OPSAWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2019

R. Even
B. Wu
Q. Wu
Huawei
Y. Cheng
China Unicom
March 8, 2019

YANG Data Model for Composed VPN Service Delivery
draft-evenwu-opsawg-yang-composed-vpn-03

Abstract

This document defines a YANG data model that can be used by a network operator to configure a VPN service that spans multiple administrative domains and that is constructed from component VPNs in each of those administrative domains. The component VPNs may be L2VPN or L3VPN or a mixture of the two. This model is intended to be instantiated at the management system to deliver the end to end service (i.e., performing service provision and activation functions at different levels through a unified interface).

The model is not a configuration model to be used directly on network elements. This model provides an abstracted common view of VPN service configuration components segmented at different layer and administrative domain. It is up to a management system to take this as an input and generate specific configurations models to configure the different network elements within each administrative domain to deliver the service. How configuration of network elements is done is out of scope of the document.

Table of Contents

1. Introduction

In some cases, a VPN service needs to span different administrative domains. This will usually arise when there are internal administrative boundaries within a single Service Provider's (SP's)

network. The boundaries may reflect geographic dispersal or functional decomposition, e.g., access, metro, backhaul, core, and data center.

In particular, the different domains could deploy Layer 2 or Layer 3 technologies or both, and could establish layer-dependent connectivity services. For example, some SPs offer a L2VPN service in the metro access network and extend it across the core network as an IP VPN to provide end-to-end BGP IP VPN services to their enterprise customers.

Some SPs integrate Mobile Backhaul Network and Core networks to provide mobile broadband services. These require stitching multiple layer-dependent connectivity services at different administrative domain boundaries.

This document defines a YANG data model that can be used by a network operator to construct an end-to-end service across multiple administrative domains. This service is delivered by provisioning VPN services utilising Layer 2 or Layer 3 technologies in each domain.

This model is intended to be instantiated at the management system to deliver the overall service per [RFC8309]. It is not a configuration model to be used directly on network elements. This model provides an abstracted common view of VPN service configuration components segmented at different layers and administrative domains. It is up to a management system to take this as an input and generate specific configurations models to configure the different network elements within each administrative domain to deliver the service. How configuration of network elements is done is out of scope of the document. END

1.1. Terminology

The following terms are defined in [RFC6241] and are not redefined here:

- client
- server
- configuration data
- state data

The following terms are defined in [RFC7950] and are not redefined here:

- augment
- data model
- data node

The terminology for describing YANG data models is found in [RFC7950].

1.1.1. Requirements Language

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

1.2. Tree diagram

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Definitions

This document uses the following terms:

Service Provider (SP): The organization (usually a commercial undertaking) responsible for operating the network that offers VPN services to clients and customers.

Customer Edge (CE) Device: Equipment that is dedicated to a particular customer and is directly connected to one or more PE devices via attachment circuits. A CE is usually located at the customer premises, and is usually dedicated to a single VPN, although it may support multiple VPNs if each one has separate attachment circuits. The CE devices can be routers, bridges, switches, or hosts.

Provider Edge (PE) Device: Equipment managed by the SP that can support multiple VPNs for different customers, and is directly connected to one or more CE devices via attachment circuits. A PE is usually located at an SP point of presence (PoP) and is managed by the SP.

Administrative Domain: A collection of End Systems, Intermediate Systems, and subnetworks operated by a single organization or administrative authority. The components which make up the domain are assumed to interoperate with a significant degree of mutual

trust among themselves, but interoperate with other Administrative Domains in a mutually suspicious manner [RFC1136].

A group of hosts, routers, and networks operated and managed by a single organization. Routing within an Administrative Domain is based on a consistent technical plan. An Administrative Domain is viewed from the outside, for purposes of routing, as a cohesive entity, of which the internal structure is unimportant. Information passed by other Administrative Domains is trusted less than information from one's own Administrative Domain.

Administrative Domains can be organized into a loose hierarchy that reflects the availability and authoritativeness of routing information. This hierarchy does not imply administrative containment, nor does it imply a strict tree topology.

Routing Domain: A set of End Systems and Intermediate Systems which operate according to the same routing procedures and which is wholly contained within a single Administrative Domain [RFC1136].

A Routing Domain is a set of Intermediate Systems and End Systems bound by a common routing procedure; namely: they are using the same set of routing metrics, they use compatible metric measurement techniques, they use the same information distribution protocol, and they use the same path computation algorithm" An Administrative Domain may contain multiple Routing Domains. A Routing Domain may never span multiple Administrative Domains.

An Administrative Domain may consist of only a single Routing Domain, in which case they are said to be Congruent. A congruent Administrative Domain and Routing Domain is analogous to an Internet Autonomous System.

Access point(AP): Describe an VPN's end point characteristics and its reference to a Termination Point (TP) of the Provider Edge (PE) Node; used as service access point for connectivity service segment in the end-to-end manner and per administrative domain.

Site: Represent a connection of a customer office to one or more VPN services and contain a list of network accesses associated with the site. Each network access can connect to different VPN service.

Segment VPN Describe generic information about a VPN in a single administrative domain, and specific information about APs that connect the Segment VPN to sites or to other Segment VPNs.

Composed VPN Describe generic end-to-end information about a VPN that spans multiple administrative domains, and specific customer-facing information about APsconnecting to each site.

3. Service Model Usage

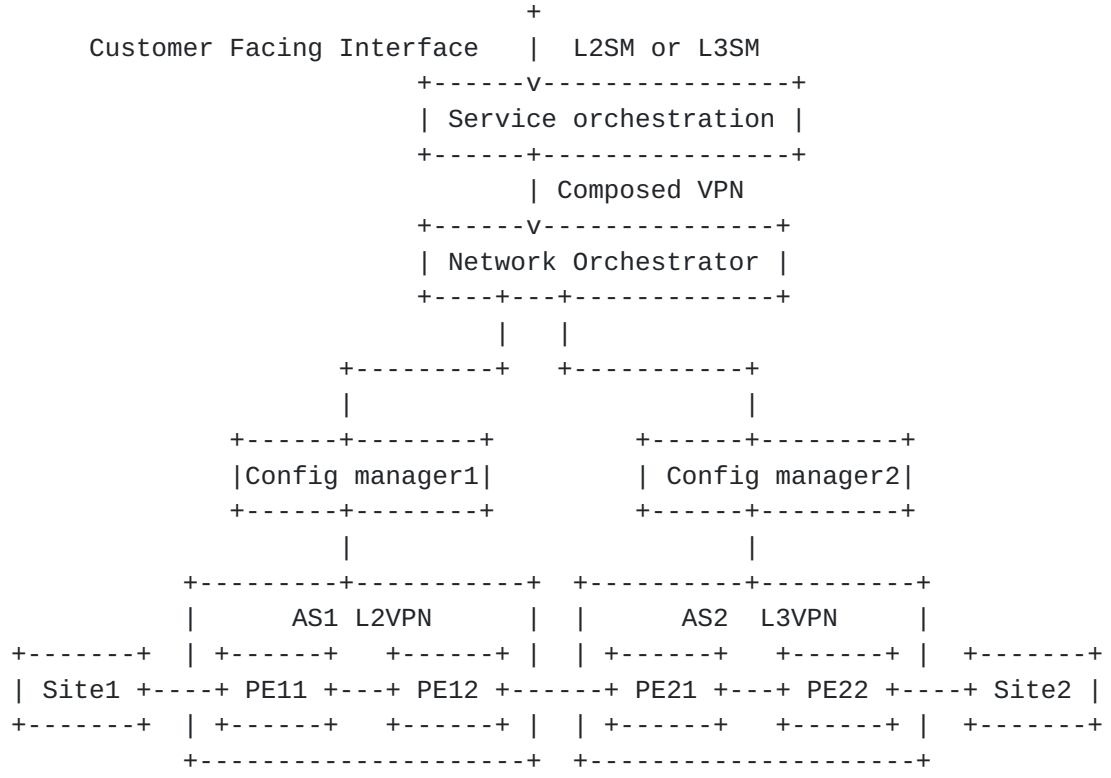


Figure 1: Service Model Usage

In the above use case, the network orchestrator controls and manages the two distinct network domains, each controlled or managed by their own management system or domain controller. There are two typical ways to deploy the composed VPN model:

One typical scenario would be to use the model as an independent model. The orchestration layer could use composed VPN model as an input, and translate it to segmented VPN model for each administrative domain. And the domain management system could further configure network elements based on configuration obtained from the segment VPN.

