-ucpe:device-index? int64
      +--rw ietf-ucpe:sf-cp-params* [sf-connection-point-id]
        +--rw ietf-ucpe:sf-connection-point-id string
        +--rw ietf-ucpe:io-acceleration
          +--rw ietf-ucpe:interface-type? enumeration
          +--rw ietf-ucpe:interface-model? enumeration
          +--rw ietf-ucpe:number-of-queues? uint64
        +--rw ietf-ucpe:mac-params
          +--rw ietf-ucpe:mac-type? enumeration
          +--rw ietf-ucpe:custom-mac-address? string
      +--rw ietf-ucpe:simplified-lne-props
        +--rw ietf-ucpe:sf-connection-points* \
          [sf-connection-point-id]
          +--rw ietf-ucpe:sf-connection-point-id string
        +--rw ietf-ucpe:ram? uint64
        +--rw ietf-ucpe:cpu? uint64
        +--rw ietf-ucpe:storages* [id]
          +--rw ietf-ucpe:id string
          +--rw ietf-ucpe:location? string
      +--rw ietf-ucpe:day0-config
        +--rw ietf-ucpe:location? string
        +--rw ietf-ucpe:day0-var-path? string
        +--rw ietf-ucpe:variable* [name]
          +--rw ietf-ucpe:name string
          +--rw ietf-ucpe:value? string
module: ietf-network

```

```

+--rw networks
  +--rw network* [network-id]
    |   +--rw network-id                network-id
    |   +--rw network-types
    |     |   +--rw tet:te-topology!
    |     |   +--rw tet-sf:sf!
    |   +--rw supporting-network* [network-ref]
    |     |   +--rw network-ref    -> /networks/network\
    |     |                       /network-id
    |   +--rw node* [node-id]
    |     |   +--rw node-id                node-id
    |     |   +--rw supporting-node* [network-ref node-ref]
    |     |     |   +--rw network-ref    -> ../../../../supporting\
    |     |     |   -network/network-ref
    |     |     |   +--rw node-ref      -> /networks/network\
    |     |     |   /node/node-id
    |     |   +--rw nt:termination-point* [tp-id]
    |     |     |   +--rw nt:tp-id                tp-id
    |     |     |   +--rw nt:supporting-termination-point* \
    |     |     |     [network-ref node-ref tp-ref]
    |     |     |   +--rw nt:network-ref    -> ../../../../\
    |     |     |     nw:supporting-node/network-ref
    |     |     |   +--rw nt:node-ref      -> ../../../../\
    |     |     |     nw:supporting-node/node-ref
    |     |     |   |   |   +--rw nt:tp-ref    -> \
    |     |     |     |   /nw:networks/network[nw:network-id=current()/ \
    |     |     |     |   ../../network-ref]/node[nw:node-id=current()/ \
    |     |     |     |   ../../node-ref]/termination-point/tp-id
    |     |   +--rw tet:te-node-id?                te-types:te-node-id
    |     +--rw tet:te!
    |       |   +--rw tet:te-node-template*          -> ../../../../\
    |       |   /te/templates/node-template\
    |       |   /name {template}?
    |       +--rw tet:te-node-attributes
    |         +--rw tet-sf:service-function
    |           +--rw tet-sf:connectivity-matrices
    |             +--rw tet-sf:connectivity-matrix* [id]
    |               +--rw tet-sf:id                uint32
    |               +--rw tet-sf:from
    |                 |   +--rw tet-sf:service-function-id?    string
    |                 |   +--rw tet-sf:sf-connection-point-id? string
    |               +--rw tet-sf:to
    |                 |   +--rw tet-sf:service-function-id?    string
    |                 |   +--rw tet-sf:sf-connection-point-id? string
    |               +--rw tet-sf:enabled?            boolean
    |               +--rw tet-sf:direction? \
    |               connectivity-direction
    |             +--rw tet-sf:virtual-link-id?    string

```

```

| | | | +--rw tet-sf:link-terminations
| | | |   +--rw tet-sf:link-termination* [id]
| | | |     +--rw tet-sf:id                               uint32
| | | |     +--rw tet-sf:from
| | | |       | +--rw tet-sf:tp-ref?  -> ../../../../\
| | | |       | ../../nt:termination-point/tp-id
| | | |     +--rw tet-sf:to
| | | |       | +--rw tet-sf:service-function-id?  \
| | | |       | string
| | | |       | +--rw tet-sf:sf-connection-point-id? \
| | | |       | string
| | | |     +--rw tet-sf:enabled?                      boolean
| | | |     +--rw tet-sf:direction?                    \
| | | |     connectivity-direction
| | | |     +--rw lt-vlink-id:virtual-link-id?        string

```

```

module: ietf-network-instance

```

```

+--rw network-instances
  +--rw network-instance* [name]
    +--rw name                                     string
    +--rw enabled?                               boolean
    +--rw description?                           string
    +--rw (ni-type)?
    +--rw (root-type)
      +--:(vrf-root)
        | +--rw vrf-root
      +--:(vsi-root)
        | +--rw vsi-root
        | +--rw ietf-ucpe-ni:network-instance-properties
        |   +--rw ietf-ucpe-ni:sf-connection-points* \
        |   [sf-connection-point-id]
        |     | +--rw ietf-ucpe-ni:sf-connection-point-id  string
        |     | +--rw ietf-ucpe-ni:stacked-vlans
        |     |   +--rw ietf-ucpe-ni:outer-VLAN-0x8100?  \
        |     |   dlq:vid-range
        |     |   | +--rw ietf-ucpe-ni:inner-VLANs-0x8100*  uint16
        |     | +--rw ietf-ucpe-ni:QinQ
        |     |   +--rw ietf-ucpe-ni:svlan-0x88a8?      dlq:vid-range
        |     |   +--rw ietf-ucpe-ni:cvlans-0x8100*    uint16
        |     | +--rw ietf-ucpe-ni:dot1q-vlan
        |     | +--rw ietf-ucpe-ni:access-tag?          \
        |   dlq:vid-range
        |   +--rw ietf-ucpe-ni:trunk-allowed-vlans*    \
        |   uint16
        |   +--rw ietf-ucpe-ni:port-mode?              \
        |   enumeration
        +--rw ietf-ucpe-ni:io-acceleration

```



```

|         |         | +--rw ietf-ucpe-ni:dppk
|         |         |     +--rw ietf-ucpe-ni:rx-queues?    uint16
|         |         |     +--rw ietf-ucpe-ni:tx-queues?    uint16
|         |         |     +--rw ietf-ucpe-ni:cpu-mask?      uint16
|         |         |   +--rw ietf-ucpe-ni:ebpf-xdp
|         |       +--rw ietf-ucpe-ni:ni-area?                identityref
|         |       +--rw ietf-ucpe-ni:supporting-node?        -> \
|         |           /nw:networks\
|         |           /network/node/node-id
+--:(vv-root)
  +--rw vv-root

module: ietf-ucpe-lne-properties
  +--rw nfvd
    +--rw vnfd* [id]
      +--rw id                        string
      +--rw provider                  string
      +--rw product-name              string
      +--rw software-version          string
      +--rw version                   string
      +--rw product-info-name?        string
      +--rw product-info-description? string
      +--rw vnf-info*                 string
      +--rw localization-language?    string
      +--rw default-localization-language? string
      +--rw vdu* [id]
        +--rw id                      string
        +--rw name                     string
        +--rw description?             string
        +--rw int-cpd* [id]
etc... Following ETSI SOL006
```

At this time, no security considerations are addressed by this memo.

No request to IANA at this time.

the authors would like to thank:

- o Mahesh Jethanandani.

- o Robert Varga.
- o Bill Wu.
- o Joe Clarke.
- o Tom Petch.
- o Martin Bjorklund.
- o Schonwalder Jurgen.
- o Dean Bogdanovic.
- o Bo Wu.

for their valuable comments.

11. Normative References

[I-D.ietf-netmod-yang-packages]

Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", draft-ietf-netmod-yang-packages-01 (work in progress), November 2020.

[I-D.ietf-teas-sf-aware-topo-model]

Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L., Ceccarelli, D., and J. Tantsura, "SF Aware TE Topology YANG Model", draft-ietf-teas-sf-aware-topo-model-03 (work in progress), March 2019.

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

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.

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

[RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <<https://www.rfc-editor.org/info/rfc8530>>.

Appendix A. Example of the uCPE resources management

This section provides an overview of the YIN format.

```
<networks xmlns="urn:ietf:params:xml:ns:yang:ietf-network">
  <network>
    <network-id>network-1</network-id>
    <network-types>
      <te-topology xmlns="urn:ietf:params:xml:ns:yang:ietf-te-topology">
        <sf xmlns="urn:ietf:params:xml:ns:yang:ietf-te-topology-sf"/>
      </te-topology>
    </network-types>
    <node>
      <node-id>ucpe1</node-id>
      <te-node-id xmlns="urn:ietf:params:xml:ns:yang:ietf-te-topology" \
        >0.0.0.0</te-node-id>
      <te xmlns="urn:ietf:params:xml:ns:yang:ietf-te-topology">
        <te-node-attributes>
          <service-function
            xmlns="urn:ietf:params:xml:ns:yang:ietf-te-topology-sf">
            <connectivity-matrices>
              <connectivity-matrix>
                <id>1</id>
                <from>
                  <service-function-id>VMone</service-function-id>
                  <sf-connection-point-id>1</sf-connection-point-id>
                </from>
                <to>
                  <service-function-id>SwitchOne</service-function-id>
                  <sf-connection-point-id>11</sf-connection-point-id>
                </to>
                <virtual-link-id>l11</virtual-link-id>
              </connectivity-matrix>
              <connectivity-matrix>
                <id>2</id>
                <from>
                  <service-function-id>VMtwo</service-function-id>
                  <sf-connection-point-id>1</sf-connection-point-id>
                </from>
                <to>
                  <service-function-id>SwitchOne</service-function-id>
                  <sf-connection-point-id>12</sf-connection-point-id>
```

```

        </to>
        <virtual-link-id>l12</virtual-link-id>
    </connectivity-matrix>
    <connectivity-matrix>
        <id>3</id>
        <from>
            <service-function-id>VMthree</service-function-id>
            <sf-connection-point-id>1</sf-connection-point-id>
        </from>
        <to>
            <service-function-id>SwitchOne</service-function-id>
            <sf-connection-point-id>13</sf-connection-point-id>
        </to>
        <virtual-link-id>l13</virtual-link-id>
    </connectivity-matrix>
    <connectivity-matrix>
        <id>4</id>
        <from>
            <service-function-id>VMfour</service-function-id>
            <sf-connection-point-id>1</sf-connection-point-id>
        </from>
        <to>
            <service-function-id>SwitchOne</service-function-id>
            <sf-connection-point-id>14</sf-connection-point-id>
        </to>
        <virtual-link-id>l14</virtual-link-id>
    </connectivity-matrix>
</connectivity-matrices>
</service-function>
</te-node-attributes>
</te>
</node>
</network>
</networks>

<logical-network-elements \
  xmlns="urn:ietf:params:xml:ns:yang:ietf-logical-network-element">
  <logical-network-element>
    <name>VMfour</name>
    <logical-network-element-properties \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe-lne-properties">
      <sf-connection-points>
        <sf-connection-point-id>1</sf-connection-point-id>
      </sf-connection-points>
      <supporting-node>ucpe1</supporting-node>
      <ram>1024</ram>
      <cpu>4</cpu>
      <storages>

```

```
        <id>1</id>
        <location>vm4.qcow2</location>
    </storages>
</logical-network-element-properties>
</logical-network-element>
<logical-network-element>
    <name>VMone</name>
    <logical-network-element-properties \
        xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe-lne-properties">
        <sf-connection-points>
            <sf-connection-point-id>1</sf-connection-point-id>
        </sf-connection-points>
        <supporting-node>ucpe1</supporting-node>
        <ram>1024</ram>
        <cpu>4</cpu>
        <storages>
            <id>1</id>
            <location>vm1.qcow2</location>
        </storages>
    </logical-network-element-properties>
</logical-network-element>
<logical-network-element>
    <name>VMthree</name>
    <logical-network-element-properties \
        xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe-lne-properties">
        <sf-connection-points>
            <sf-connection-point-id>1</sf-connection-point-id>
        </sf-connection-points>
        <supporting-node>ucpe</supporting-node>
        <ram>1024</ram>
        <cpu>4</cpu>
        <storages>
            <id>1</id>
            <location>vm3qcow2</location>
        </storages>
    </logical-network-element-properties>
</logical-network-element>
<logical-network-element>
    <name>VMtwo</name>
    <logical-network-element-properties \
        xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe-lne-properties">
        <sf-connection-points>
            <sf-connection-point-id>1</sf-connection-point-id>
        </sf-connection-points>
        <supporting-node>ucpe1</supporting-node>
        <ram>1024</ram>
        <cpu>4</cpu>
        <storages>
```

```
        <id>1</id>
        <location>vm4.iso</location>
      </storages>
    </logical-network-element-properties>
  </logical-network-element>
</logical-network-elements>
<network-instances \
  xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
  <network-instance>
    <name>SwitchOne</name>
    <network-instance-properties \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe-ni-properties">
      <sf-connection-points>
        <sf-connection-point-id>10</sf-connection-point-id>
        <dot1q-vlan>
          <trunk-allowed-vlans>112</trunk-allowed-vlans>
          <trunk-allowed-vlans>113</trunk-allowed-vlans>
          <trunk-allowed-vlans>114</trunk-allowed-vlans>
          <port-mode>trunk</port-mode>
        </dot1q-vlan>
      </sf-connection-points>
      <sf-connection-points>
        <sf-connection-point-id>11</sf-connection-point-id>
      </sf-connection-points>
      <dot1q-vlan>
        <access-tag>111</access-tag>
      </dot1q-vlan>
      <sf-connection-points>
        <sf-connection-point-id>12</sf-connection-point-id>
      </sf-connection-points>
      <sf-connection-points>
        <sf-connection-point-id>13</sf-connection-point-id>
      </sf-connection-points>
      <sf-connection-points>
        <sf-connection-point-id>14</sf-connection-point-id>
      </sf-connection-points>
      <supporting-node>ucpe1</supporting-node>
    </network-instance-properties>
  </network-instance>
</network-instances>
```

Appendix B. Example of the uCPE resources management (deprecated)

This section provides an overview of the deprecated YANG Model that MAY give an alternative view on the uCPE management.

```

module: ietf-example-ucpe
  +--rw ucpe* [name]
    +--rw name          string
    +--rw links* [link]
      | +--rw link      string
    +--rw phyInterfaces* [interface]
      | +--rw interface  string
      | +--rw ports* [port]
      |   +--rw port      string
      |   +--rw link?    -> ../../../../links/link
    +--rw switches* [switch]
      | +--rw switch      string
      | +--rw ports* [port]
      |   +--rw port      string
      |   +--rw name?     string
      |   +--rw link?    -> ../../../../links/link
    +--rw vms* [vm]
      +--rw vm            string
      +--rw ports* [port]
        | +--rw port      string
        | +--rw name?     string
        | +--rw link?    -> ../../../../links/link
      +--rw ram?          uint64
      +--rw cpu?          uint64
      +--rw storages* [id]
        | +--rw id        string
        | +--rw location? string
      +--rw day0-config
        +--rw location?   string
        +--rw day0-var-path? string
        +--rw variable* [name]
          +--rw name      string
          +--rw value?    string

```

Appendix C. Deprecated VNF YANG Model

This section provides a deprecated yang model that addresses the configuration of the uCPE resources presented above.

```

<CODE BEGINS> file "ietf-example-ucpe@2019-10-28.yang"
module ietf-example-ucpe {
  namespace "urn:ietf:params:xml:ns:yang:ietf-example-ucpe";
  prefix ietf-example-ucpe;

  organization
    "SFR";

```

contact

"Dmytro Shytyi
EMail:ietf.dmytro@shytyi.net";

description

"This is a Network Function Virtualization (NFV) YANG
service model.

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 2019-10-28 {

description

"Yang model with vPorts assigned to the interfaces";

reference

"draft-shytyi-opsawg-vysm-05";

}

revision 2019-10-19 {

description

"Yang model was cleaned. Interfaces added";

reference

"draft-shytyi-opsawg-vysm-04";

}

revision 2019-09-16 {

description

"Added 0day config for VNFs.
Yang model modified according
to the received comments.";

reference

"draft-shytyi-opsawg-vysm-00";

}

revision 2018-01-07 {

description

"Initial revision.";

reference

"draft-shytyi-netmod-vysm-01";

}


```
list ucpe {
  key "name";
  leaf name {
    type string;
    description
      "ID of uCPE where
       a service is instantiated";
  }
  list links {
    key "link";
    leaf link {
      type string;
      description
        "Name of the virtual link from the pool
         of the links";
    }
    description
      "Pool of the virtual links that connect VMs and
       Interfaces";
  }
  list phyInterfaces {
    key "interface";
    leaf interface {
      type string;
      description
        "Name of physical interface";
    }
  }
  list ports {
    key "port";
    leaf port {
      type string;
      description
        "Name of the connector";
    }
    leaf link {
      type leafref {
        path "../.../links/link";
      }
      description
        "Link that is connected to
         the port via connector";
    }
    description
      "Set of the connectors the
       physical interface has";
  }
  description
    "Set of physical interfaces";
}
```

```
}
list switches {
  key "switch";
  leaf switch {
    type string;
    description
      "Name of the forwarding domain";
  }
  list ports {
    key "port";
    leaf port {
      type string;
      description
        "Name of the connector";
    }
    leaf name {
      type string;
      description
        "Name of the
        subconnector";
    }
    leaf link {
      type leafref {
        path "../.../links/link";
      }
      description
        "Link that is connected to the
        switch via port";
    }
    description
      "Set of the connectors the
      forwarding domain has";
  }
  description
    "Set of the forwarding domains";
}
list vms {
  key "vm";
  leaf vm {
    type string;
    description
      "ID of the Virtual Machine";
  }
  list ports {
    key "port";
    leaf port {
      type string;
      description
```

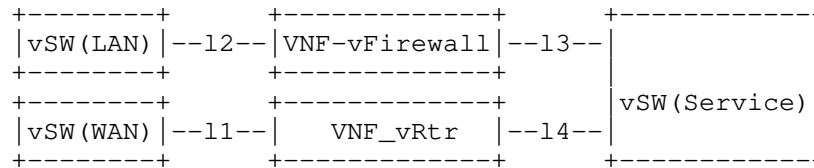
```
        "Name of the connector";
    }
    leaf name {
        type string;
        description
            "Name of
             the subconnector";
    }
    leaf link {
        type leafref {
            path "../../links/link";
        }
        description
            "Link that connects the
             VM with a switch or Interface
             via connector";
    }
    description
        "Set of Virtual Machine connectors";
}
leaf ram {
    type uint64;
    description
        "Size of RAM to allocate for
         the Guest OS";
}
leaf cpu {
    type uint64;
    description
        "Number of vCPUs to
         allocate for the Guest OS";
}
list storages {
    key "id";
    leaf id {
        type string;
        description
            "Number of
             vDisk attached to the VM";
    }
    leaf location {
        type string;
        description
            "External location where
             the image (ex.qcow2) is saved.";
    }
    description
        "Virtual storage/vDisk
```

```
        attached to the Virtual Machine";
    }
    container day0-config {
        leaf location {
            type string;
            description
                "0day configuration location";
        }
        leaf day0-var-path {
            type string;
            description
                "path of the file
                that contains the 0day variables";
        }
        list variable {
            key "name";
            leaf name {
                type string;
                description
                    "variable name";
            }
            leaf value {
                type string;
                description
                    "variable value";
            }
            description
                "list of variables";
        }
        description
            "0day configuration:init config";
    }
    description
        "Set of the Virtual Machines configured
        on the universal Customer-Premises Equipment";
}
description
    "This is an uCPE management service";
}
}

<CODE ENDS>
```

Appendix D. XML example of deprecated YANG model

The XML example below presents the configuration of the next service in the uCPE, where: vSW(LAN), vSW(WAN), vSW(Service) - virtual switches; l1,l2,l3,l4 - virtual links; VMs represent PNFs (Physical Network Functions) that could be bootstrapped with 0day config/license.



```

<ucpe xmlns="urn:ietf:params:xml:ns:yang:ietf-ucpe">
  <name>ucpe1</name>
  <links>
    <link>l1</link>
  </links>
  <links>
    <link>l2</link>
  </links>
  <links>
    <link>l3</link>
  </links>
  <links>
    <link>l4</link>
  </links>
  <switches>
    <switch>lan</switch>
    <ports>
      <port>10</port>
      <name>l2p10</name>
      <link>l2</link>
    </ports>
  </switches>
  <switches>
    <switch>service</switch>
    <ports>
      <port>10</port>
      <name>l3p10</name>
      <link>l3</link>
    </ports>
  </switches>
</ucpe>

```

```
        <port>11</port>
        <name>l4p10</name>
        <link>l4</link>
    </ports>
</switches>
<switches>
    <switch>wan</switch>
    <ports>
        <port>10</port>
        <link>l1</link>
    </ports>
</switches>
<vms>
    <vm>VNF-vRtr</vm>
    <ports>
        <port>1</port>
        <name>l1p1</name>
        <link>l1</link>
    </ports>
    <ports>
        <port>2</port>
        <name>l4p2</name>
        <link>l4</link>
    </ports>
    <ram>2048</ram>
    <cpu>2</cpu>
    <storages>
        <id>1</id>
        <location>http://192.168.2.1/vRtr-x86.qcow2</location>
    </storages>
    <day0-config>
        <location>https://192.168.2.1/vRtr-day0.iso</location>
        <day0-var-path>/config.rom</day0-var-path>
        <variable>
            <name>hostname</name>
            <value>IETF-vRtr</value>
        </variable>
        <variable>
            <name>ipaddress</name>
            <value>192.168.1.2 255.255.255.0</value>
        </variable>
    </day0-config>
</vms>
<vms>
    <vm>VNF-vFirewall</vm>
    <ports>
        <port>1</port>
        <name>l3p1</name>
```