The other scenario is to use customer facing model such as L3SM service model as an input for the service orchestration layer that will be responsible for translating the parameters of VPN and site in L3SM model to the corresponding parameters of the composed VPN model, then with extra provisioning parameters added ,the composed VPN model

can be further broken down into per domain segmented VPN model and additional Access point configuration.

The usage of this composed VPN model is not limited to this example; it can be used by any component of the management system but not directly by network elements.

4. The Composed VPN Service Model

A composed VPN represents an end-to-end IP or Ethernet connectivity between the access points of PE where the AP can interconnect with the enterprise customer's network or other types of overlay network. The Composed VPN model provides a common understanding of how the corresponding composed VPN service is to be deployed in an end to end manner over the multi-domain infrastructure.

This document presents the Composed VPN Service Delivery Model using the YANG data modeling language [RFC7950] as a formal language that is both human-readable and parsable by software for use with protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040].

4.1. VPN Service Types

From a technology perspective, a Composed VPN can be classified into three categories based on the domain specific VPN types including L2VPN and L3VPN, see Figure 2. And in each category, the interworking option may vary depending on the inter-domain technology, such as IP or MPLS forwarding. In some cases, the number of transit domain can be zero or multiple.

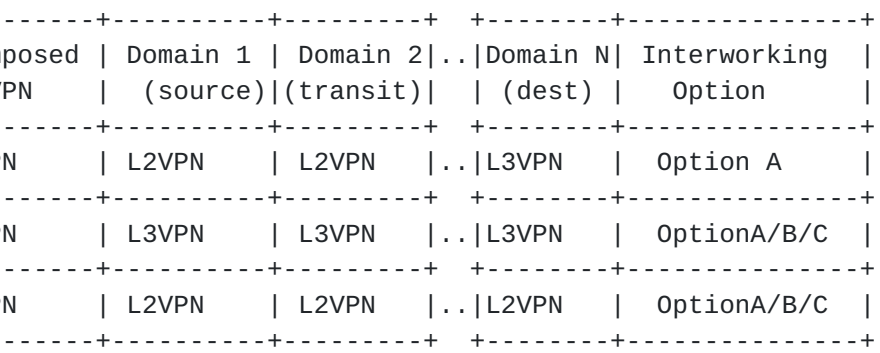


Figure 2: Composed VPN classification

4.2. Composed VPN Physical Network Topology

Figure 3 describes a scenario where connectivity in the form of an L3VPN is provided across a Mobile Backhaul Network. The network has two ASes: connectivity across AS A is achieved with an L2VPN, and

across AS B an L3VPN. The ASes are interconnected, and the composed VPN is achieved by interconnecting the L2VPN with the L3VPN.

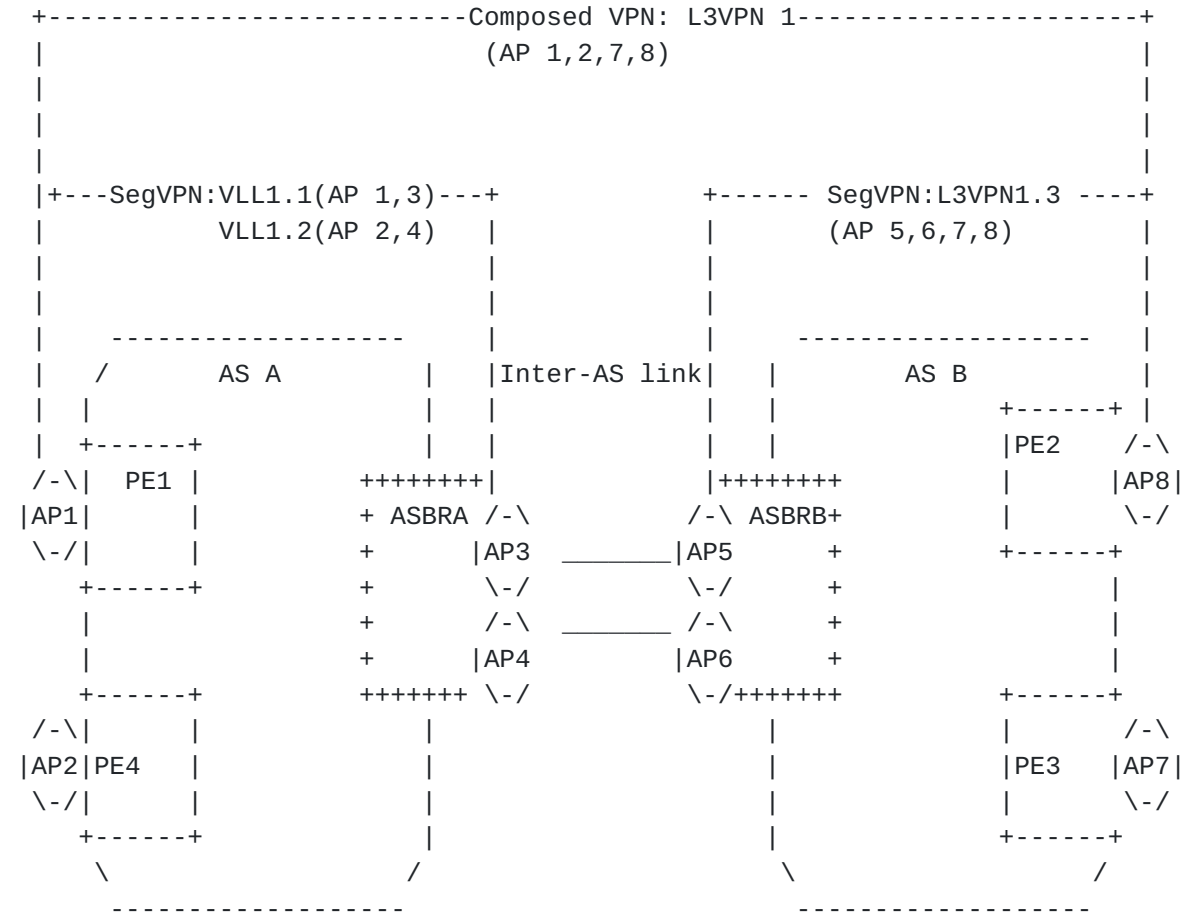


Figure 3: Mobile Backhaul Network Scenario

The Composed VPN is a service that provides connectivity between AP1, AP2, AP7 and AP8. As the APs of the VPN are spanning the two domains, the ASBR A and B and their associated links are required to be identified. Based on the decomposition, two Segment VPN could be constructed to provide per domain connections. Segment VPN 1.1 and Segment VPN 1.2 are connections between AP 1,2,3,4 in the domain A of access metro network, which are L2VPN. Segment VPN 1.3 is the connection between AP 5,6,7,8 in the core network, which is L3VPN. The ASBR A and B at the edge of the access metro network is performing the VPN stitching between Layer 2 VPN and Layer 3 VPN using the technology such as bridging or other interconnection technology.