```
        <link>13</link>
    </ports>
    <ports>
        <port>2</port>
        <name>l2p2</name>
        <link>l2</link>
    </ports>
    <ram>2048</ram>
    <cpu>2</cpu>
    <storages>
        <id>1</id>
        <location>http://192.168.2.1/vFirewall-x86.qcow2</location>
    </storages>
    <day0-config>
        <location>https://192.168.2.1/vFirewall-day0.iso</location>
        <day0-var-path>/config.rom</day0-var-path>
        <variable>
            <name>hostname</name>
            <value>vFirewall</value>
        </variable>
        <variable>
            <name>ipaddress</name>
            <value>192.168.1.3 255.255.255.0</value>
        </variable>
    </day0-config>
</vms>
</ucpe>
```

Authors' Addresses

Dmytro Shytyi
SFR
Paris , Ile-de-France
France

Email: ietf.dmytro@shytyi.net
URI: <https://dmytro.shytyi.net>

Laurent Beylier
SFR
Paris , Ile-de-France
France

Email: laurent.beylier@sfr.com

Luigi Iannone
Telecom ParisTech
Paris , Ile-de-France
France

Email: luigi.iannone@telecom-paristech.fr

OPSAWG
Internet-Draft
Intended status: Informational
Expires: 26 August 2022

H. Song
Futurewei
F. Qin
China Mobile
H. Chen
China Telecom
J. Jin
LG U+
J. Shin
SK Telecom
22 February 2022

A Framework for In-situ Flow Information Telemetry
draft-song-opsawg-ifit-framework-17

Abstract

As network scale increases and network operation becomes more sophisticated, existing Operation, Administration, and Maintenance (OAM) methods are no longer sufficient to meet the monitoring and measurement requirements. Emerging data-plane on-path telemetry techniques which provide high-precision flow insight and which issue notifications in real time can supplement existing proactive and reactive methods that run in active and passive modes. These new approaches are collectively known as in-situ flow information telemetry (IFIT). They enable quality of experience for users and applications, and identification of network faults and deficiencies.

This document outlines a high-level framework for IFIT to collect and correlate performance measurement information from the network. It identifies the components that coordinate existing protocol tools and telemetry mechanisms, and addresses deployment challenges for flow-oriented on-path telemetry techniques, especially in carrier networks.

The document is a guide for system designers applying the referenced techniques. It is also intended to motivate further work to enhance the OAM ecosystem.

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 26 August 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
1.1. Classification and Modes of On-path Telemetry	4
1.2. Requirements and Challenges	6
1.3. Scope	8
1.4. Relationship with Network Telemetry Framework (NTF)	8
1.5. Glossary	9
2. Architectural Concepts and Key Components	9
2.1. Reference Deployment	9
2.2. Key Components	11
2.2.1. Flexible Flow, Packet, and Data Selection	11
2.2.2. Flexible Data Export	13
2.2.3. Dynamic Network Probe	15
2.2.4. On-demand Technique Selection and Integration	17
2.3. IFIT for Reflective Telemetry	18
2.3.1. Intelligent Multipoint Performance Monitoring	19
2.3.2. Intent-based Network Monitoring	19
3. Guidance for Solution Developers	20
3.1. Encapsulation in Transport Protocols	20
3.2. Tunneling Support	21
3.3. Deployment Automation	21

4. Security Considerations	22
5. IANA Considerations	22
6. Contributors	22
7. Acknowledgments	22
8. References	22
8.1. Normative References	22
8.2. Informative References	23
Authors' Addresses	27

1. Introduction

Efficient network operation increasingly relies on high-quality data-plane telemetry to provide the necessary visibility into the behavior of traffic flows and network resources. Existing Operation, Administration, and Maintenance (OAM) methods, which include proactive and reactive techniques, running both active and passive modes, are no longer sufficient to meet the monitoring and measurement requirements when networks becomes more autonomous [RFC8993] and application-aware [I-D.li-apn-framework]. The complexity of today's networks and service quality requirements demand new high-precision and real-time OAM techniques.

The ability to expedite network failure detection, fault localization, and recovery mechanisms, particularly in the case of soft failures or path degradation is expected, and it must not cause service disruption. Emerging on-path telemetry techniques can provide high-precision flow insight and real-time network issue notification (e.g., jitter, latency, packet loss, significant bit error variations, and unequal load-balancing). On-Path Telemetry (OPT) refers to data-plane telemetry techniques that directly tap and measure network traffic by embedding instructions or metadata into user packets. The data provided by on-path telemetry are especially useful for verifying Service Level Agreement (SLA) compliance, user experience enhancement, service path enforcement, fault diagnosis, and network resource optimization. It is essential to recognize that existing work on this topic includes a variety of on-path telemetry techniques, including In-situ OAM (IOAM) [I-D.ietf-ippm-ioam-data], IOAM Direct Export (DEX) [I-D.ietf-ippm-ioam-direct-export], Marking-based Postcard-based Telemetry (PBT-M) [I-D.song-ippm-postcard-based-telemetry], Enhanced Alternate Marking (EAM) [I-D.zhou-ippm-enhanced-alternate-marking], and Hybrid Two-Step (HTS) [I-D.mirsky-ippm-hybrid-two-step], have been developed or proposed. These techniques can provide flow information on the entire forwarding path on a per-packet basis in real-time. The aforementioned on-path telemetry techniques differ from the active and passive OAM schemes in that they directly modify and monitor the user packets in networks so as to achieve high measurement accuracy. Formally, these on-path telemetry techniques can be classified as the

OAM hybrid type I, since they involve "augmentation or modification of the stream of interest, or employment of methods that modify the treatment of the streams", according to [RFC7799]. We name these techniques as "In-situ Flow Information Telemetry" (IFIT).

On-path telemetry is useful for application-aware networking operations, not only in data center and enterprise networks, but also in carrier networks which may cross multiple domains. The techniques can provide benefits for carrier network operators in various scenarios. For example, it is critical for the operators who offer high-bandwidth, latency and loss-sensitive services such as video streaming and online gaming to closely monitor the relevant flows in real-time as the basis for any further optimizations.

This framework document is intended to guide system designers attempting to use the referenced techniques as well as to motivate further work to enhance the telemetry ecosystem. It highlights requirements and challenges, outlines important techniques that are applicable, and provides examples of how these might be applied for critical use cases.

The document scope is discussed in Section 1.3.

1.1. Classification and Modes of On-path Telemetry

The operation of IFIT differs from both active OAM and passive OAM as defined in [RFC7799]. It does not generate any active probe packets or passively observe unmodified user packets. Instead, it modifies selected user packets in order to collect useful information about them. Therefore, the operation is categorized as the hybrid OAM type I method per [RFC7799].

This hybrid OAM type I method can be further partitioned into two modes [passport-postcard]. In the passport mode, each node on the path can add telemetry data to the user packets (i.e., stamps the passport). The accumulated data trace is exported at a configured end node. In the postcard mode, each node directly exports the telemetry data using an independent packet (i.e., sends a postcard) while the user packets are unmodified. It is possible to combine the two modes together in one solution. We call this the hybrid mode.

Figure 1 shows the classification of the on-path telemetry techniques.

Mode	Passport	Postcard	Hybrid
Technique	IOAM Trace IOAM E2E	IOAM DEX PBT-M EAM	Multicast Telemetry HTS

Figure 1: On-path Telemetry Technique Classification

IOAM Trace and E2E options are described in [I-D.ietf-ippm-ioam-data].

EAM is described in [I-D.zhou-ippm-enhanced-alternate-marking].

IOAM DEX option is described in [I-D.ietf-ippm-ioam-direct-export].

PBT-M is described in [I-D.song-ippm-postcard-based-telemetry].

Multicast Telemetry is described in [I-D.ietf-mboned-multicast-telemetry].

HTS is described in [I-D.mirsky-ippm-hybrid-two-step].

The advantages of the passport mode include:

- * It automatically retains the telemetry data correlation along the entire path. The self-describing feature simplifies the data consumption.
- * The on-path data for a packet is only exported once so the data export overhead is low.
- * Only the head and tail nodes of the paths need to be configured for header insertion and removal, so the configuration overhead is low.

The disadvantages of the passport mode include:

- * The telemetry data carried by user packets inflate the packet size, which may be undesirable or prohibitive.
- * Approaches for encapsulating the instruction header and data in transport protocols need to be standardized.
- * Carrying sensitive data along the path is vulnerable to security and privacy breach.

- * If a packet is dropped on the path, the data collected are also lost.

The postcard mode complements the passport mode. The advantages of the postcard mode include:

- * Either there is no packet header overhead (e.g., PBT-M) or the overhead is small and fixed (e.g., IOAM DEX).
- * The encapsulation requirement may be avoided (e.g., PBT-M).
- * The telemetry data can be secured before export.
- * Even if a packet is dropped on the path, the partial data collected are still available.

The disadvantages of the postcard mode include:

- * Telemetry data are spread in multiple postcards so extra effort is needed to correlate the data.
- * Every node exports a postcard for a packet which increases the data export overhead.
- * In case of PBT-M, every node on the path needs to be configured, so the configuration overhead is high.
- * In case of IOAM DEX, the transport encapsulation requirement remains.

The hybrid mode either tailors for some specific application scenario (e.g., Multicast Telemetry) or provides some alternative approach (e.g., HTS). A postcard can be sent per segment of a path or the telemetry data can be carried in a companion packet following each monitored use packet. The hybrid mode combines the advantages of both the passport mode and the postcard mode, but it may incur extra processing complexity.

1.2. Requirements and Challenges

Although on-path telemetry is beneficial, successfully applying such techniques in carrier networks must consider performance, deployability, and flexibility. Specifically, we need to address the following practical deployment challenges:

- * C1: On-path telemetry incurs extra packet processing which may cause stress on the network data plane. The potential impact on the forwarding performance creates an unfavorable "observer

effect" (where the actions of performing on-path telemetry may change the behavior of the traffic being measured). This will not only damage the fidelity of the measurement, but also defy the purpose of the measurement.

- * C2: On-path telemetry can generate a considerable amount of data which may claim too much transport bandwidth and inundate the servers for data collection, storage, and analysis. For example, if the technique is applied to all the traffic, one node may collect a few tens of bytes as telemetry data for each packet. The whole forwarding path might accumulate telemetry data with a size similar to or even exceeding that of the original packet.
- * C3: The collectible data defined currently are essential but limited. This, in turn, limits the management and operational techniques that can be applied. Flexibility and extensibility of data definition, aggregation, acquisition, and filtering, must be considered.
- * C4: Applying only a single underlying on-path telemetry technique may miss some important events or lead to incorrect results. For example, packet drop can cause the loss of the flow telemetry data and the packet drop location and reason remains unknown if only the In-situ OAM trace option is used. A comprehensive solution needs the flexibility to switch between different underlying techniques and adjust the configurations and parameters at runtime. Thus, system-level orchestration is needed.
- * C5: We must provide solutions to support an incremental deployment strategy. That is, we need to support established encapsulation schemes for various predominant protocols such as Ethernet, IPv6, and MPLS with backward compatibility and properly handle various transport tunnels.
- * C6: The development of simplified on-path telemetry primitives and models for configuration and queries is essential. Telemetry models may be utilized via an API-based telemetry service for external applications, for end-to-end performance measurement and application performance monitoring. Standard-based protocols and methods are needed for network configuration and programming, and telemetry data pre-processing and export, to provide interoperability.

1.3. Scope

Following the network telemetry framework discussed in [I-D.ietf-opsawg-ntf], this document focuses on the on-path telemetry, a specific class of data-plane telemetry techniques, and provides a high-level framework which addresses the challenges for deployment listed in Section 1.2, especially in carrier networks.

This document aims to clarify the problem space, essential requirements, and summarizes best practices and general system design considerations. This document provides some examples to show the novel network telemetry applications under the framework.

As an informational document, it describes an open framework with a few key components. The framework does not enforce any specific implementation on each component, neither does it define interfaces (e.g., API, protocol) between components. The choice of underlying on-path telemetry techniques and other implementation details is determined by the application implementer. Therefore, the framework is not a solution specification. It only provides a high-level overview and is not necessarily a mandatory recommendation for on-path telemetry applications.

The standardization of the underlying techniques and interfaces mentioned in this document is undertaken by various working groups. Due to the limited scope and intended status of this document, it has no overlap or conflict with those works.

1.4. Relationship with Network Telemetry Framework (NTF)

[I-D.ietf-opsawg-ntf] describes a Network Telemetry Framework (NTF). One dimension used by NTF to partition network telemetry techniques and systems is based on the three planes in networks (i.e., control plane, management plane, and forwarding plane) and external data sources. IFIT fits in the category of forwarding-plane telemetry and deals with the specific on-path technical branch of the forwarding-plane telemetry.