The operator can predefine several VPN provisioning policies based on the offered business. The policy description may include the naming, path selection, VPN concatenation rules,and resource pools, such as

route target, route distinguisher. How VPN provision policies configuration of network elements is done is out of scope of the document.

5. Design of the Data Model

The idea of the composed VPN model is to decompose an end-to-end L2VPN or L3VPN service across multiple administrative domains into point-to-point VPN segments or multi-point VPN segments in each administrative domain, and to stich these segments together by using different interworking options. Therefore, a complete composed VPN instance consists of:

- One composed VPN with corresponding composed VPN set of parameter
- Two or more APs, each with a corresponding set of AP parameters
- One or more segment VPN with corresponding segment VPN set of parameter

Similar to the L3SM [RFC8299] and L2SM [RFC8466] modelling structure, the composed VPN model consists of two main components, the VPN component and the AP component.

The figure below describes the overall structure of the YANG module:

```
module: ietf-composed-vpn-svc
  +--rw composed-vpns
     +--rw composed-vpn* [vpn-id]
        +--rw vpn-id           yang:uuid
        +--rw vpn-name?        string
        +--rw customer-name?   yang:uuid
        +--rw topo?            svpn:vpn-topology
        +--rw service-type?    svpn:service-type
        +--rw tunnel-type?     svpn:tunnel-type
        +--rw admin-state?     svpn:admin-state
        +--ro oper-State?      svpn:oper-state
        +--ro sync-state?      svpn:sync-state
        +--rw start-time?      yang:date-and-time
        +--rw segment-vpn* [vpn-id]
        |  +--rw vpn-id          yang:uuid
        |  +--rw vpn-name?       string
        |  +--rw service-type?   service-type
        |  +--rw topo?           vpn-topology
        |  +--rw tunnel-type?    tunnel-type
        |  +--rw admin-state?    admin-state
        |  +--ro oper-state?     oper-state
        |  +--ro sync-state?     sync-state
```

```
|  +--rw access-point* [tp-id]
|     +--rw tp-id                                yang:uuid
|     +--rw tp-common-attribute
|     |  +--rw tp-id?              yang:uuid
|     |  +--rw tp-name?            string
|     |  +--rw node-id?            yang:uuid
|     |  +--rw access-point-type?  access-point-type
|     |  +--rw inter-as-option?    enumeration
|     |  +--rw topology-role?      topology-role
|     +--rw peer-remote-node
|     |  +--rw remote-id?          yang:uuid
|     |  +--rw location?           string
|     |  +--rw remote-tp-address?  inet:ip-address
|     |  +--rw remote-node-id?     yang:uuid
|     |  +--rw remote-tp-id?       yang:uuid
|     +--rw tp-connection-specific-attribute
|     |  +--rw connection* [connection-class]
|     |  |  +--rw connection-class    layer-rate
|     |  |  +--rw (connection-type)?
|     |  |     +--:(lr-eth)
|     |  |     |  +--rw eth
|     |  |     |     +--rw access-type?   eth-encap-type
|     |  |     |     +--rw (accessVlanValue)?
|     |  |     |     |  +--:(qinq)
|     |  |     |     |  |  +--rw qinq
|     |  |     |     |  |     +--rw cvlan*   uint64
|     |  |     |     |  |     +--rw svlan?   uint64
|     |  |     |     |  +--:(dot1q)
|     |  |     |     |     +--rw dot1q
|     |  |     |     |        +--rw dot1q*   uint64
|     |  |     |     +--rw vlan-action?   ethernet-action
|     |  |     |     +--rw action?        string
|     |  |     +--:(lr-ip)
|     |  |     |  +--rw ip
|     |  |     |     +--rw ip-address?   inet:ip-address
|     |  |     |     +--rw mtu?          uint64
|     |  |     +--:(lr-pw)
|     |  |        +--rw pw
|     |  |           +--rw control-word?   boolean
|     |  |           +--rw vlan-action?    pwtagmode
|     |  +--rw security-attribute
|     |  |  +--rw security
|     |  |     +--rw authentication
|     |  |     +--rw encryption {encryption}?
|     |  |        +--rw enabled?              boolean
|     |  |        +--rw layer?                enumeration
|     |  |        +--rw algorithm?            string
|     |  |        +--rw (key-type)?
```

```
        |     |  |          +--:(psk)
        |     |  |             +--rw preshared-key?   string
        |     |  +--rw qos-attribute
        |     |  |  +--rw svc-input-bandwidth     uint64
        |     |  |  +--rw svc-output-bandwidth    uint64
        |     |  |  +--rw svc-mtu                 uint16
        |     |  |  +--rw qos {qos}?
        |     |  |     +--rw qos-classification-policy
        |     |  |     |  +--rw rule* [id]
        |     |  |     |     +--rw id                          string
        |     |  |     |     +--rw (match-type)?
        |     |  |     |     |  +--:(match-flow)
        |     |  |     |     |  |  +--rw match-flow
        |     |  |     |     |  |     +--rw dscp?               inet:dscp
        |     |  |     |     |  |     +--rw exp?                inet:dscp
        |     |  |     |     |  |     +--rw dot1p?              uint8
        |     |  |     |     |  |     +--rw ipv4-src-prefix?    inet:ipv4-
prefix
        |     |  |     |     |  |     +--rw ipv6-src-prefix?    inet:ipv6-
prefix
        |     |  |     |     |  |     +--rw ipv4-dst-prefix?    inet:ipv4-
prefix
        |     |  |     |     |  |     +--rw ipv6-dst-prefix?    inet:ipv6-
prefix
        |     |  |     |     |  |     +--rw l4-src-port?        inet:port-
number
        |     |  |     |     |  |     +--rw peer-remote-node*   string
        |     |  |     |     |  |     +--rw l4-src-port-range
        |     |  |     |     |  |     |  +--rw lower-port?   inet:port-number
        |     |  |     |     |  |     |  +--rw upper-port?   inet:port-number
        |     |  |     |     |  |     +--rw l4-dst-port?        inet:port-
number
        |     |  |     |     |  |     +--rw l4-dst-port-range
        |     |  |     |     |  |     |  +--rw lower-port?   inet:port-number
        |     |  |     |     |  |     |  +--rw upper-port?   inet:port-number
        |     |  |     |     |  |     +--rw src-mac?            yang:mac-
address
        |     |  |     |     |  |     +--rw dst-mac?            yang:mac-
address
        |     |  |     |     |  |     +--rw protocol-field?     union
        |     |  |     |     |  +--:(match-application)
        |     |  |     |     |     +--rw match-application?   identityref
        |     |  |     |     +--rw target-class-id?          string
        |     |  |     +--rw qos-profile
        |     |  |        +--rw (qos-profile)?
        |     |  |           +--:(standard)
        |     |  |           |  +--rw profile?   string
        |     |  |           +--:(custom)
```

```
|     |  |              +--rw classes {qos-custom}?
|     |  |                 +--rw class* [class-id]
|     |  |                    +--rw class-id      string
|     |  |                    +--rw direction?    identityref
|     |  |                    +--rw rate-limit?   decimal64
|     |  |                    +--rw latency
|     |  |                    |  +--rw (flavor)?
|     |  |                    |     +--:(lowest)
```

```
        |     |  |                    |     |  +--rw use-lowest-latency?
empty
        |     |  |                    |     +--:(boundary)
        |     |  |                    |        +--rw latency-boundary?
uint16
        |     |  |                    +--rw jitter
        |     |  |                    |  +--rw (flavor)?
        |     |  |                    |     +--:(lowest)
        |     |  |                    |     |  +--rw use-lowest-jitter?   empty
        |     |  |                    |     +--:(boundary)
        |     |  |                    |        +--rw latency-boundary?
uint32
        |     |  |                    +--rw bandwidth
        |     |  |                       +--rw guaranteed-bw-percent
decimal64
        |     |  |                       +--rw end-to-end?              empty
        |     |  +--rw protection-attribute
        |     |     +--rw access-priority?   uint32
        |     +--rw routing-protocol* [type]
        |        +--rw type            protocol-type
        |        +--rw (para)?
        |           +--:(static)
        |           |  +--rw static* [index]
        |           |     +--rw index               uint32
        |           |     +--rw dest-cidr?          string
        |           |     +--rw egress-tp?          yang:uuid
        |           |     +--rw route-preference?   string
        |           |     +--rw next-hop?           inet:ip-address
        |           +--:(bgp)
        |              +--rw bgp* [index]
        |                 +--rw index                uint32
        |                 +--rw autonomous-system    uint32
        |                 +--rw address-family*      address-family
        |                 +--rw max-prefix?          int32
        |                 +--rw peer-address?        inet:ip-address
        |                 +--rw crypto-algorithm     identityref
        |                 +--rw key-string
        |                    +--rw (key-string-style)?
        |                       +--:(keystring)
        |                       |  +--rw keystring?            string
        |                       +--:(hexadecimal) {hex-key-string}?
        |                          +--rw hexadecimal-string?   yang:hex-string
        +--rw access-point* [tp-id]
           +--rw tp-id                              yang:uuid
           +--rw tp-name?             string
           +--rw node-id?             yang:uuid
           +--rw access-point-type?   access-point-type
           +--rw inter-as-option?     enumeration
```

```
          +--rw topology-role?       topology-role
          +--rw peer-remote-node
          |  +--rw remote-id?           yang:uuid
          |  +--rw location?            string
```

```
|  +--rw remote-tp-address?   inet:ip-address
|  +--rw remote-node-id?      yang:uuid
|  +--rw remote-tp-id?        yang:uuid
+--rw tp-connection-specific-attribute
|  +--rw connection* [connection-class]
|  |  +--rw connection-class    layer-rate
|  |  +--rw (connection-type)?
|  |     +--:(lr-eth)
|  |     |  +--rw eth
|  |     |     +--rw access-type?   eth-encap-type
|  |     |     +--rw (accessVlanValue)?
|  |     |     |  +--:(qinq)
|  |     |     |  |  +--rw qinq
|  |     |     |  |     +--rw cvlan*   uint64
|  |     |     |  |     +--rw svlan?   uint64
|  |     |     |  +--:(dot1q)
|  |     |     |     +--rw dot1q
|  |     |     |        +--rw dot1q*   uint64
|  |     |     +--rw vlan-action?   ethernet-action
|  |     |     +--rw action?        string
|  |     +--:(lr-ip)
|  |     |  +--rw ip
|  |     |     +--rw ip-address?   inet:ip-address
|  |     |     +--rw mtu?          uint64
|  |     +--:(lr-pw)
|  |        +--rw pw
|  |           +--rw control-word?   boolean
|  |           +--rw vlan-action?    pwtagmode
|  +--rw security-attribute
|  |  +--rw security
|  |     +--rw authentication
|  |     +--rw encryption {encryption}?
|  |        +--rw enabled?               boolean
|  |        +--rw layer?                 enumeration
|  |        +--rw algorithm?             string
|  |        +--rw (key-type)?
|  |           +--:(psk)
|  |              +--rw preshared-key?   string
|  +--rw qos-attribute
|  |  +--rw svc-input-bandwidth     uint64
|  |  +--rw svc-output-bandwidth    uint64
|  |  +--rw svc-mtu                 uint16
|  |  +--rw qos {qos}?
|  |     +--rw qos-classification-policy
|  |     |  +--rw rule* [id]
|  |     |     +--rw id                          string
|  |     |     +--rw (match-type)?
|  |     |     |  +--:(match-flow)
```

```
|  |     |     |  |  +--rw match-flow
|  |     |     |  |     +--rw dscp?                inet:dscp
|  |     |     |  |     +--rw exp?                 inet:dscp
|  |     |     |  |     +--rw dot1p?               uint8
|  |     |     |  |     +--rw ipv4-src-prefix?     inet:ipv4-prefix
|  |     |     |  |     +--rw ipv6-src-prefix?     inet:ipv6-prefix
|  |     |     |  |     +--rw ipv4-dst-prefix?     inet:ipv4-prefix
|  |     |     |  |     +--rw ipv6-dst-prefix?     inet:ipv6-prefix
|  |     |     |  |     +--rw l4-src-port?         inet:port-number
|  |     |     |  |     +--rw peer-remote-node*    string
|  |     |     |  |     +--rw l4-src-port-range
|  |     |     |  |     |  +--rw lower-port?   inet:port-number
|  |     |     |  |     |  +--rw upper-port?   inet:port-number
|  |     |     |  |     +--rw l4-dst-port?         inet:port-number
|  |     |     |  |     +--rw l4-dst-port-range
|  |     |     |  |     |  +--rw lower-port?   inet:port-number
|  |     |     |  |     |  +--rw upper-port?   inet:port-number
|  |     |     |  |     +--rw src-mac?             yang:mac-address
|  |     |     |  |     +--rw dst-mac?             yang:mac-address
|  |     |     |  |     +--rw protocol-field?      union
|  |     |     |  |  +--:(match-application)
|  |     |     |  |     +--rw match-application?   identityref
|  |     |     |  +--rw target-class-id?           string
|  |     |     +--rw qos-profile
|  |     |        +--rw (qos-profile)?
|  |     |           +--:(standard)
|  |     |           |  +--rw profile?   string
|  |     |           +--:(custom)
|  |     |              +--rw classes {qos-custom}?
|  |     |                 +--rw class* [class-id]
|  |     |                    +--rw class-id      string
|  |     |                    +--rw direction?    identityref
|  |     |                    +--rw rate-limit?   decimal64
|  |     |                    +--rw latency
|  |     |                    |  +--rw (flavor)?
|  |     |                    |     +--:(lowest)
|  |     |                    |     |  +--rw use-lowest-latency?   empty
|  |     |                    |     +--:(boundary)
|  |     |                    |        +--rw latency-boundary?     uint16
|  |     |                    +--rw jitter
|  |     |                    |  +--rw (flavor)?
|  |     |                    |     +--:(lowest)
|  |     |                    |     |  +--rw use-lowest-jitter?   empty
|  |     |                    |     +--:(boundary)
|  |     |                    |        +--rw latency-boundary?    uint32
|  |     |                    +--rw bandwidth
|  |     |                       +--rw guaranteed-bw-percent    decimal64
|  |     |                       +--rw end-to-end?              empty
```

```
|  +--rw protection-attribute
|     +--rw access-priority?   uint32
+--rw routing-protocol* [type]
   +--rw type            protocol-type
   +--rw (para)?
      +--:(static)
      |  +--rw static* [index]
      |     +--rw index               uint32
      |     +--rw dest-cidr?          string
      |     +--rw egress-tp?          yang:uuid
      |     +--rw route-preference?   string
      |     +--rw next-hop?           inet:ip-address
      +--:(bgp)
         +--rw bgp* [index]
            +--rw index                uint32
            +--rw autonomous-system    uint32
            +--rw address-family*      address-family
            +--rw max-prefix?          int32
            +--rw peer-address?        inet:ip-address
            +--rw crypto-algorithm     identityref
            +--rw key-string
               +--rw (key-string-style)?
                  +--:(keystring)
                  |  +--rw keystring?            string
                  +--:(hexadecimal) {hex-key-string}?
                     +--rw hexadecimal-string?   yang:hex-string
```

5.1. VPN Hierarchy

The composed VPN and segment VPN contain the following common parameters:

- vpn-id: Refers to an internal reference for this VPN service
- vpn-service-type: Combination of L3VPN service type and L2VPN service type per [RFC8466] and [RFC8299], including VPWS,VPLS,EVPN and L3VPN.
- vpn-topology: Combination of L3VPN topology and L2VPN topology, including hub-spoke, any-to-any and point-to-point.
- Tunnel-type:MPLS,MPLS-TP,SR,SRv6

Suppose a composed VPN is a L3VPN which could initially has sites connected to a single SP domain and may later add more sites to other domains in the SP network. Thus, a composed VPN could has one segment VPN at the beginning, and later has more segment VPNs.