According to NTF, an on-path telemetry application mainly subscribes to event-triggered or streaming data. The key functional components of IFIT described in Section 2.2 match the general components in NTF with more specific functions. "On-demand Technique Selection and Integration" is an application layer function, matching the "Data Query, Analysis, and Storage" component in NTF; "Flexible Flow, Packet, and Data Selection" matches the "Data Configuration and Subscription" component; "Flexible Data Export" matches the "Data Encoding and Export" component; "Dynamic Network Probe" matches the "Data Generation and Processing" component.

1.5. Glossary

This section defines and explains the acronyms and terms used in this document.

On-path Telemetry: Remotely acquiring performance and behavior data about network flows on a per-packet basis on the packet's forwarding path. The term refers to a class of data-plane telemetry techniques, including IOAM, PBT, EAM, and HTS. Such techniques may need to mark user packets, or insert instruction/metadata into the headers of user packets.

IFIT: In-situ Flow Information Telemetry is a high-level reference framework that shows how network data-plane monitoring and measurement applications can address the deployment challenges of the flow-oriented on-path telemetry techniques.

Reflective Telemetry: The reflective telemetry functions in a dynamic and closed-loop fashion. A new telemetry action is provisioned as a result of self-knowledge acquired through prior telemetry actions.

2. Architectural Concepts and Key Components

To address the challenges mentioned in Section 1.2, a high-level framework which can help to build a workable and efficient on-path telemetry application is presented. In-situ Flow Information Telemetry (IFIT) is dedicated to on-path telemetry data about user and application traffic flows. It covers a class of on-path telemetry techniques and works at a level higher than any specific underlying technique. The framework is comprised of some key functional components (Section 2.2). By assembling these components, IFIT supports reflective telemetry that enables autonomous network operations (Section 2.3).

2.1. Reference Deployment

Figure 2 shows a reference deployment scenario of on-path telemetry.

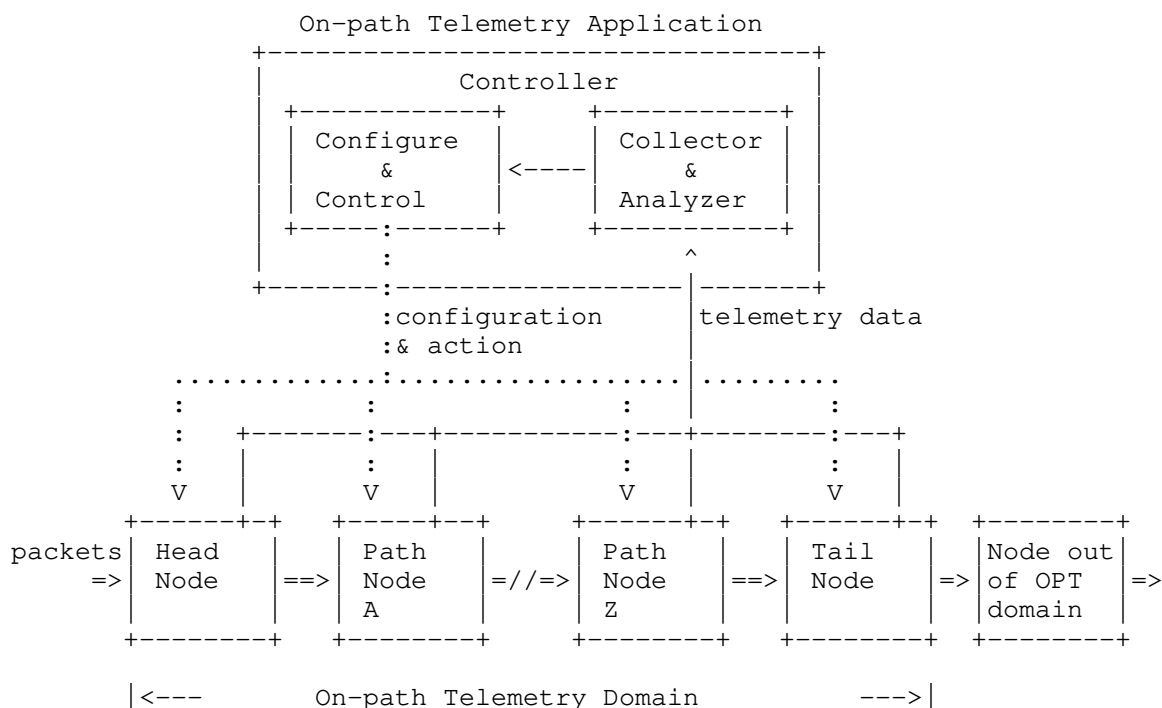


Figure 2: Deployment Scenario

An on-path telemetry application can conduct network data-plane monitoring and measurement tasks over a limited domain [RFC8799] by applying one or more underlying techniques. The application contains multiple elements, including configuring the network nodes and processing the telemetry data. The application usually uses a logically centralized controller for configuring the network nodes in the domain, and collecting and analyzing telemetry data. The configuration determines which underlying technique is used, what telemetry data are of interest, which flows and packets are concerned with, how the telemetry data are collected, etc. The process can be dynamic and interactive: after the telemetry data processing and analyzing, the application may instruct the controller to modify the configuration of the nodes, which affects the future telemetry data collection.

From the system-level view, it is recommended to use standardized configuration and data collection interfaces, regardless of the underlying technique. The specification of these interfaces and the implementation of the controller are out of scope for this document.

The on-path telemetry domain encompasses the head nodes and the end nodes, and may cross multiple network domains. The head nodes are responsible for enabling the on-path telemetry functions and the end nodes are responsible for terminating them. All capable nodes in this domain will be capable of executing the instructed on-path telemetry function. It is important to note that any application must, through configuration and policy, guarantee that any packet with on-path telemetry header and metadata will not leak out of the domain.

The underlying on-path telemetry techniques covered by the IFIT framework can be of any modes discussed in Section 1.1.

2.2. Key Components

The key components of IFIT to address the challenges listed in Section 1.2 are as follows. The components are described in more detail in the sections that follow.

- * Flexible flow, packet, and data selection policy, addressing the challenge C1 described in Section 1;
- * Flexible data export, addressing the challenge C2;
- * Dynamic network probe, addressing C3;
- * On-demand technique selection and integration, addressing C4.

Note that the challenges C5 and C6 are mostly standard-related, and are fundamental to IFIT. We discuss the protocol implications and guidance for solution developers in Section 3.

2.2.1. Flexible Flow, Packet, and Data Selection

In most cases, it is impractical to enable data collection for all the flows and for all the packets in a flow due to the potential performance and bandwidth impact. Therefore, a workable solution usually need to select only a subset of flows and flow packets on which to enable data collection, even though this means the loss of some information and accuracy.

In the data plane, a flow filter like those used for an Access Control List (ACL) provides an ideal means to determine the subset of flows. An application can set a sample rate or probability to a flow to allow only a subset of flow packets to be monitored, collect a different set of data for different packets, and disable or enable data collection on any specific network node. An application can further allow any node to accept or deny the data collection process in full or partially.

Based on these flexible mechanisms, IFIT allows applications to apply flexible flow and data selection policies to suit their requirements. The applications can dynamically change the policies at any time based on the network load, processing capability, focus of interest, and any other criteria.

2.2.1.1. Block Diagram

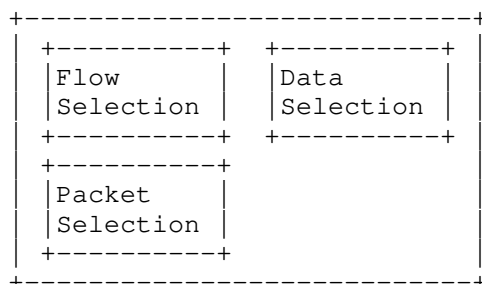


Figure 3: Flexible Flow, Packet, and Data Selection

Figure 3 shows the block diagram of this component. The flow selection block defines the policies to choose target flows for monitoring. Flow has different granularity. A basic flow is defined by 5-tuple IP header fields. Flow can also be aggregated at interface level, tunnel level, protocol level, and so on. The packet selection block defines the policies to choose packets from a target flow. The policy can be either a sampling interval, a sampling probability, or some specific packet signature. The data selection block defines the set of data to be collected. This can be changed on a per-packet or per-flow basis.

2.2.1.2. Example: Sketch-guided Elephant Flow Selection

Network operators are usually more interested in elephant flows which consume more resource and are sensitive to changes in network conditions. A CountMin Sketch [CMSketch] can be used on the data path of the head nodes, which identifies and reports the elephant flows periodically. The controller maintains a current set of elephant flows and dynamically enables the on-path telemetry for only these flows.

2.2.1.3. Example: Adaptive Packet Sampling

Applying on-path telemetry on all packets of the selected flows can still be out of reach. A sample rate should be set for these flows and telemetry should only be enabled on the sampled packets. However, the head nodes have no clue on the proper sampling rate. An overly high rate would exhaust the network resource and even cause packet drops; An overly low rate, on the contrary, would result in the loss of information and inaccuracy of measurements.

An adaptive approach can be used based on the network conditions to dynamically adjust the sampling rate. Every node gives user traffic forwarding higher priority than telemetry data export. In case of network congestion, the telemetry can sense some signals from the data collected (e.g., deep buffer size, long delay, packet drop, and data loss). The controller may use these signals to adjust the packet sampling rate. In each adjustment period (i.e., RTT of the feedback loop), the sampling rate is either decreased or increased in response of the signals. An Additive Increase/Multiplicative Decrease (AIMD) policy similar to the TCP flow control mechanism for rate adjustment can be used.

2.2.2. Flexible Data Export

The flow telemetry data can catch the dynamics of the network and the interactions between user traffic and network. Nevertheless, the data may contain redundancy. It is advisable to remove the redundancy from the data in order to reduce the data transport bandwidth and server processing load.

In addition to efficient export data encoding (e.g., IPFIX [RFC7011] or protobuf (<https://developers.google.com/protocol-buffers/>)), nodes have several other ways to reduce the export data by taking advantage of network device's capability and programmability. Nodes can cache the data and send the accumulated data in batches if the data is not time sensitive. Various deduplication and compression techniques can be applied on the batched data.

From the application perspective, an application may only be interested in some special events which can be derived from the telemetry data. For example, in the case that the forwarding delay of a packet exceeds a threshold, or a flow changes its forwarding path is of interest, it is unnecessary to send the original raw data to the data collecting and processing servers. Rather, IFIT takes advantage of the in-network computing capability of network devices to process the raw data and only push the event notifications to the subscribing applications.

Such events can be expressed as policies. A policy can request data export only on change, on exception, on timeout, or on threshold.

2.2.2.1. Block Diagram

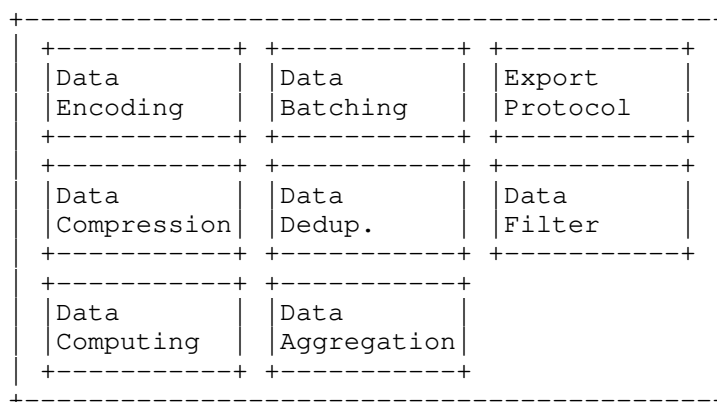


Figure 4: Flexible Data Export

Figure 4 shows the block diagram of this component. The data encoding block defines the method to encode the telemetry data. The data batching block defines the size of batch data buffered at the device side before export. The export protocol block defines the protocol used for telemetry data export. The data compression block defines the algorithm to compress the raw data. The data deduplication block defines the algorithm to remove the redundancy in the raw data. The data filter block defines the policies to filter the needed data. The data computing block defines the policies to preprocess the raw data and generate some new data. The data aggregation block defines the procedure to combine and synthesize the data.

2.2.2.2. Example: Event-based Anomaly Monitor

Network operators are interested in anomalies such as path change, network congestion, and packet drop. Such anomalies are hidden in raw telemetry data (e.g., path trace, timestamp). Such anomalies can be described as events and programmed into the device data plane. Only the triggered events are exported. For example, if a new flow appears at any node, a path change event is triggered; if the packet delay exceeds a predefined threshold in a node, the congestion event is triggered; if a packet is dropped due to buffer overflow, a packet drop event is triggered.

The export data reduction due to such optimization is substantial. For example, given a single 5-hop 10Gbps path, assume a moderate number of 1 million packets per second are monitored, and the telemetry data plus the export packet overhead consume less than 30 bytes per hop. Without such optimization, the bandwidth consumed by the telemetry data can easily exceed 1Gbps (more than 10% of the path bandwidth). When the optimization is used, the bandwidth consumed by the telemetry data is negligible. Moreover, the pre-processed telemetry data greatly simplify the work of data analyzers.

2.2.3. Dynamic Network Probe

Due to limited data plane resource and network bandwidth, it is unlikely one can monitor all the data all the time. On the other hand, the data needed by applications may be arbitrary but ephemeral. It is critical to meet the dynamic data requirements with limited resource.

Fortunately, data plane programmability allows new data probes to be dynamically loaded. These on-demand probes are called Dynamic Network Probes (DNP). DNP is the technique to enable probes for customized data collection in different network planes. When working with an on-path telemetry technique, DNP is loaded into the data plane through incremental programming or configuration. The DNP can effectively conduct data generation, processing, and aggregation.

DNP introduces flexibility and extensibility to IFIT. It can implement the optimizations for export data reduction motioned in the previous section. It can also generate custom data as required by today's and tomorrow's applications.

2.2.3.1. Block Diagram

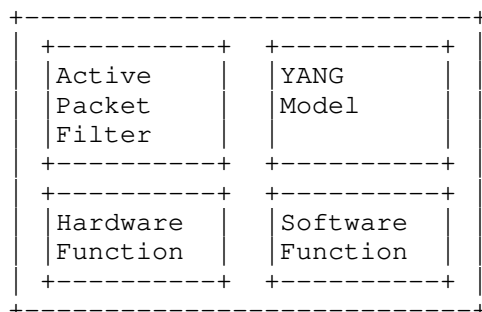


Figure 5: Dynamic Network Probes

Figure 5 shows the block diagram of this component. The active packet filter block is available in most hardware and it defines DNP through dynamically update the packet filtering policies (including flow selection and action). YANG models can be dynamically deployed to enable different data processing and filtering functions. Some hardware allows dynamically loading hardware-based functions into the forwarding path at runtime through mechanisms such as reserved pipelines and function stubs. Dynamically loadable software functions can be implemented in the control processors in capable nodes.

2.2.3.2. Examples

Following are some possible DNPs that can be dynamically deployed to support applications.

On-demand Flow Sketch: A flow sketch is a compact online data structure (usually a variation of multi-hashing table) for approximate estimation of multiple flow properties. It can be used to facilitate flow selection. The aforementioned CountMin Sketch [CMSketch] is such an example. Since a sketch consumes data plane resources, it should only be deployed when actually needed.

Smart Flow Filter: The policies that choose flows and packet sampling rate can change during the lifetime of an application.

Smart Statistics: An application may need to count flows based on different flow granularity or maintain hit counters for selected flow table entries.

Smart Data Reduction: DNP can be used to program the events that conditionally trigger data export.

2.2.4. On-demand Technique Selection and Integration

With multiple underlying data collection and export techniques at its disposal, IFIT can flexibly adapt to different network conditions and different application requirements.

For example, depending on the types of data that are of interest, IFIT may choose either passport or postcard mode to collect the data; if an application needs to track down where the packets are lost, switching from passport mode to postcard mode should be supported.

IFIT can further integrate multiple data plane monitoring and measurement techniques together and present a comprehensive data plane telemetry solution.

Based on the application requirements and the real-time telemetry data analysis results, new configurations and actions can be deployed.

2.2.4.1. Block Diagram

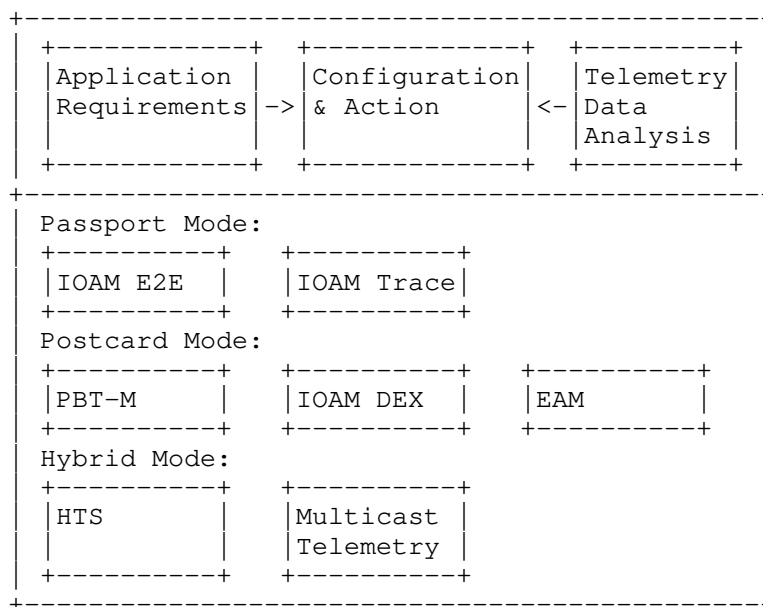


Figure 6: Technique Selection and Integration

Figure 6 shows the block diagram of this component, which lists the candidate on-path telemetry techniques.