5.2. Access Point(AP)

As the site containers of the L3SM and L2SM represent the connection characteristics that the CE connects to the provider network from the perspective of the customer, AP represents the connection characteristics that the PE connects to VPN from the perspective of the provider. Therefore, there are two main aspects relates to the AP modelling:

- The AP component under composed VPN container describes the intent parameters mapping from the L3SM and L2SM, and the AP component under the segment VPN container describes the configuration parameters of the specific domain derived from the decomposition of composed VPN model.

- In a specific segment VPN, the AP component not only describes the CE-PE connection, but also defines inter-domain connection parameters between ASBR peer. The connection between PE and ASBR is related to configuration of network elements and not part of segment VPN model.

5.2.1. AP peering with CE

The AP parameters contains the following group of parameters:

Basic AP parameters: topology role could be hub role, leaf role

Connection: has a knob to accommodate either Layer2 or Layer 3 data plane connection

Control plane peering: has a knob to accommodate either Layer 2 protocol or Layer 3 routing protocol

QoS profile and QoS-classification-policy: has a knob to accommodate either Layer 2 QoS profile and qos-classification-policy or Layer 3 QoS profile and Qos-classification-policy, to describe both per AP bandwidth and per flow QoS.

Security Policy: has a knob to accommodate either Layer 2 QoS profile and qos-classification-policy or Layer 3 QoS profile and qos-classification-policy, to describe both per AP bandwidth and per flow QoS.

Although both the composed VPN and segment VPN use the AP to describe the connection parameters of the CE and the PE, the AP parameter of the composed VPN may not be directly mapped to the AP parameters of the segment VPN. For example, a composed VPN is a L3VPN with one of its AP which specifies the IP connection parameters and per flow QOS

requirement. During decomposition, depending on the capability of the accessed domain which the segment VPN resides, the AP of the segment VPN could only support Ethernet connection and per port bandwidth guarantee. Therefore, the AP could only configure with L2 connection and per AP bandwidth setting.

5.2.2. AP peering for inter-domains connection

The AP which describes the inter-domains connection could only exist in segment VPN. There are three options in connecting segment VPN across inter-domain link. With L3VPN, L2VPN or mixture, the option could be:

Interworking Option	AP type	AP CP remote peer	AP DP
Option A	ASBR LTP	ASBR	Interface
Option B	ASBR	ASBR	LSP label
Option C	PE,ASBR	remote PE	LSP label

The AP parameters contains the following group of parameters:

Basic AP parameters: Inter-AS interworking option could be Option A, Option B or Option C.

Connection: only specifies in Options A, Option B and C use dynamically allocated MPLS labels.

Control plane peering: BGP peering or static routing.

QoS profile and QoS-classification-policy: only applicable in Options A, Option B and C can only use MPLS EXP to differentiate the traffic.

Security policy: Options A use the similar mechanism like CE-PE peering, Option B could use BGP authentication to secure control plane communication and enable mpls label security, and Option C depends on the trust between the inter-domains.

5.2.2.1. Secure inter-domain connection

This model is applied to a single SP. Although there are different domain separation, implicit trust exists between the ASs because they

have the same operational control, for example from orchestrator's perspective.

The model specifies different security parameters depending on the various Inter-AS options:

- Option A uses interfaces or subinterfaces between autonomous system border routers (ASBRs) to keep the VPNs separate, so there is strict separation between VPNs.

- Option B can be secured with configuration on the control plane and the data plane. On the control plane, the session can be secured by use of peer authentication of BGP with message digest 5 (MD5) and TCP Authentication Option(TCP-AO), maximum route limits per peer and per VPN, dampening, and so on. In addition, prefix filters can be deployed to control which routes can be received from the other AS. On the data plane, labeled packets are exchanged. The label is derived from the MP-eBGP session; therefore, the ASBR announcing a VPN-IPv4 prefix controls and assigns the label for each prefix it announces. On the data plane, the incoming label is then checked to verify that this label on the data plane has really been assigned on the control plane. Therefore, it is impossible to introduce fake labels from one AS to another. The Authentication parameter could be set under the BGP peering configuration. An MPLS label security could be enabled under the connection node.

- Option C can also be secured well on the control plane, but the data plane does not provide any mechanism to check and block the packets to be sent into the other AS. On the control plane, model C has two interfaces between autonomous systems: The ASBRs exchange IPv4 routes with labels via eBGP. The purpose is to propagate the PE loopback addresses to the other AS so that LSPs can be established end to end. The other interface is the RRs exchange VPN-IPv4 routes with labels via multihop MP-eBGP. The prefixes exchanged can be controlled through route maps, equally the route targets. On the data plane, the traffic exchanged between the ASBRs contains two labels. One is VPN label set by the ingress PE to identify the VPN. The other is PE label Specifies the LSP to the egress PE. The Authentication and routing policy parameter could be set under the BGP peering configuration.

The security options supported in the model are limited but may be extended via augmentation.

5.2.2.2. Inter-domain QoS decomposition

The APs connected between the domains are aggregation points, and traffic from different CEs of the combined VPN cross-domain will interact through these aggregation points. To provide consistent QoS configuration, when several domains are involved in the provisioning of a VPN, topology, domain functionality and other factors need to be considered.

Option A can achieve most granular QoS implementation since IP traffic passes the inter-domain connection. Thus, Option A can set configuration with per sub-interface and IP DSCP. Option B and Option C only provide MPLS EXP differentiation. QoS mechanisms that are applied only to IP traffic cannot be carried.

In some cases, there is need to re-mark packets at Layer 3 to indicate whether traffic is in agreement. Because MPLS labels include 3 bits that commonly are used for QoS marking, it is possible for "tunnel DiffServ" to preserve Layer 3 DiffServ markings through a service provider's MPLS VPN cloud while still performing re-marking (via MPLS EXP bits) within the cloud to indicate in- or out-of-agreement traffic.