Located in the logically centralized controller, this component makes all the control and configuration dynamically to the capable nodes in the domain which will affect the future telemetry data. The configuration and action decisions are based on the inputs from the application requirements and the realtime telemetry data analysis results. Note that here the telemetry data source is not limited to the data plane. The data can come from all the sources mentioned in [I-D.ietf-opsawg-ntf], including external data sources.

2.3. IFIT for Reflective Telemetry

The components described in Section 2.2 can work together to support reflective telemetry, as shown in Figure 7.

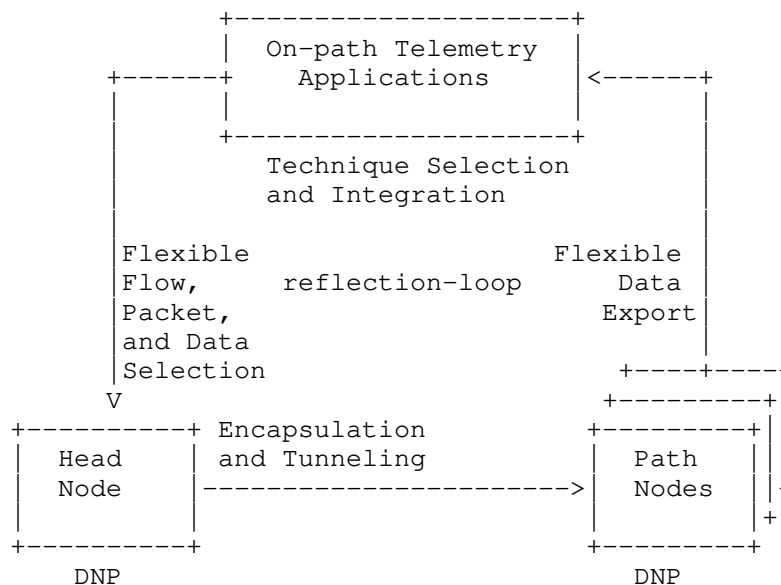


Figure 7: IFIT-based Reflective Telemetry

An application may pick a suite of telemetry techniques based on its requirements and apply an initial technique to the data plane. It then configures the head nodes to decide the initial target flows/packets and telemetry data set, the encapsulation and tunneling scheme based on the underlying network architecture, and the IFIT-capable nodes to decide the initial telemetry data export policy. Based on the network condition and the analysis results of the telemetry data, the application can change the telemetry technique, the flow/data selection policy, and the data export approach in real time without breaking the normal network operation. Many of such dynamic changes can be done through loading and unloading DNP.

The reflective telemetry enabled by the IFIT allows numerous new applications. Two examples are provided below.

2.3.1. Intelligent Multipoint Performance Monitoring

[RFC8889] describes an intelligent performance management based on the network condition. The idea is to split the monitoring network into clusters. The cluster partition that can be applied to every type of network graph and the possibility to combine clusters at different levels enable the so-called Network Zooming. It allows a controller to calibrate the network telemetry, so that it can start without examining in depth and monitor the network as a whole. In case of necessity (packet loss or too high delay), an immediate detailed analysis can be reconfigured. In particular, the controller, that is aware of the network topology, can set up the most suitable cluster partition by changing the traffic filter or activate new measurement points and the problem can be localized with a step-by-step process.

An application on top of the controllers can manage such mechanism, whose dynamic and reflective operations are supported by the IFIT framework.

2.3.2. Intent-based Network Monitoring

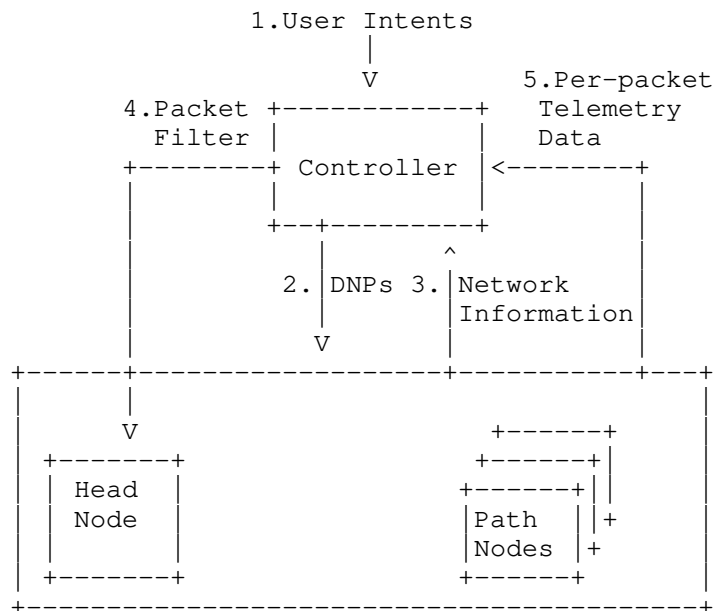


Figure 8: Intent-based Monitoring

In this example, a user can express high level intents for network monitoring. The controller translates an intent and configures the corresponding DNPs in capable nodes which collect necessary network information. Based on the real-time information feedback, the controller runs a local algorithm to determine the suspicious flows. It then deploys specific packet filters to the head node to initiate the high precision per-packet on-path telemetry for these flows.

3. Guidance for Solution Developers

Having a high-level framework covering a class of related techniques promotes a holistic approach for standard development and helps to avoid duplicated efforts and piecemeal solutions that only focus on a specific technique while omitting the compatibility and extensibility issues, which is important to a healthy ecosystem for network telemetry.

A complete IFIT-based solution needs standard interfaces for configuration and data extraction, and standard encapsulation on various transport protocols. It may also need standard API and primitives for application programming and deployment.

[I-D.ietf-ippm-ioam-deployment] summarizes some techniques for encapsulation and data export for IOAM. Solution developers need to consider the aspects set out in the following subsections.

3.1. Encapsulation in Transport Protocols

Since the introduction of IOAM, the IOAM option header encapsulation schemes in various network protocols have been defined (e.g., [I-D.ietf-ippm-ioam-ipv6-options]). Similar encapsulation schemes are needed to cover the other on-path telemetry techniques. Meanwhile, the on-path telemetry header/data encapsulation schemes in some popular protocols, such as MPLS and SRv6, are also needed. PBT-M [I-D.song-ippm-postcard-based-telemetry] does not introduce new headers to the packets so the trouble of encapsulation for a new header is avoided. While there are some proposals which allow new header encapsulation in MPLS packets (e.g., [I-D.song-mpls-extension-header]) or in SRv6 packets (e.g., [I-D.song-spring-siam]), they are still in their infancy stage and require further work. Before standards are available, in a confined domain, pre-standard encapsulation approaches may be applied.

3.2. Tunneling Support

In carrier networks, it is common for user traffic to traverse various tunnels for QoS, traffic engineering, or security. Both the uniform mode and the pipe mode for tunnel support are required and described in [I-D.song-ippm-ioam-tunnel-mode]. The uniform mode treats the nodes in a tunnel uniformly as the nodes outside of the tunnel on a path. In contrast, the pipe mode abstracts all the nodes between the tunnel ingress and egress as a circuit so no nodes in the tunnel is visible to the nodes outside of the tunnel. With such flexibility, the operator can either gain a true end-to-end visibility or apply a hierarchical approach which isolates the monitoring domain between customer and provider.

3.3. Deployment Automation

Standard approaches that automate the function configuration, and capability query and advertisement, could either be deployed in a centralized fashion or a distributed fashion. The draft [I-D.ietf-ippm-ioam-yang] provides a YANG model for IOAM configuration. Similar models needs to be defined for other techniques. It is also helpful to provide standards-based approaches for configuration in various network environments. For example, in Segment Routing (SR) networks, extensions to BGP or Path Computation Element Communication Protocol (PCEP) can be defined to distribute SR policies carrying on-path telemetry information, so that telemetry behavior can be enabled automatically when the SR policy is applied. [I-D.chen-pce-sr-policy-ifit] defines extensions to PCEP to configure SR policies for on-path telemetry. [I-D.ietf-idr-sr-policy-ifit] defines extensions to BGP for the same purpose. Additional capability discovery and dissemination will be needed for other types of networks.

To realize the potential of on-path telemetry, programming and deploying DNP are important. ForCES [RFC5810] is a standard protocol for network device programming, which can be used for DNP deployment. Currently some related works such as [I-D.www-netmod-event-yang] and [I-D.bwd-netmod-eca-framework] have proposed to use YANG models to define the smart policies which can be used to implement DNP. In the future, other approaches for hardware and software-based functions can be development to enhance the programmability and flexibility.

4. Security Considerations

In addition to the specific security issues discussed in each individual document on on-path telemetry, this document considers the overall security issues at the system level. This should serve as a guide to the on-path telemetry application developers and users. General security and privacy considerations for any network telemetry system are also discussed in [I-D.ietf-opsawg-ntf].

Since the on-path telemetry techniques work on the network forwarding plane, the IFIT framework poses some security risks. The important and sensitive information about a network could be exposed to an attacker. Further, the on-path telemetry data might swamp various parts of the network, leading to a possible DoS attack.

Fortunately, security measures can be enforced on various parts of the framework to mitigate such threats. For example, the configuration can filter and rate limit the monitored traffic; encryption and authentication can be applied on the exported telemetry data; different underlying techniques can be chosen to adapt to the different network conditions.

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

Other major contributors of this document include Giuseppe Fioccola, Daniel King, Zhenqiang Li, Zhenbin Li, Tianran Zhou, and James Guichard.

7. Acknowledgments

We thank Diego Lopez, Shwetha Bhandari, Joe Clarke, Adrian Farrel, Frank Brockners, Al Morton, Alex Clemm, Alan DeKok, Benoit Claise, and Warren Kumari for their constructive suggestions for improving this document.

8. References

8.1. Normative References

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

- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.

8.2. Informative References

- [CMSketch] Cormode, G. and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications", 2005, <<http://dx.doi.org/10.1016/j.jalgor.2003.12.001>>.
- [I-D.bwd-netmod-eca-framework]
Boucadair, M., Wu, Q., Wang, M., King, D., and C. Xie, "Framework for Use of ECA (Event Condition Action) in Network Self Management", Work in Progress, Internet-Draft, draft-bwd-netmod-eca-framework-00, 3 November 2019, <<https://www.ietf.org/archive/id/draft-bwd-netmod-eca-framework-00.txt>>.
- [I-D.chen-pce-sr-policy-ifit]
Chen, H., Yuan, H., Zhou, T., Li, W., Fioccola, G., and Y. Wang, "PCEP SR Policy Extensions to Enable IFIT", Work in Progress, Internet-Draft, draft-chen-pce-sr-policy-ifit-02, 10 July 2020, <<https://www.ietf.org/archive/id/draft-chen-pce-sr-policy-ifit-02.txt>>.
- [I-D.herbert-ipv4-eh]
Herbert, T., "IPv4 Extension Headers and Flow Label", Work in Progress, Internet-Draft, draft-herbert-ipv4-eh-01, 2 May 2019, <<https://www.ietf.org/archive/id/draft-herbert-ipv4-eh-01.txt>>.
- [I-D.ietf-idr-sr-policy-ifit]
Qin, F., Yuan, H., Zhou, T., Fioccola, G., and Y. Wang, "BGP SR Policy Extensions to Enable IFIT", Work in Progress, Internet-Draft, draft-ietf-idr-sr-policy-ifit-03, 10 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-idr-sr-policy-ifit-03.txt>>.

[I-D.ietf-ippm-ioam-data]

Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-data-17, 13 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-data-17.txt>>.

[I-D.ietf-ippm-ioam-deployment]

Brockners, F., Bhandari, S., Bernier, D., and T. Mizrahi, "In-situ OAM Deployment", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-deployment-00, 19 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-deployment-00.txt>>.

[I-D.ietf-ippm-ioam-direct-export]

Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F., Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ OAM Direct Exporting", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-direct-export-07, 13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-direct-export-07.txt>>.

[I-D.ietf-ippm-ioam-ipv6-options]

Bhandari, S. and F. Brockners, "In-situ OAM IPv6 Options", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-ipv6-options-07, 6 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-ipv6-options-07.txt>>.

[I-D.ietf-ippm-ioam-yang]

Zhou, T., Guichard, J., Brockners, F., and S. Raghavan, "A YANG Data Model for In-Situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-yang-03, 25 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-yang-03.txt>>.

[I-D.ietf-mboned-multicast-telemetry]

Song, H., McBride, M., Mirsky, G., Mishra, G., Asaeda, H., and T. Zhou, "Multicast On-path Telemetry Solutions", Work in Progress, Internet-Draft, draft-ietf-mboned-multicast-telemetry-02, 4 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-mboned-multicast-telemetry-02.txt>>.

[I-D.ietf-opsawg-ntf]

Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", Work in Progress, Internet-Draft, draft-ietf-opsawg-ntf-13, 3 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-ntf-13.txt>>.

[I-D.li-apn-framework]

Li, Z., Peng, S., Voyer, D., Li, C., Liu, P., Cao, C., Mishra, G., Ebisawa, K., Previdi, S., and J. N. Guichard, "Application-aware Networking (APN) Framework", Work in Progress, Internet-Draft, draft-li-apn-framework-04, 25 October 2021, <<https://www.ietf.org/archive/id/draft-li-apn-framework-04.txt>>.

[I-D.mirsky-ippm-hybrid-two-step]

Mirsky, G., Lingqiang, W., Zhui, G., and H. Song, "Hybrid Two-Step Performance Measurement Method", Work in Progress, Internet-Draft, draft-mirsky-ippm-hybrid-two-step-12, 26 January 2022, <<https://www.ietf.org/archive/id/draft-mirsky-ippm-hybrid-two-step-12.txt>>.

[I-D.song-ippm-ioam-tunnel-mode]

Song, H., Li, Z., Zhou, T., and Z. Wang, "In-situ OAM Processing in Tunnels", Work in Progress, Internet-Draft, draft-song-ippm-ioam-tunnel-mode-00, 27 June 2018, <<https://www.ietf.org/archive/id/draft-song-ippm-ioam-tunnel-mode-00.txt>>.

[I-D.song-ippm-postcard-based-telemetry]

Song, H., Mirsky, G., Filsfils, C., Abdelsalam, A., Zhou, T., Li, Z., Shin, J., and K. Lee, "In-Situ OAM Marking-based Direct Export", Work in Progress, Internet-Draft, draft-song-ippm-postcard-based-telemetry-11, 15 November 2021, <<https://www.ietf.org/archive/id/draft-song-ippm-postcard-based-telemetry-11.txt>>.

[I-D.song-mpls-extension-header]

Song, H., Li, Z., Zhou, T., Andersson, L., and Z. Zhang, "MPLS Extension Header", Work in Progress, Internet-Draft, draft-song-mpls-extension-header-06, 10 January 2022, <<https://www.ietf.org/archive/id/draft-song-mpls-extension-header-06.txt>>.

[I-D.song-spring-siam]

Song, H. and T. Pan, "SRv6 In-situ Active Measurement", Work in Progress, Internet-Draft, draft-song-spring-siam-02, 6 December 2021, <<https://www.ietf.org/archive/id/draft-song-spring-siam-02.txt>>.

[I-D.wwx-netmod-event-yang]

Wu, Q., Bryskin, I., Birkholz, H., Liu, X., and B. Claise, "A YANG Data model for ECA Policy Management", Work in Progress, Internet-Draft, draft-wwx-netmod-event-yang-10, 1 November 2020, <<https://www.ietf.org/archive/id/draft-wwx-netmod-event-yang-10.txt>>.

[I-D.zhou-ippm-enhanced-alternate-marking]

Zhou, T., Fioccola, G., Liu, Y., Lee, S., Cociglio, M., and W. Li, "Enhanced Alternate Marking Method", Work in Progress, Internet-Draft, draft-zhou-ippm-enhanced-alternate-marking-08, 4 January 2022, <<https://www.ietf.org/archive/id/draft-zhou-ippm-enhanced-alternate-marking-08.txt>>.

[passport-postcard]

Handigol, N., Heller, B., Jeyakumar, V., Mazieres, D., and N. McKeown, "Where is the debugger for my software-defined network?", 2012, <<https://doi.org/10.1145/2342441.2342453>>.

[RFC5810]

Doria, A., Ed., Hadi Salim, J., Ed., Haas, R., Ed., Khosravi, H., Ed., Wang, W., Ed., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, DOI 10.17487/RFC5810, March 2010, <<https://www.rfc-editor.org/info/rfc5810>>.

[RFC7011]

Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.

[RFC8889]

Fioccola, G., Ed., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8889, DOI 10.17487/RFC8889, August 2020, <<https://www.rfc-editor.org/info/rfc8889>>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <<https://www.rfc-editor.org/info/rfc8993>>.

Authors' Addresses

Haoyu Song
Futurewei
2330 Central Expressway
Santa Clara,
United States of America
Email: haoyu.song@futurewei.com

Fengwei Qin
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing, 100032
P.R. China
Email: qinfengwei@chinamobile.com

Huanan Chen
China Telecom
Email: chenhuan6@chinatelecom.cn

Jaehwan Jin
LG U+
South Korea
Email: daenamul@lguplus.co.kr

Jongyoon Shin
SK Telecom
South Korea
Email: jongyoon.shin@sk.com

Networking Working Group