6. Composed VPN YANG Module

```
<CODE BEGINS> file "ietf-composed-vpn-svc.yang"
module ietf-composed-vpn-svc {
    namespace "urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc" ;
    prefix composed-vpn ;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-segment-vpn {
        prefix segment-vpn;
    }
    organization "IETF OPSAWG Working Group";
    contact "
     WG Web:   <https://datatracker.ietf.org/wg/opsawg>
        WG List:  <mailto:netmod@ietf.org>

        Editor:   Roni Even
                  <mailto:roni.even@huawei.com>
                  Bo Wu
                  <mailto:lana.wubo@huawei.com>
                  Qin Wu
                  <mailto:bill.wu@huawei.com>
                  Ying Cheng
                  <mailto:chengying10@chinaunicom.cn>";
```

```
    description "ietf-compsed-vpn";
    revision 2018-08-21 {
        reference "draft-evenwu-opsawg-yang-composed-vpn-00";
    }

    grouping vpn-basic {
        description "VPNBasicInfo Grouping.";
        leaf topo {
            type segment-vpn:vpn-topology;
            description "current support for full-mesh and
            point_to_multipoint(hub-spoke), others is reserved for
            future extensions." ;
        }
        leaf service-type {
            type segment-vpn:service-type;
            description "current support for mpls l3vpn/vxlan/L2VPN/hybrid
            VPN overlay, others is reserved for future extensions." ;
        }
        leaf tunnel-type {
            type segment-vpn:tunnel-type;
            description "mpls|vxlan overlay l3vpn|eth over sdh|nop";
        }
        leaf admin-state {
            type segment-vpn:admin-state;
            description "administrative status." ;
         }
       leaf oper-State {
            type segment-vpn:oper-state;
            config false;
            description "Operational status." ;
        }
        leaf sync-state {
            type segment-vpn:sync-state;
            config false;
            description "Sync status." ;
        }
        leaf start-time {
            type yang:date-and-time;
            description "Service lifecycle: request for service start
            time." ;
        }
    }

    container composed-vpns{
        description "";
        list composed-vpn {
            key "vpn-id";
            description "List for composed VPNs.";
```

```
            uses composedvpn;
        }
    }

    grouping composedvpn {
        description "ComposedVPN Grouping.";
        leaf vpn-id {
            type yang:uuid;
            description "Composed VPN identifier." ;
        }
        leaf vpn-name {
            type string  {length "0..200";}
            description "Composed VPN Name. Local administration meaning" ;
        }
        leaf customer-name {
            type yang:uuid;
            description
              "Name of the customer that actually uses the VPN service.
              In the case that any intermediary (e.g., Tier-2 provider
              or partner) sells the VPN service to their end user
              on behalf of the original service provider (e.g., Tier-1
              provider), the original service provider may require the
              customer name to provide smooth activation/commissioning
              and operation for the service." ;
        }
        uses vpn-basic;
        list segment-vpn {
            key "vpn-id";
            description "SegVpn list ";
            uses segment-vpn:VPN;
        }
        list access-point {
            key "tp-id";
            description "TP list of the access links which associated
            with CE and PE";
            uses segment-vpn:pe-termination-point;
        }
    }
}
<CODE ENDS>
```

7. Segment VPN YANG Module

```
<CODE BEGINS> file "ietf-segment-vpn.yang"
module ietf-segment-vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-vpn";
  prefix segment-vpn;
```

```
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-key-chain {
    prefix keychain;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg>
     WG List:  <mailto:netmod@ietf.org>

     Editor:
       Roni Even
          <mailto:roni.even@huawei.com>
          Bo Wu
          <mailto:lana.wubo@huawei.com>
          Qin Wu
          <mailto:bill.wu@huawei.com>
          Cheng Ying
          <mailto:chengying10@chinaunicom.cn>";
  description
    "This YANG module defines a generic service configuration
     model for segment VPNs.";

  revision 2019-01-30 {
    reference
      "draft-opsawg-evenwu-yang-composed-vpn-02";
  }

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

  feature qos {
    description
      "Enables support of classes of services.";
  }

  feature qos-custom {
```

```
    description
      "Enables support of the custom QoS profile.";
  }

  feature hex-key-string {
    description
      "Support hexadecimal key string.";
  }

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

  identity tcp {
    base protocol-type;
    description
      "TCP protocol type.";
  }

  identity udp {
    base protocol-type;
    description
      "UDP protocol type.";
  }

  identity icmp {
    base protocol-type;
    description
      "ICMP protocol type.";
  }

  identity icmp6 {
    base protocol-type;
    description
      "ICMPv6 protocol type.";
  }

  identity gre {
    base protocol-type;
    description
      "GRE protocol type.";
  }

  identity ipip {
    base protocol-type;
    description
      "IP-in-IP protocol type.";
```

```
  }

  identity hop-by-hop {
    base protocol-type;
    description
      "Hop-by-Hop IPv6 header type.";
  }

  identity routing {
    base protocol-type;
    description
      "Routing IPv6 header type.";
  }

  identity esp {
    base protocol-type;
    description
      "ESP header type.";
  }

  identity ah {
    base protocol-type;
    description
      "AH header type.";
  }

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

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

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

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

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

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

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

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

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

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

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

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

```
  identity outbound {
    base qos-profile-direction;
    description
      "Identity for outbound direction.";
  }

  identity inbound {
    base qos-profile-direction;
    description
      "Identity for inbound direction.";
  }

  identity both {
    base qos-profile-direction;
    description
      "Identity for both inbound direction
       and outbound direction.";
  }

  typedef access-point-type {
    type enumeration {
      enum ce-peering {
        description
          "indicates access type with connection to CE";
      }
      enum remote-as-peering {
        description
          "indicates access type with connection to ASBR with opion A,B,C ";
      }
    }
    description
      "The access-point-type could be peering with CE or ASBR
       depending on which network that a PE interconnects with.";
  }

  typedef bgp-password-type {
    type string;
    description
      "Authentication Type (None, Simple Password, Keyed MD5,
       Meticulous Keyed MD5, Keyed SHA1, Meticulous Keyed SHA1";
  }

  typedef topology-role {
    type enumeration {
      enum hub {
        description
          "hub";
      }
```

```
      enum spoke {
        description
          "spoke";
      }
      enum other {
        description
          "other";
      }
    }
    description
      "Topo Node Role.";
  }

  typedef qos-config-type {
    type enumeration {
      enum template {
        description
          "standard.";
      }
      enum customer {
        description
          "custom.";
      }
    }
    description
      "Qos Config Type.";
  }

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

  typedef tp-type {
    type enumeration {
      enum phys-tp {
        description
          "Physical termination point";
```

```
      }
      enum ctp {
        description
          "CTP";
      }
      enum trunk {
        description
          "TRUNK";
      }
      enum loopback {
        description
          "LoopBack";
      }
      enum tppool {
        description
          "TPPool";
      }
    }
    description
      "Tp Type.";
  }

  typedef layer-rate {
    type enumeration {
      enum lr-unknow {
        description
          "Layer Rate UNKNOW.";
      }
      enum lr-ip {
        description
          "Layer Rate IP.";
      }
      enum lr-eth {
        description
          "Layer Rate Ethernet.";
      }
      enum lr_vxlan {
        description
          "Layer Rate VXLAN.";
      }
    }
    description
      "Layer Rate.";
  }

  typedef admin-state {
    type enumeration {
      enum active {
```

```
        description
          "Active status";
      }
      enum inactive {
        description
          "Inactive status";
      }
      enum partial {
        description
          "Partial status";
      }
    }
    description
      "Admin State.";
  }

  typedef oper-state {
    type enumeration {
      enum up {
        description
          "Up status";
      }
      enum down {
        description
          "Down status";
      }
      enum degrade {
        description
          "Degrade status";
      }
    }
    description
      "Operational Status.";
  }

  typedef sync-state {
    type enumeration {
      enum sync {
        description
          "Sync status";
      }
      enum out-sync {
        description
          "Out sync status";
      }
    }
    description
      "Sync Status";
```

```
  }

  typedef eth-encap-type {
    type enumeration {
      enum default {
        description
          "DEFAULT";
      }
      enum dot1q {
        description
          "DOT1Q";
      }
      enum qinq {
        description
          "QINQ";
      }
      enum untag {
        description
          "UNTAG";
      }
    }
    description
      "Ethernet Encap Type.";
  }

  typedef protocol-type {
    type enumeration {
      enum static {
        description
          "Static Routing";
      }
      enum bgp {
        description
          "bgp";
      }
      enum rip {
        description
          "rip";
      }
      enum ospf {
        description
          "ospf";
      }
      enum isis {
        description
          "isis";
      }
    }
```