Internet-Draft
Intended status: Informational
Expires: April 28, 2020

Q. Wu, Ed.
Huawei
M. Boucadair, Ed.
Orange
D. Lopez
Telefonica I+D
C. Xie
China Telecom
L. Geng
China Mobile
October 26, 2019

A Framework for Automating Service and Network Management with YANG
draft-wu-model-driven-management-virtualization-07

Abstract

Data models for service and network management provides a programmatic approach for representing (virtual) services or networks and deriving (1) configuration information that will be communicated to network and service components that are used to build and deliver the service and (2) state information that will be monitored and tracked. Indeed, data models can be used during various phases of the service and network management life cycle, such as service instantiation, service provisioning, optimization, monitoring, and diagnostic. Also, data models are instrumental in the automation of network management. They also provide closed-loop control for the sake of adaptive and deterministic service creation, delivery, and maintenance.

This document provides a framework that describes and discusses an architecture for service and network management automation that takes advantage of YANG modeling technologies. This framework is drawn from a network provider perspective irrespective of the origin of a data module; it can accommodate even modules that are developed outside the IETF.

The document aims to exemplify an approach that specifies the journey from technology-agnostic services to technology-specific actions.

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 28, 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	5
3. Architectural Concepts & Goals	5
3.1. Data Models: Layering and Representation	5
3.2. Automation of Service Delivery Procedures	6
3.3. Service Fullfillment Automation	7
3.4. YANG Modules Integration	7
4. Architecture Overview	8
4.1. Service Lifecycle Management Procedure	9
4.1.1. Service Exposure	10
4.1.2. Service Creation/Modification	10
4.1.3. Service Optimization	10
4.1.4. Service Diagnosis	11
4.1.5. Service Decommission	11
4.2. Service Fullfillment Management Procedure	11
4.2.1. Intended Configuration Provision	11
4.2.2. Configuration Validation	12
4.2.3. Performance Monitoring	12
4.2.4. Fault Diagnostic	13
4.3. Multi-layer/Multi-domain Service Mapping	13
4.4. Service Decomposing	13

5. YANG Data Model Integration Examples	13
5.1. L3VPN Service Delivery	13
5.2. VN Lifecycle Management Example	15
6. Security Considerations	16
7. IANA Considerations	16
8. Acknowledgements	16
9. Contributors	16
10. Informative References	17
Appendix A. Layered YANG Modules Example Overview	25
A.1. Service Models: Definition and Samples	25
A.2. Network Models: Definitions and Samples	26
A.3. Device Models: Definitions and Samples	29
A.3.1. Model Composition	30
A.3.2. Device Models: Definitions and Samples	30
Authors' Addresses	33

1. Introduction

The service management system usually comprises service activation/provision and service operation. Current service delivery procedures, from the processing of customer's requirements and order to service delivery and operation, typically assume the manipulation of data sequentially into multiple OSS/BSS applications that may be managed by different departments within the service provider's organization (e.g., billing factory, design factory, network operation center, etc.). In addition, many of these applications have been developed in-house over the years and operating in a silo mode:

- o The lack of standard data input/output (i.e., data model) also raises many challenges in system integration and often results in manual configuration tasks.
- o Secondly, many current service fulfillment system might have limited visibility to the network and therefore have slow response to the network changes.

Software Defined Networking (SDN) becomes crucial to address these challenges. SDN techniques [RFC7149] are meant to automate the overall service delivery procedures and typically rely upon (standard) data models that are used to not only reflect service providers' savoir-faire but also to dynamically instantiate and enforce a set of (service-inferred) policies that best accommodate what has been (contractually) defined (and possibly negotiated) with the customer. [RFC7149] provides a first tentative to rationalize that service provider's view on the SDN space by identifying concrete technical domains that need to be considered and for which solutions can be provided:

- o Techniques for the dynamic discovery of topology, devices, and capabilities, along with relevant information and data models that are meant to precisely document such topology, devices, and their capabilities.
- o Techniques for exposing network services [RFC8309] and their characteristics.
- o Techniques used by service-requirement-derived dynamic resource allocation and policy enforcement schemes, so that networks can be programmed accordingly.
- o Dynamic feedback mechanisms that are meant to assess how efficiently a given policy (or a set thereof) is enforced from a service fulfillment and assurance perspective.

Models are key for each of these technical items. Service and network management automation is an important step to improve the agility of network operations and infrastructures. Models are also important to ease integrating multi-vendor solutions.

YANG module developers have taken both top-down and bottom-up approaches to develop modules [RFC8199], and also to establish a mapping between network technology and customer requirements on the top or abstracting common construct from various network technologies on the bottom. At the time of writing this document (2019), there are many data models including configuration and service models that have been specified or are being specified by the IETF. They cover many of the networking protocols and techniques. However, how these models work together to configure a device, manage a set of devices involved in a service, or even provide a service is something that is not currently documented either within the IETF or other SDOs (e.g., MEF).

This document provides a framework that describes and discusses an architecture for service and network management automation that takes advantage of YANG modeling technologies and investigates how different layer YANG data models interact with each other (e.g., service mapping, model composing) in the context of service delivery and fulfillment.

This framework is drawn from a network provider perspective irrespective of the origin of a data module; it can accommodate even modules that are developed outside the IETF.

The document also identifies a list of use cases to exemplify the proposed approach, but it does not claim to be exhaustive.

2. Terminology

The following terms are defined in [RFC8309][RFC8199] and are not redefined here:

- o Network Operator
- o Customer
- o Service
- o Data Model
- o Service Model
- o Network Element Module

The document makes use of the following terms:

Network Model: The Network Model describes network level abstraction or various aspects of a network infrastructure, including devices and their subsystems, and relevant protocols operating at the link and network layers across multiple devices. It can be used by a network operator to allocate the resource(e.g., tunnel resource, topology resource) for the service or schedule the resource to meet the service requirements define in the Service Model.

Device Model: Network Element YANG data module described in [RFC8199].

3. Architectural Concepts & Goals

3.1. Data Models: Layering and Representation

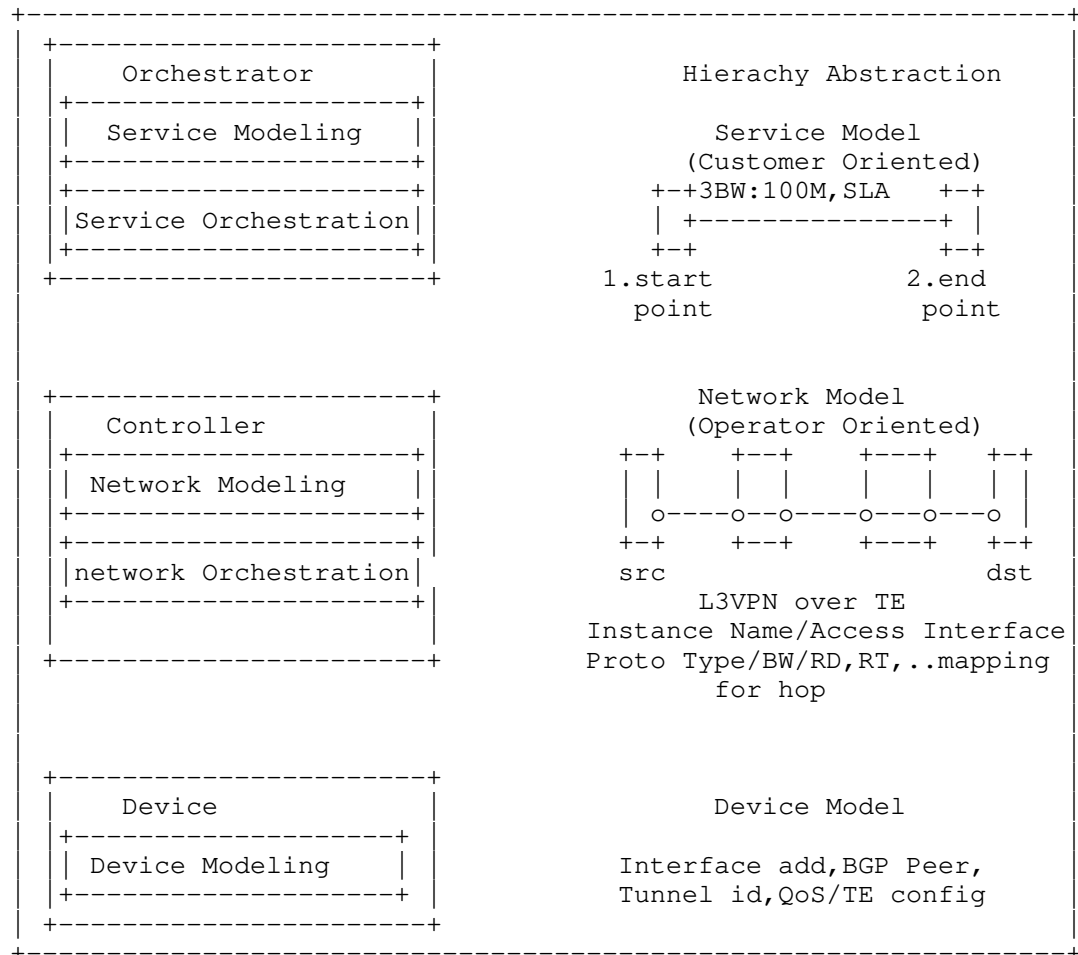
As described in [RFC8199], layering of modules allows for better reusability of lower-layer modules by higher-level modules while limiting duplication of features across layers.

The data modules developed by IETF can be classified into service level, network level and device level modules. Different service model at service level may rely on the same set of network level or device level models. Service models usually follow top down approach and are mostly customer-facing modules providing a common model construct for higher level network services, which can be further mapped to network technology-specific modules at lower layer.

Network level modules are mainly network resource-facing modules and describe various aspects of a network infrastructure, including

devices and their subsystems, and relevant protocols operating at the link and network layers across multiple devices (e.g., Network topology and TE Tunnel modules).

Device level modules usually follow a bottom-up approach and are mostly technology-specific modules used to realize a service.



Layering and representation

3.2. Automation of Service Delivery Procedures

To dynamically offer and deliver service offerings, Service level modules can be used by an operator. One or more monolithic Service modules can be used in the context of a composite service activation

request (e.g., delivery of a caching infrastructure over a VPN). Such modules are used to feed a decision-making intelligence to adequately accommodate customer's needs.

Also, such modules may be used jointly with services that require dynamic invocation. An example is provided by the service modules defined by the DOTS WG to dynamically trigger requests to handle DDoS attacks [I-D.ietf-dots-signal-channel][I-D.ietf-dots-data-channel].

Network level modules can be derived from service level modules and used to provision, monitor, instantiate the service, and provide lifecycle management of network resources (e.g., expose network resources to customers or operators to provide service fulfillment and assurance and allow customers or operators to dynamically adjust the network resources based on service requirements as described in service level modules and the current network performance information described in the telemetry modules).

3.3. Service Fullfillment Automation

To operate the service, Device level modules derived from Service level modules or Network level modules can be used to provision each involved network function/device with the proper configuration information, and operate the network based on service requirements as described in the Service level module(s).

In addition, the operational state including configuration that is in effect together with statistics should be exposed to upper layers to provide better network visibility (and assess to what extent the derived low level modules are consistent with the upper level inputs).

Note that it is important to correlate telemetry data with configuration data to be used for closed loops at the different stages of service delivery, from resource allocation to service operation, in particular.

3.4. YANG Modules Integration

To support top-down service delivery, YANG modules at different level or at the same level need to be integrated together to enable function, feature in the network device and get network setup. For example, the service parameters captured in service level modules need to be decomposed into a set of (configuration/notification) parameters that may be specific to one or more technologies; these technology-specific parameters are grouped together to define technology-specific device level models or network level models.

In addition, these technology-specific device level models or network level models can be further integrated with each other using schema mount mechanism [RFC8528] to provision each involved network function/device or each involved administrative domain to support newly added module or features. A collection of device models integrated together can be loaded and validated during implementation time.

Policies provide a higher layer of abstraction. Policy models can be defined at service level, network level, or device level to provide policy-based management and telemetry automation, e.g., telemetry data can trigger a new policy that captures new network service requirements.

Performance measurement telemetry can be used to provide service assurance at service level or at the network level. Performance measurement telemetry model can tie with network level model or service level model to monitor network performance or service level agreement.

4. Architecture Overview

The architectural considerations described in Section 3 lead to the architecture described in this section and illustrated in Figure 1.

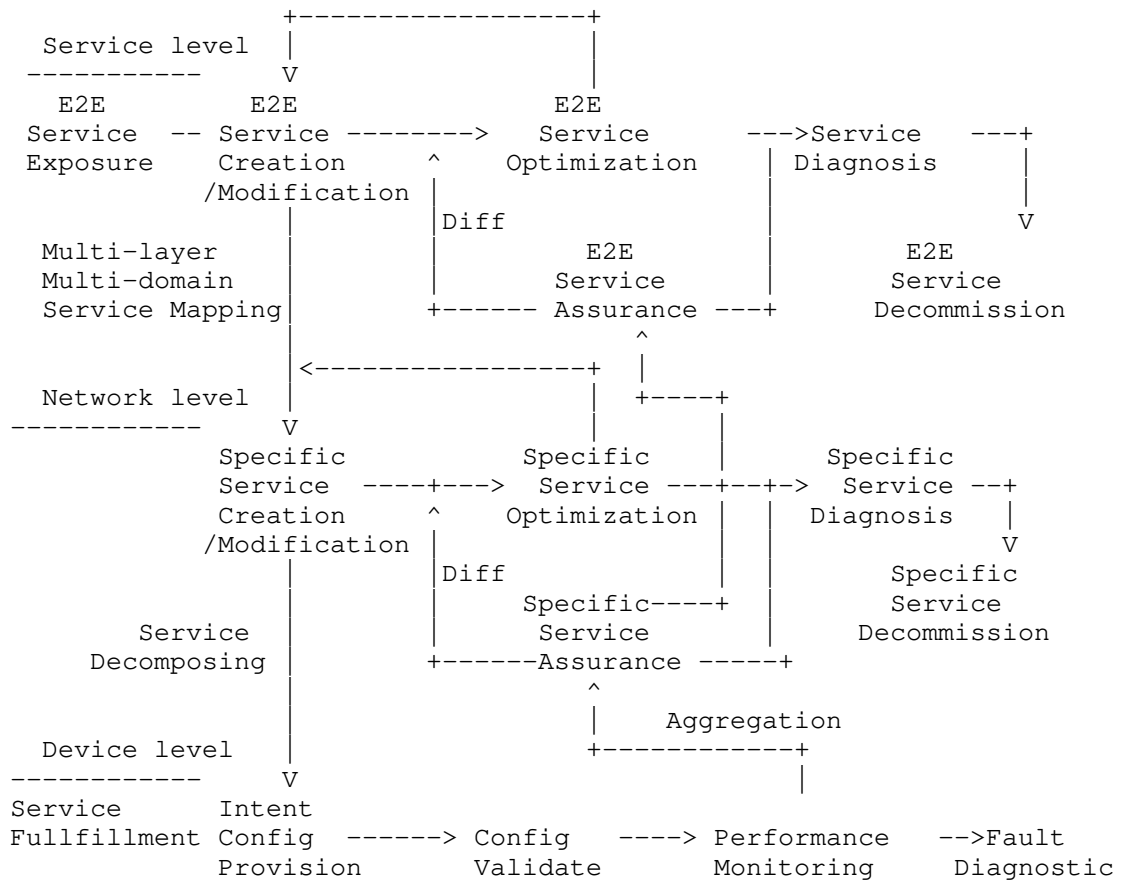


Figure 1: Service and Network Lifecycle Management

4.1. Service Lifecycle Management Procedure

Service lifecycle management includes end to end service lifecycle management at the service level and specific network lifecycle management at the network level. The end-to-end service lifecycle management is multi-domain or multi-layer service management while specific service lifecycle management is domain specific or layer specific service lifecycle management.

- o Note: Clarify what is meant by "domain".

4.1.1.1. Service Exposure

A service in the context of this document (sometimes called a Network Service) is some form of connectivity between customer sites and the Internet or between customer sites across the network operator's network and across the Internet.

Service exposure is used to capture services offered to customers (ordering and order handling). One typical example is that a customer can use a L3SM service model to request L3VPN service by providing the abstract technical characterization of the intended service between customer sites.

Service model catalogs can be created along to expose the various services and the information needed to invoke/order a given service.

4.1.1.2. Service Creation/Modification

A customer is (usually) unaware of the technology that the network operator has available to deliver the service, so the customer does not make requests specific to the underlying technology but is limited to making requests specific to the service that is to be delivered. This service request can be issued using the service model.

The service orchestrator/management system maps such service request to its view. This view can be described as a network model and this mapping may include a choice of which networks and technologies to use depending on which service features have been requested.

In addition, a customer may require to change underlying network infrastructure to adapt to new customer's needs and service requirements. This service modification can be issued in the same service model used by the service request.

4.1.1.3. Service Optimization

Service optimization is a technique that gets the configuration of the network updated due to network change, incident mitigation, or new service requirements. One typical example is once the tunnel or the VPN is setup, Performance monitoring information or telemetry information per tunnel or per VPN can be collected and fed into the management system, if the network performance doesn't meet the service requirements, the management system can create new VPN policies capturing network service requirements and populate them into the network.

Both network performance information and policies can be modelled using YANG. With Policy-based management, self-configuration and self-optimization behavior can be specified and implemented.

4.1.4. Service Diagnosis