```
    description
      "Routing Protocol Type";
  }

  typedef tunnel-type {
    type enumeration {
      enum MPLS {
        description
          "MPLS";
      }
      enum MPLS-TP {
        description
          "MPLS-TP";
      }
      enum MPLS-SR {
        description
          "MPLS Segment Routing";
      }
      enum SRv6 {
        description
          "SRv6";
      }
    }
    description
      "VPN Tunnel Type.";
  }

  typedef service-type {
    type enumeration {
      enum l3vpn {
        description
          "l3vpn";
      }
      enum l2vpn {
        description
          "l2vpn";
      }
    }
    description
      "VPN Service Type.";
  }

  typedef vpn-topology {
    type enumeration {
      enum point-to-point {
        description
          "point to point";
      }
```

```
      enum any-to-any {
        description
          "any to any";
      }
      enum hub-spoke {
        description
          "hub and spoke VPN topology.";
      }
      enum hub-spoke-disjoint {
        description
          "Hub and spoke VPN topology where
           Hubs cannot communicate with each other ";
      }
    }
    description
      "Topology.";
  }

  typedef ethernet-action {
    type enumeration {
      enum nop {
        description
          "nop";
      }
      enum untag {
        description
          "UNTAG";
      }
      enum stacking {
        description
          "STACKING";
      }
    }
    description
      "Ethernet Action.";
  }

  typedef color-type {
    type enumeration {
      enum green {
        description
          "green";
      }
      enum yellow {
        description
          "yellow";
      }
      enum red {
```

```
        description
          "red";
      }
      enum all {
        description
          "all";
      }
    }
    description
      "Color Type.";
  }

  typedef action-type {
    type enumeration {
      enum nop {
        description
          "nop";
      }
      enum bandwidth {
        description
          "bandwidth";
      }
      enum pass {
        description
          "pass";
      }
      enum discard {
        description
          "discard";
      }
      enum remark {
        description
          "remark";
      }
      enum redirect {
        description
          "redirect";
      }
      enum recolor {
        description
          "recolor";
      }
      enum addRt {
        description
          "addRt";
      }
    }
    description
```

```
      "Action Type";
  }

  typedef pwtagmode {
    type enumeration {
      enum raw {
        description
          "RAW";
      }
      enum tagged {
        description
          "TAGGED";
      }
    }
    description
      "PWTagMode";
  }

  grouping QinQVlan {
    description
      "QinQVlan Grouping.";
    leaf-list cvlan {
      type uint64;
      description
        "cvlan List.";
    }
    leaf svlan {
      type uint64;
      description
        "svlan.";
    }
  }

  grouping Dot1QVlan {
    description
      "Dot1QVlan Grouping.";
    leaf-list dot1q {
      type uint64;
      description
        "dot1q Vlan List";
    }
  }

  grouping tp-connection-type {
    description
      "Tp Type Spec Grouping.";
    choice connection-type {
      description
```

```
        "Spec Value";
      case lr-eth {
        container eth {
          description
            "ethernetSpec";
          uses ethernet-spec;
        }
      }
      case lr-ip {
        container ip {
          description
            "ipSpec";
          uses ipspec;
        }
      }
      case lr-pw {
        container pw {
          description
            "PwSpec";
          uses pwspec;
        }
      }
    }
  }

  grouping security-authentication {
    container authentication {
      description
        "Authentication parameters.";
    }
    description
      "This grouping defines authentication parameters for a site.";
  }

  grouping security-encryption {
    container encryption {
      if-feature "encryption";
      leaf enabled {
        type boolean;
        default "false";
        description
          "If true, traffic encryption on the connection is required.";
      }
      leaf layer {
        when "../enabled = 'true'" {
          description
            " Require a value for layer when enabled is true.";
        }
```

```
        type enumeration {
          enum layer2 {
            description
              "Encryption will occur at Layer 2.";
          }
          enum layer3 {
            description
              "Encryption will occur at Layer 3.
               For example, IPsec may be used when
               a customer requests Layer 3 encryption.";
          }
        }
        description
          "Layer on which encryption is applied.";
      }
      leaf algorithm {
        type string;
        description
          "Encryption algorithm to be used.";
      }
      choice key-type {
        default "psk";
        case psk {
          leaf preshared-key {
            type string;
            description
              " Pre-Shared Key(PSK) coming from customer.";
          }
        }
        description
          "Type of keys to be used.";
      }
      description
        "Encryption parameters.";
    }
    description
      "This grouping defines encryption parameters for a site.";
  }

  grouping security-attribute {
    container security {
      uses security-authentication;
      uses security-encryption;
      description
        "Site-specific security parameters.";
    }
    description
      "Grouping for security parameters.";
```

```
  }

  grouping flow-definition {
    container match-flow {
      leaf dscp {
        type inet:dscp;
        description
          "DSCP value.";
      }
      leaf exp {
        type inet:dscp;
        description
          "EXP value.";
      }
      leaf dot1p {
        type uint8 {
          range "0..7";
        }
        description
          "802.1p matching.";
      }
      leaf ipv4-src-prefix {
        type inet:ipv4-prefix;
        description
          "Match on IPv4 src address.";
      }
      leaf ipv6-src-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 src address.";
      }
      leaf ipv4-dst-prefix {
        type inet:ipv4-prefix;
        description
          "Match on IPv4 dst address.";
      }
      leaf ipv6-dst-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 dst address.";
      }
      leaf l4-src-port {
        type inet:port-number;
        must 'current() < ../l4-src-port-range/lower-port or current() > ../l4-
src-port-range/upper-port' {
          description
            "If l4-src-port and l4-src-port-range/lower-port and
             upper-port are set at the same time, l4-src-port
```

```
            should not overlap with l4-src-port-range.";
```

```
        }
        description
          "Match on Layer 4 src port.";
      }
      leaf-list peer-remote-node {
        type string;
        description
          "Identify a peer remote node as traffic destination.";
      }
      container l4-src-port-range {
        leaf lower-port {
          type inet:port-number;
          description
            "Lower boundary for port.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            description
              "Upper boundary for port.  If it
               exists, the upper boundary must be
               higher than the lower boundary.";
          }
          description
            "Upper boundary for port.";
        }
        description
          "Match on Layer 4 src port range.  When
           only the lower-port is present, it represents
           a single port.  When both the lower-port and
           upper-port are specified, it implies
           a range inclusive of both values.";
      }
      leaf l4-dst-port {
        type inet:port-number;
        must 'current() < ../l4-dst-port-range/lower-port or current() > ../l4-
dst-port-range/upper-port' {
          description
            "If l4-dst-port and l4-dst-port-range/lower-port
             and upper-port are set at the same time,
             l4-dst-port should not overlap with
             l4-src-port-range.";
        }
        description
          "Match on Layer 4 dst port.";
      }
      container l4-dst-port-range {
        leaf lower-port {
```

```
          type inet:port-number;
```

```
          description
            "Lower boundary for port.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            description
              "Upper boundary must be
               higher than lower boundary.";
          }
          description
            "Upper boundary for port.  If it exists,
             upper boundary must be higher than lower
             boundary.";
        }
        description
          "Match on Layer 4 dst port range.  When only
           lower-port is present, it represents a single
           port.  When both lower-port and upper-port are
           specified, it implies a range inclusive of both
           values.";
      }
      leaf src-mac {
        type yang:mac-address;
        description
          "Source MAC.";
      }
      leaf dst-mac {
        type yang:mac-address;
        description
          "Destination MAC.";
      }
      leaf protocol-field {
        type union {
          type uint8;
          type identityref {
            base protocol-type;
          }
        }
        description
          "Match on IPv4 protocol or IPv6 Next Header field.";
      }
      description
        "Describes flow-matching criteria.";
    }
    description
      "Flow definition based on criteria.";
  }
```