Operations, Administration, and Maintenance (OAM) are important networking functions for service diagnosis that allow operators to:

- o monitor network communications (i.e., reachability verification and Continuity Check)
- o troubleshoot failures (i.e., fault verification and localization)
- o monitor service-level agreements and performance (i.e., performance management)

When the network is down, service diagnosis should be in place to pinpoint the problem and provide recommendation (or instructions) for the network recovery.

The service diagnosis information can be modelled as technology-independent RPC operations for OAM protocols and technology-independent abstraction of key OAM constructs for OAM protocols [RFC8531][RFC8533]. These models can provide consistent configuration, reporting, and presentation for the OAM mechanisms used to manage the network.

4.1.5. Service Decommission

Service decommission allow the customer to stop the service and remove the service from active status and release the network resource that is allocated to the service. Customer can also use the service model to withdraw the subscription to a service.

4.2. Service Fullfillment Management Procedure

4.2.1. Intended Configuration Provision

Intended configuration at the device level is derived from network model at the network level or service model at the service level and represents the configuration that the system attempts to apply. Take L3SM service model as an example, to deliver a L3VPN service, we need to map L3VPN service view defined in Service model into detailed intended configuration view defined by specific configuration models for network elements, configuration information includes:

- o VRF definition, including VPN Policy expression

- o Physical Interface
- o IP layer (IPv4, IPv6)
- o QoS features such as classification, profiles, etc.
- o Routing protocols: support of configuration of all protocols listed in the document, as well as routing policies associated with those protocols.
- o Multicast Support
- o NAT or address sharing
- o Security function

This specific configuration models can be used to configure PE and CE devices within the site, e.g., A BGP policy model can be used to establish VPN membership between sites and VPN Service Topology.

4.2.2. Configuration Validation

Configuration validation is used to validate intended configuration and ensure the configuration take effect. For example, a customer creates an interface "et-0/0/0" but the interface does not physically exist at this point, then configuration data appears in the <intended> status but does not appear in <operational> datastore.

4.2.3. Performance Monitoring

When configuration is in effect in the device, <operational> datastore holds the complete operational state of the device including learned, system, default configuraton and system state. However the configurations and state of a particular device does not have the visibility to the whole network or information of the flow packets are going to take through the entire network. Therefore it becomes more difficult to operate the network without understanding the current status of the network.

The management system should subscribe to updates of a YANG datastore in all the network devices for performance monitoring purpose and build full topological visibility to the network by aggregating and filtering these operational state from different sources.

4.2.4. Fault Diagnostic

When configuration is in effect in the device, some device may be misconfigured (e.g., device links are not consistent on both sides of the network connection), network resources be misallocated and services may be negatively affected without knowing what is going on in the network.

Technology-dependent nodes and remote procedure call (RPC) commands are defined in technology-specific YANG data models which can use and extend the base model described in Section 4.1.4 can be used to deal with these challenges.

These RPC command received in the technology dependent node can be used to trigger technology specific OAM message exchange for fault verification and fault isolation, e.g., TRILL Multicast Tree Verification (MTV) RPC command [I-D.ietf-trill-yang-oam] can be used to trigger Multi-Destination Tree Verification Message defined in [RFC7455] to verify TRILL distribution tree integrity.

4.3. Multi-layer/Multi-domain Service Mapping

Multi-layer/Multi-domain Service Mapping allow you map end to end abstract view of the service segmented at different layer or different administrative domain into domain specific view. One example is to map service parameters in L3VPN service model into configuration parameters such as RD, RT, and VRF in L3VPN network model. Another example is to map service parameters in L3VPN service model into TE tunnel parameter (e.g., Tunnel ID) in TE model and VN parameters (e.g., AP list, VN member) in TEAS VN model [I-D.ietf-teas-actn-vn-yang].

4.4. Service Decomposing

Service Decomposing allows to decompose service model at the service level or network model at the network level into a set of device/function models at the device level. These device models may be tied to specific device type or classified into a collection of related YANG modules based on service type and feature offered and load at the implementation time before configuration is loaded and validated.

5. YANG Data Model Integration Examples

5.1. L3VPN Service Delivery

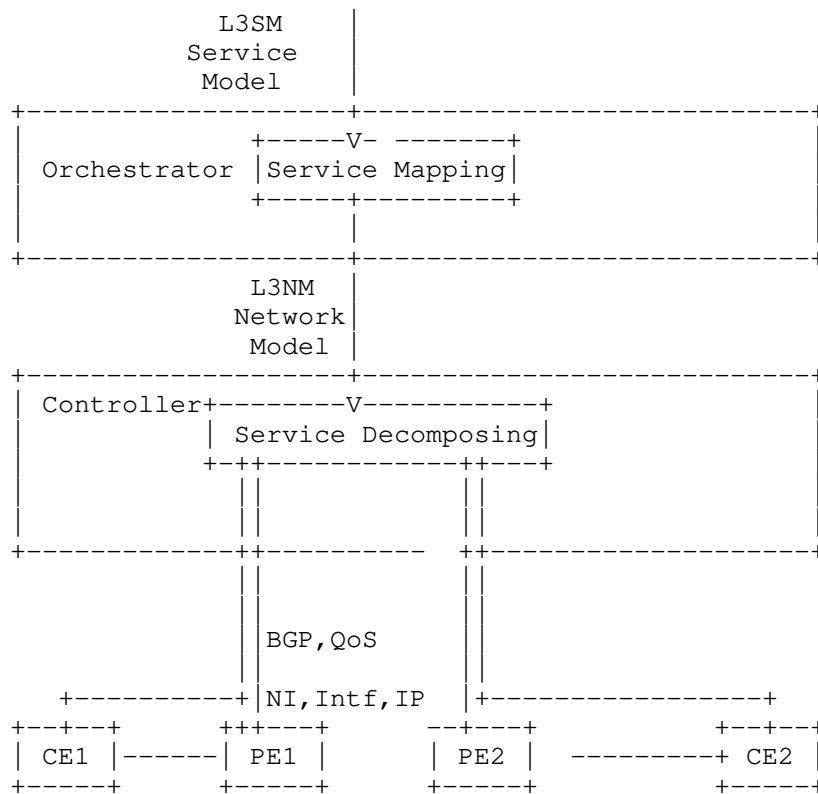


Figure 2: L3VPN Service Delivery Example

In reference to Figure 2, the following steps are performed to deliver the L3VPN service within the network management automation architecture defined in this document:

1. Customer Requests to create two sites based on L3SM Service model with each having one network access connectivity:

Site A: Network-Access A, Bandwidth=20M, for class "foo",
guaranteed-bw-percent = 10, One-Way-Delay=70 msec

Site B: Network-Access B, Bandwidth=30M, for class "foo1",
guaranteed-bw-percent = 15, One-Way-Delay=60 msec

2. The Orchestrator extracts the service parameters from the L3SM model. Then, it uses them as input to translate them into an orchestrated configuration of network elements (e.g., RD, RT, VRF, etc.) that is part of the L3NM network model.

3. The Controller takes orchestrated configuration parameters in the L3NM network model and translates them into orchestrated configuration of network elements that is part of BGP model, QoS model, Network Instance model, IP management model, interface model, etc.

5.2. VN Lifecycle Management Example

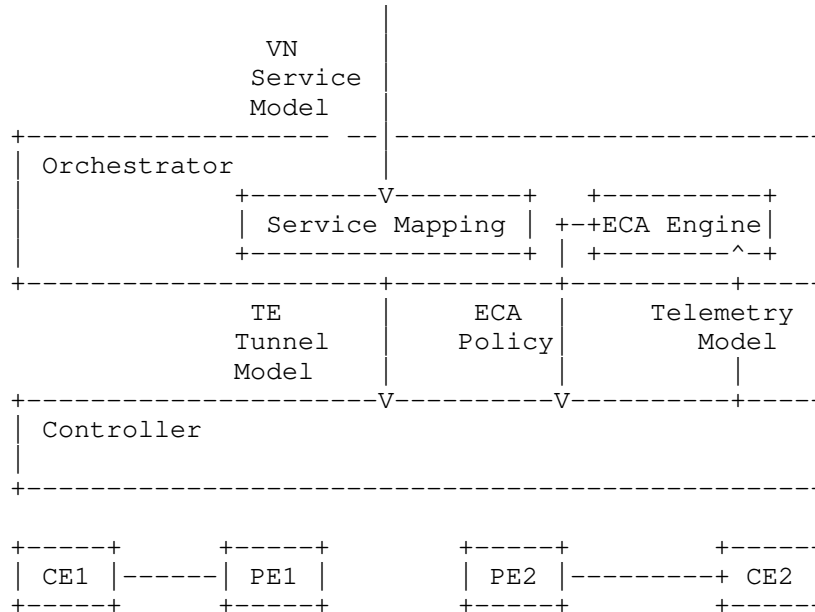


Figure 3

In reference to Figure 3, the following steps are performed to deliver the VN service within the network management automation architecture defined in this document:

1. Customer requests to create 'VN' based on Access point, association between VN and Access point, VN member defined in the VN YANG module.
2. The orchestrator creates the single abstract node topology based on the information captured in an VN YANG module.
3. The Customer exchanges connectivity-matrix on abstract node and explicit path using TE topology model with the orchestrator. This information can be used to instantiate VN and setup tunnels between source and destination endpoints.

4. The telemetry which augments the TEAS VN model and corresponding TE Tunnel model can be used to notify all the parameter changes and network performance change related to VN topology or Tunnel [I-D.ietf-teas-actn-pm-telemetry-autonomics]. This information can be further used as input to ECA engine in the orchestrator and generate ECA policy model to optimize the network.

6. Security Considerations

Security considerations specific to each of the technologies and protocols listed in the document are discussed in the specification documents of each of these techniques.

(Potential) security considerations specific to this document are listed below:

- o Create forwarding loops by mis-configuring the underlying network.
- o Leak sensitive information: special care should be considered when translating between the various layers introduced in the document.
- o ...tbc

7. IANA Considerations

There are no IANA requests or assignments included in this document.

8. Acknowledgements

Thanks to Joe Clark, Greg Mirsky, and Shunsuke Homma for the review.

9. Contributors

Christian Jacquenet
Orange
Rennes, 35000
France
Email: Christian.jacquenet@orange.com

Luis Miguel Contreras Murillo
Telifonica

Email: luismiguel.contrerasmurillo@telefonica.com

Oscar Gonzalez de Dios
Telefonica
Madrid
ES

Email: oscar.gonzalezdedios@telefonica.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf.bri@chinatelecom.cn

Weiqiang Cheng
China Mobile

Email: chengweiqiang@chinamobile.com

Young Lee
Sung Kyun Kwan University

Email: younglee.tx@gmail.com

10. Informative References

[I-D.arkko-arch-virtualization]

Arkko, J., Tantsura, J., Halpern, J., and B. Varga,
"Considerations on Network Virtualization and Slicing",
draft-arkko-arch-virtualization-01 (work in progress),
March 2018.

[I-D.asechoud-netmod-diffserv-model]

Choudhary, A., Shah, S., Jethanandani, M., Liu, B., and N.
Strahle, "YANG Model for Diffserv", draft-asechoud-netmod-
diffserv-model-03 (work in progress), June 2015.

- [I-D.claccla-netmod-model-catalog]
Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-claccla-netmod-model-catalog-03 (work in progress), April 2018.
- [I-D.homma-slice-provision-models]
Homma, S., Nishihara, H., Miyasaka, T., Galis, A., OV, V., Lopez, D., Contreras, L., Ordonez-Lucena, J., Martinez-Julia, P., Qiang, L., Rokui, R., Ciavaglia, L., and X. Foy, "Network Slice Provision Models", draft-homma-slice-provision-models-01 (work in progress), July 2019.
- [I-D.ietf-bess-evpn-yang]
Brissette, P., Shah, H., Hussain, I., Tiruveedhula, K., and J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-07 (work in progress), March 2019.
- [I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.
- [I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-04 (work in progress), October 2018.
- [I-D.ietf-bfd-yang]
Rahman, R., Zheng, L., Jethanandani, M., Networks, J., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-17 (work in progress), August 2018.
- [I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-09 (work in progress), April 2019.
- [I-D.ietf-ccamp-flexigrid-media-channel-yang]
Madrid, U., Perdices, D., Lopezalvarez, V., Dios, O., King, D., Lee, Y., and G. Galimberti, "YANG data model for Flexi-Grid media-channels", draft-ietf-ccamp-flexigrid-media-channel-yang-02 (work in progress), March 2019.

- [I-D.ietf-ccamp-flexigrid-yang]
Madrid, U., Perdices, D., Lopezalvarez, V., King, D., and Y. Lee, "YANG data model for Flexi-Grid Optical Networks", draft-ietf-ccamp-flexigrid-yang-04 (work in progress), July 2019.
- [I-D.ietf-ccamp-llcsm-yang]
Lee, Y., Lee, K., Zheng, H., Dhody, D., Dios, O., and D. Ceccarelli, "A YANG Data Model for L1 Connectivity Service Model (L1CSM)", draft-ietf-ccamp-llcsm-yang-10 (work in progress), September 2019.
- [I-D.ietf-ccamp-mw-yang]
Ahlberg, J., Ye, M., Li, X., Spreafico, D., and M. Vaupotic, "A YANG Data Model for Microwave Radio Link", draft-ietf-ccamp-mw-yang-13 (work in progress), November 2018.
- [I-D.ietf-ccamp-otn-topo-yang]
Zheng, H., Guo, A., Busi, I., Sharma, A., Liu, X., Belotti, S., Xu, Y., Wang, L., and O. Dios, "A YANG Data Model for Optical Transport Network Topology", draft-ietf-ccamp-otn-topo-yang-08 (work in progress), September 2019.
- [I-D.ietf-ccamp-otn-tunnel-model]
Zheng, H., Busi, I., Belotti, S., Lopezalvarez, V., and Y. Xu, "OTN Tunnel YANG Model", draft-ietf-ccamp-otn-tunnel-model-08 (work in progress), October 2019.
- [I-D.ietf-ccamp-wson-tunnel-model]
Lee, Y., Zheng, H., Guo, A., Lopezalvarez, V., King, D., Yoon, B., and R. Vilata, "A Yang Data Model for WSON Tunnel", draft-ietf-ccamp-wson-tunnel-model-04 (work in progress), September 2019.
- [I-D.ietf-dots-data-channel]
Boucadair, M. and R. K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification", draft-ietf-dots-data-channel-31 (work in progress), July 2019.
- [I-D.ietf-dots-signal-channel]
K, R., Boucadair, M., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", draft-ietf-dots-signal-channel-37 (work in progress), July 2019.

- [I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-07 (work in progress), October 2019.
- [I-D.ietf-ippm-stamp-yang]
Mirsky, G., Xiao, M., and W. Luo, "Simple Two-way Active Measurement Protocol (STAMP) Data Model", draft-ietf-ippm-stamp-yang-05 (work in progress), October 2019.
- [I-D.ietf-ippm-twamp-yang]
Civil, R., Morton, A., Rahman, R., Jethanandani, M., and K. Pentikousis, "Two-Way Active Measurement Protocol (TWAMP) Data Model", draft-ietf-ippm-twamp-yang-13 (work in progress), July 2018.
- [I-D.ietf-mpls-base-yang]
Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A YANG Data Model for MPLS Base", draft-ietf-mpls-base-yang-11 (work in progress), September 2019.
- [I-D.ietf-pim-igmp-ml-d-snooping-yang]
Zhao, H., Liu, X., Liu, Y., Sivakumar, M., and A. Peter, "A Yang Data Model for IGMP and MLD Snooping", draft-ietf-pim-igmp-ml-d-snooping-yang-08 (work in progress), June 2019.
- [I-D.ietf-pim-igmp-ml-d-yang]
Liu, X., Guo, F., Sivakumar, M., McAllister, P., and A. Peter, "A YANG Data Model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-ml-d-yang-15 (work in progress), June 2019.
- [I-D.ietf-pim-yang]
Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG Data Model for Protocol Independent Multicast (PIM)", draft-ietf-pim-yang-17 (work in progress), May 2018.
- [I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

- [I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", draft-ietf-rtgwg-policy-model-07 (work in progress), September 2019.
- [I-D.ietf-software-iftunnel]
Boucadair, M., Farrer, I., and R. Asati, "Tunnel Interface Types YANG Module", draft-ietf-software-iftunnel-07 (work in progress), June 2019.
- [I-D.ietf-software-yang]
Farrer, I. and M. Boucadair, "YANG Modules for IPv4-in-IPv6 Address plus Port (A+P) Softwires", draft-ietf-software-yang-16 (work in progress), January 2019.
- [I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Lindem, A., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", draft-ietf-spring-sr-yang-13 (work in progress), July 2019.
- [I-D.ietf-supra-generic-policy-data-model]
Halpern, J. and J. Strassner, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supra-generic-policy-data-model-04 (work in progress), June 2017.
- [I-D.ietf-teas-actn-vn-yang]
Lee, Y., Dhody, D., Ceccarelli, D., Bryskin, I., and B. Yoon, "A Yang Data Model for VN Operation", draft-ietf-teas-actn-vn-yang-06 (work in progress), July 2019.
- [I-D.ietf-teas-sf-aware-topo-model]
Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L., Ceccarelli, D., and J. Tantsura, "SF Aware TE Topology YANG Model", draft-ietf-teas-sf-aware-topo-model-03 (work in progress), March 2019.
- [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", draft-ietf-teas-te-service-mapping-yang-02 (work in progress), September 2019.
- [I-D.ietf-teas-yang-l3-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Layer 3 TE Topologies", draft-ietf-teas-yang-l3-te-topo-05 (work in progress), July 2019.

- [I-D.ietf-teas-yang-path-computation]
Busi, I. and S. Belotti, "Yang model for requesting Path Computation", draft-ietf-teas-yang-path-computation-06 (work in progress), July 2019.
- [I-D.ietf-teas-yang-rsvp-te]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Bryskin, I., and H. Shah, "A YANG Data Model for RSVP-TE Protocol", draft-ietf-teas-yang-rsvp-te-07 (work in progress), July 2019.
- [I-D.ietf-teas-yang-sr-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and S. Litkowski, "YANG Data Model for SR and SR TE Topologies", draft-ietf-teas-yang-sr-te-topo-05 (work in progress), July 2019.
- [I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-21 (work in progress), April 2019.
- [I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-22 (work in progress), June 2019.
- [I-D.ietf-trill-yang-oam]
Kumar, D., Senevirathne, T., Finn, N., Salam, S., Xia, L., and H. Weiguo, "YANG Data Model for TRILL Operations, Administration, and Maintenance (OAM)", draft-ietf-trill-yang-oam-05 (work in progress), March 2017.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <<https://www.rfc-editor.org/info/rfc4664>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.

- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <<https://www.rfc-editor.org/info/rfc7276>>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <<https://www.rfc-editor.org/info/rfc7297>>.
- [RFC7455] Senevirathne, T., Finn, N., Salam, S., Kumar, D., Eastlake 3rd, D., Aldrin, S., and Y. Li, "Transparent Interconnection of Lots of Links (TRILL): Fault Management", RFC 7455, DOI 10.17487/RFC7455, March 2015, <<https://www.rfc-editor.org/info/rfc7455>>.
- [RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <<https://www.rfc-editor.org/info/rfc8077>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8328] Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)", RFC 8328, DOI 10.17487/RFC8328, March 2018, <<https://www.rfc-editor.org/info/rfc8328>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <<https://www.rfc-editor.org/info/rfc8346>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.
- [RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <<https://www.rfc-editor.org/info/rfc8512>>.
- [RFC8513] Boucadair, M., Jacquenet, C., and S. Sivakumar, "A YANG Data Model for Dual-Stack Lite (DS-Lite)", RFC 8513, DOI 10.17487/RFC8513, January 2019, <<https://www.rfc-editor.org/info/rfc8513>>.
- [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <<https://www.rfc-editor.org/info/rfc8519>>.

- [RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <<https://www.rfc-editor.org/info/rfc8528>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.
- [RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <<https://www.rfc-editor.org/info/rfc8530>>.
- [RFC8531] Kumar, D., Wu, Q., and Z. Wang, "Generic YANG Data Model for Connection-Oriented Operations, Administration, and Maintenance (OAM) Protocols", RFC 8531, DOI 10.17487/RFC8531, April 2019, <<https://www.rfc-editor.org/info/rfc8531>>.
- [RFC8532] Kumar, D., Wang, Z., Wu, Q., Ed., Rahman, R., and S. Raghavan, "Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8532, DOI 10.17487/RFC8532, April 2019, <<https://www.rfc-editor.org/info/rfc8532>>.
- [RFC8533] Kumar, D., Wang, M., Wu, Q., Ed., Rahman, R., and S. Raghavan, "A YANG Data Model for Retrieval Methods for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8533, DOI 10.17487/RFC8533, April 2019, <<https://www.rfc-editor.org/info/rfc8533>>.

Appendix A. Layered YANG Modules Example Overview

It is not the intent of this document to provide an inventory of tools and mechanisms used in specific network and service management domains; such inventory can be found in documents such as [RFC7276].

A.1. Service Models: Definition and Samples

As described in [RFC8309], the service is "some form of connectivity between customer sites and the Internet and/or between customer sites across the network operator's network and across the Internet". More concretely, an IP connectivity service can be defined as the IP transfer capability characterized by a (Source Nets, Destination Nets, Guarantees, Scope) tuple where "Source Nets" is a group of

unicast IP addresses, "Destination Nets" is a group of IP unicast and/or multicast addresses, and "Guarantees" reflects the guarantees (expressed in terms of Quality Of Service (QoS), performance, and availability, for example) to properly forward traffic to the said "Destination" [RFC7297].

For example:

- o L3SM model [RFC8299] defines the L3VPN service ordered by a customer from a network operator.
- o L2SM model [RFC8466] defines the L2VPN service ordered by a customer from a network operator.
- o VN model [I-D.ietf-teas-actn-vn-yang] provides a YANG data model generally applicable to any mode of Virtual Network (VN) operation.

A.2. Network Models: Definitions and Samples

Figure 4 depicts a set of Network models such as topology models or tunnel models:

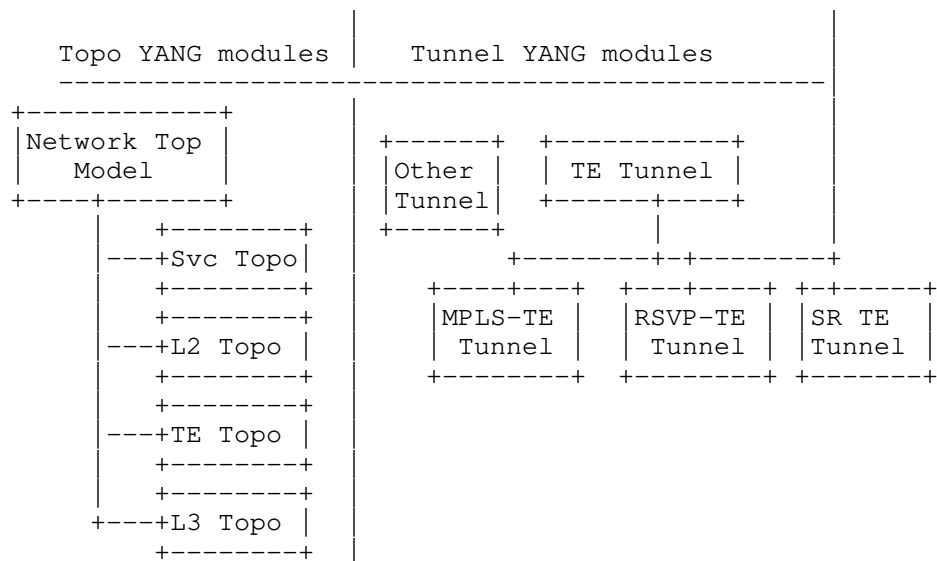


Figure 4: Sample Resource Facing Network Models

Topology YANG module Examples:

- o Network Topology Models: [RFC8345] defines a base model for network topology and inventories. Network topology data include link resource, node resource, and terminate-point resources.
- o TE Topology Models: [I.D-ietf-teas-yang-te-topo] defines a data model for representing and manipulating TE topologies.

This module is extended from network topology model defined in [RFC8345] with TE topologies specifics. This model contains technology-agnostic TE Topology building blocks that can be augmented and used by other technology-specific TE Topology models.

- o L3 Topology Models

[RFC8346] defines a data model for representing and manipulating L3 Topologies. This model is extended from the network topology model defined in [RFC8345] with L3 topologies specifics.

- o L2 Topology Models

[I.D-ietf-i2rs-yang-l2-topology] defines a data model for representing and manipulating L2 Topologies. This model is extended from the network topology model defined in [RFC8345] with L2 topologies specifics.

Tunnel YANG module Examples:

- o Tunnel identities [I-D.ietf-software-iftunnel] to ease manipulating extensions to specific tunnels.
- o TE Tunnel Model

[I.D-ietf-teas-yang-te] defines a YANG module for the configuration and management of TE interfaces, tunnels and LSPs.

- o SR TE Tunnel Model

[I.D-ietf-teas-yang-te] augments the TE generic and MPLS-TE model(s) and defines a YANG module for Segment Routing (SR) TE specific data.

- o MPLS TE Model

[I.D-ietf-teas-yang-te] augments the TE generic and MPLS-TE model(s) and defines a YANG module for MPLS TE configurations, state, RPC and notifications.

- o RSVP-TE MPLS Model

[I.D-ietf-teas-yang-rsvp-te] augments the RSVP-TE generic module with parameters to configure and manage signaling of MPLS RSVP-TE LSPs.

Other Network Models:

- o Path Computation API Model

[I.D-ietf-teas-path-computation] YANG module for a stateless RPC which complements the stateful solution defined in [I.D-ietf-teas-yang-te].

- o OAM Models (including Fault Management (FM) and Performance Monitoring)

[RFC8532] defines a base YANG module for the management of OAM protocols that use Connectionless Communications. [RFC8533] defines a retrieval method YANG module for connectionless OAM protocols. [RFC8531] defines a base YANG module for connection oriented OAM protocols. These three models are intended to provide consistent reporting, configuration and representation for connection-less OAM and Connection oriented OAM separately.

Alarm monitoring is a fundamental part of monitoring the network. Raw alarms from devices do not always tell the status of the network services or necessarily point to the root cause. [I.D-ietf-ccamp-alarm-module] defines a YANG module for alarm management.

- o Generic Policy Model

The Simplified Use of Policy Abstractions (SUPA) policy-based management framework [RFC8328] defines base YANG modules [I-D.ietf-sup-generic-policy-data-model] to encode policy. These models point to device-, technology-, and service-specific YANG modules developed elsewhere. Policy rules within an operator's environment can be used to express high-level, possibly network-wide, policies to a network management function (within a controller, an orchestrator, or a network element). The network management function can then control the configuration and/or monitoring of network elements and services. This document describes the SUPA basic framework, its elements, and interfaces.

A.3. Device Models: Definitions and Samples

Network Element models (Figure 5) are used to describe how a service can be implemented by activating and tweaking a set of functions (enabled in one or multiple devices, or hosted in cloud infrastructures) that are involved in the service delivery. The following figure uses IETF defined models as an example.

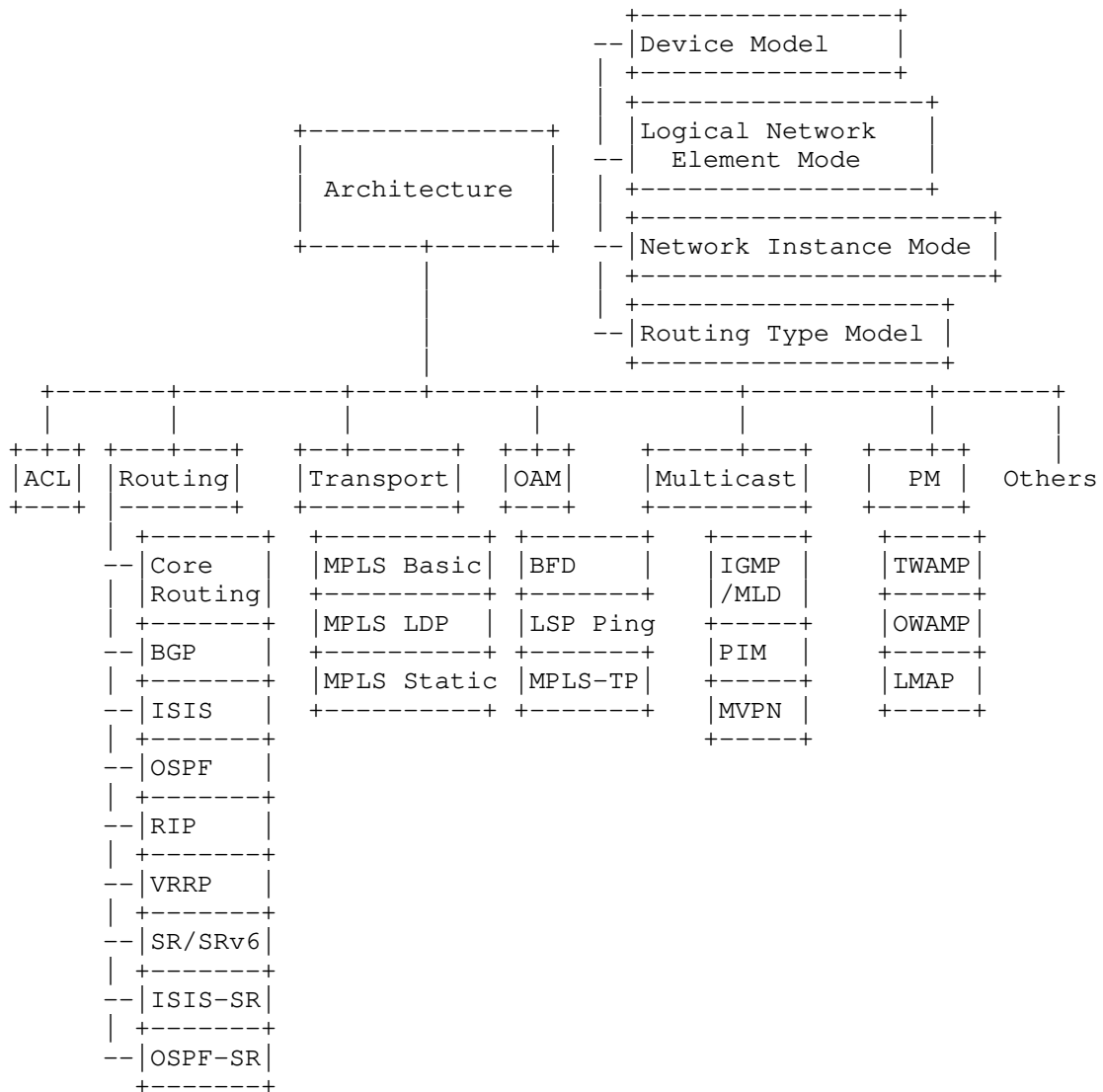


Figure 5: Network Element Modules Overview

A.3.1. Model Composition

- o Device Model

[I.D-ietf-rtgwg-device-model] presents an approach for organizing YANG modules in a comprehensive logical structure that may be used to configure and operate network devices. The structure is itself represented as an example YANG module, with all of the related component models logically organized in a way that is operationally intuitive, but this model is not expected to be implemented.

- o Logical Network Element Model

[RFC8530] defines a logical network element module which can be used to manage the logical resource partitioning that may be present on a network device. Examples of common industry terms for logical resource partitioning are Logical Systems or Logical Routers.

- o Network Instance Model

[RFC8529] defines a network instance module. This module can be used to manage the virtual resource partitioning that may be present on a network device. Examples of common industry terms for virtual resource partitioning are Virtual Routing and Forwarding (VRF) instances and Virtual Switch Instances (VSIs).

A.3.1.1. Schema Mount

Modularity and extensibility were among the leading design principles of the YANG data modeling language. As a result, the same YANG module can be combined with various sets of other modules and thus form a data model that is tailored to meet the requirements of a specific use case. [RFC8528] defines a mechanism, denoted schema mount, that allows for mounting one data model consisting of any number of YANG modules at a specified location of another (parent) schema.

That capability does not cover design time.

A.3.2. Device Models: Definitions and Samples

BGP: [I-D.ietf-idr-bgp-yang-model] defines a YANG module for configuring and managing BGP, including protocol, policy, and operational aspects based on data center, carrier and content provider operational requirements.

- MPLS: [I-D.ietf-mpls-base-yang] defines a base model for MPLS which serves as a base framework for configuring and managing an MPLS switching subsystem. It is expected that other MPLS technology YANG modules (e.g. MPLS LSP Static, LDP or RSVP-TE models) will augment the MPLS base YANG module.
- QoS: [I-D.asechoud-netmod-diffserv-model] describes a YANG module of Differentiated Services for configuration and operations.
- ACL: Access Control List (ACL) is one of the basic elements used to configure device forwarding behavior. It is used in many networking technologies such as Policy Based Routing, Firewalls, etc. [RFC8519] describes a data model of Access Control List (ACL) basic building blocks.
- NAT: For the sake of network automation and the need for programming Network Address Translation (NAT) function in particular, a data model for configuring and managing the NAT is essential. [RFC8512] defines a YANG module for the NAT function covering a variety of NAT flavors such as Network Address Translation from IPv4 to IPv4 (NAT44), Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers (NAT64), customer-side translator (CLAT), Stateless IP/ICMP Translation (SIIT), Explicit Address Mappings (EAM) for SIIT, IPv6-to-IPv6 Network Prefix Translation (NPTv6), and Destination NAT. [RFC8513] specifies a YANG module for the DS-Lite AFTR.
- Stateless Address Sharing: [I-D.ietf-softwire-yang] specifies a YANG module for A+P address sharing, including Lightweight 4over6, Mapping of Address and Port with Encapsulation (MAP-E), and Mapping of Address and Port using Translation (MAP-T) softwire mechanisms.
- Multicast: [I-D.ietf-pim-yang] defines a YANG module that can be used to configure and manage Protocol Independent Multicast (PIM) devices. [I-D.ietf-pim-igmp-mld-yang] defines a YANG module that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. [I-D.ietf-pim-igmp-mld-snooping-yang] defines a YANG module that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping devices.

EVPN: [I-D.ietf-bess-evpn-yang] defines a YANG module for Ethernet VPN services. The model is agnostic of the underlay. It apply to MPLS as well as to VxLAN encapsulation. The model is also agnostic of the services including E-LAN, E-LINE and E-TREE services. This document mainly focuses on EVPN and Ethernet-Segment instance framework.

L3VPN: [I-D.ietf-bess-l3vpn-yang] defines a YANG module that can be used to configure and manage BGP L3VPNs [RFC4364]. It contains VRF specific parameters as well as BGP specific parameters applicable for L3VPNs.

L2VPN: [I-D.ietf-bess-l2vpn-yang] defines a YANG module for MPLS based Layer 2 VPN services (L2VPN) [RFC4664] and includes switching between the local attachment circuits. The L2VPN model covers point-to-point VPWS and Multipoint VPLS services. These services use signaling of Pseudowires across MPLS networks using LDP [RFC8077][RFC4762] or BGP [RFC4761].

Routing Policy: [I-D.ietf-rtgwg-policy-model] defines a YANG module for configuring and managing routing policies in a vendor-neutral way and based on actual operational practice. The model provides a generic policy framework which can be augmented with protocol-specific policy configuration.

BFD: [I-D.ietf-bfd-yang] defines a YANG module that can be used to configure and manage Bidirectional Forwarding Detection (BFD) [RFC5880]. BFD is a network protocol which is used for liveness detection of arbitrary paths between systems.

SR/SRv6: [I-D.ietf-spring-sr-yang] a YANG module for segment routing configuration and operation. [I-D.raza-spring-srv6-yang] defines a YANG module for Segment Routing IPv6 (SRv6) base. The model serves as a base framework for configuring and managing an SRv6 subsystem and expected to be augmented by other SRv6 technology models accordingly.

Core Routing: [RFC8349] defines the core routing data model, which is intended as a basis for future data model development covering more-sophisticated routing systems. It is expected that other Routing technology YANG modules (e.g., VRRP, RIP, ISIS, OSPF models) will augment the Core Routing base YANG module.

PM:

[I.D-ietf-ippm-twamp-yang] defines a data model for client and server implementations of the Two-Way Active Measurement Protocol (TWAMP).

[I.D-ietf-ippm-stamp-yang] defines the data model for implementations of Session-Sender and Session-Reflector for Simple Two-way Active Measurement Protocol (STAMP) mode using YANG.

[RFC8194] defines a data model for Large-Scale Measurement Platforms (LMAPs).

Authors' Addresses

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

Email: bill.wu@huawei.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Diego R. Lopez
Telefonica I+D
Spain

Email: diego.r.lopez@telefonica.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf.bri@chinatelecom.cn

Liang Geng
China Mobile

Email: gengliang@chinamobile.com