```
  grouping service-qos-profile {
    container qos {
      if-feature "qos";
      container qos-classification-policy {
        list rule {
          key "id";
          ordered-by user;
          leaf id {
            type string;
            description
              "A description identifying the
               qos-classification-policy rule.";
          }
          choice match-type {
            default "match-flow";
            case match-flow {
              uses flow-definition;
            }
            case match-application {
              leaf match-application {
                type identityref {
                  base customer-application;
                }
                description
                  "Defines the application to match.";
              }
            }
            description
              "Choice for classification.";
          }
          leaf target-class-id {
            type string;
            description
              "Identification of the class of service.
               This identifier is internal to the administration.";
          }
          description
            "List of marking rules.";
        }
        description
          "Configuration of the traffic classification policy.";
      }
      container qos-profile {
        choice qos-profile {
          description
            "Choice for QoS profile.
             Can be standard profile or customized profile.";
          case standard {
```

```
            description
              "Standard QoS profile.";
            leaf profile {
              type string;
              description
                "QoS profile to be used.";
            }
          }
          case custom {
            description
              "Customized QoS profile.";
            container classes {
              if-feature "qos-custom";
              list class {
                key "class-id";
                leaf class-id {
                  type string;
                  description
                    "Identification of the class of service.
                     This identifier is internal to the
                     administration.";
                }
                leaf direction {
                  type identityref {
                    base qos-profile-direction;
                  }
                  default "both";
                  description
                    "The direction to which the QoS profile
                     is applied.";
                }
                leaf rate-limit {
                  type decimal64 {
                    fraction-digits 5;
                    range "0..100";
                  }
                  units "percent";
                  description
                    "To be used if the class must be rate-limited.
                     Expressed as percentage of the service
                     bandwidth.";
                }
                container latency {
                  choice flavor {
                    case lowest {
                      leaf use-lowest-latency {
                        type empty;
                        description
```

```
                    "The traffic class should use the path with the
                     lowest latency.";
                }
              }
              case boundary {
                leaf latency-boundary {
                  type uint16;
                  units "msec";
                  default "400";
                  description
                    "The traffic class should use a path with a
                     defined maximum latency.";
                }
              }
              description
                "Latency constraint on the traffic class.";
            }
            description
              "Latency constraint on the traffic class.";
          }
          container jitter {
            choice flavor {
              case lowest {
                leaf use-lowest-jitter {
                  type empty;
                  description
                    "The traffic class should use the path with the
                     lowest jitter.";
                }
              }
              case boundary {
                leaf latency-boundary {
                  type uint32;
                  units "usec";
                  default "40000";
                  description
                    "The traffic class should use a path with a
                     defined maximum jitter.";
                }
              }
              description
                "Jitter constraint on the traffic class.";
            }
            description
              "Jitter constraint on the traffic class.";
          }
          container bandwidth {
            leaf guaranteed-bw-percent {
```

```
                    type decimal64 {
                      fraction-digits 5;
                      range "0..100";
                    }
                    units "percent";
                    mandatory true;
                    description
                      "To be used to define the guaranteed bandwidth
                       as a percentage of the available service bandwidth.";
                  }
                  leaf end-to-end {
                    type empty;
                    description
                      "Used if the bandwidth reservation
                       must be done on the MPLS network too.";
                  }
                  description
                    "Bandwidth constraint on the traffic class.";
                }
                description
                  "List of classes of services.";
              }
              description
                "Container for list of classes of services.";
            }
          }
        }
        description
          "QoS profile configuration.";
      }
      description
        "QoS configuration.";
    }
    description
      "This grouping defines QoS parameters for a segment network.";
  }

  grouping remote-peer-tp {
    description
      "remote-peer-tp Grouping.";
    leaf remote-id {
      type yang:uuid;
      description
        "Router ID of the remote peer";
    }
    leaf location {
      type string {
        length "0..400";
```

```
      }
      description
        "CE device location ";
    }
    leaf remote-tp-address {
      type inet:ip-address;
      description
        "TP IP address";
    }
    leaf remote-node-id {
      type yang:uuid;
      description
        "directly connected NE node ID, only valid in
         asbr ";
    }
    leaf remote-tp-id {
      type yang:uuid;
      description
        "Directly connected TP id, only valid in asbr";
    }
  }

  grouping tp-connection-specific-attribute {
    description
      "tp connectin specific attributes";
    list connection {
      key "connection-class";
      leaf connection-class {
        type layer-rate;
        description
          "connection class and has one to one
         relation with the corresponding layer.";
      }
      uses tp-connection-type;
      description
        "typeSpecList";
    }
    container security-attribute {
      description
        "tp security Parameters.";
      uses security-attribute;
    }
    container qos-attribute {
      description
        "tp Qos Parameters.";
      uses segment-service-basic;
      uses service-qos-profile;
    }
```

```
    container protection-attribute {
      description
        "tp protection parameters.";
      leaf access-priority {
        type uint32;
        default "100";
        description
          "Defines the priority for the access.
           The higher the access-priority value,
           the higher the preference of the
           access will be.";
      }
    }
  }

  grouping tp-common-attribute {
    description
      "tp-common-attribute Grouping.";
    leaf tp-id {
      type yang:uuid;
      description
        "An identifier for termination point on a node.";
    }
    leaf tp-name {
      type string {
        length "0..200";
      }
      description
        "The termination point Name on a node. It conforms to
         name rule defined in system. Example FE0/0/1, GE1/2/1.1,
         Eth-Trunk1.1, etc";
    }
    leaf node-id {
      type yang:uuid;
      description
        "Identifier for a node.";
    }
    leaf access-point-type {
      type access-point-type;
      description
        "access-point-type, for example:peering with CE ";
    }
    leaf inter-as-option {
      type enumeration {
        enum optiona {
          description
            "Inter-AS Option A";
        }
```

```
        enum optionb {
          description
            "Inter-AS Option B";
        }
        enum optionc {
          description
            "Inter-AS Option C";
        }
      }
      description
        "Foo";
    }
    leaf topology-role {
      type topology-role;
      description
        "hub/spoke role, etc";
    }
  }

  grouping routing-protcol {
    description
      "Routing Protocol Grouping.";
    leaf type {
      type protocol-type;
      description
        "Protocol type";
    }
    choice para {
      description
        "para";
      case static {
        list static {
          key "index";
          uses static-config;
          description
            "staticRouteItems";
        }
      }
      case bgp {
        list bgp {
          key "index";
          uses bgp-config;
          description
            "bgpProtocols";
        }
      }
    }
  }
```

```
  grouping bgp-config {
    description
      "BGP Protocol Grouping.";
    leaf index {
      type uint32;
      description
        "index of BGP protocol item";
    }
    leaf autonomous-system {
      type uint32;
      mandatory true;
      description
        "Peer AS number in case the peer
         requests BGP routing.";
    }
    leaf-list address-family {
      type address-family;
      min-elements 1;
      description
        "If BGP is used on this site, this node
         contains configured value. This node
         contains at least one address family
         to be activated.";
    }
    leaf max-prefix {
      type int32;
      description
        "maximum number limit of prefixes.";
    }
    leaf peer-address {
      type inet:ip-address;
      description
        "peerIp";
    }
    leaf crypto-algorithm {
      type identityref {
        base keychain:crypto-algorithm;
      }
      mandatory true;
      description
        "Cryptographic algorithm associated with key.";
    }
    container key-string {
      description
        "The key string.";
      nacm:default-deny-all;
      choice key-string-style {
        description
```

```
          "Key string styles";
        case keystring {
          leaf keystring {
            type string;
            description
              "Key string in ASCII format.";
          }
        }
        case hexadecimal {
          if-feature "hex-key-string";
          leaf hexadecimal-string {
            type yang:hex-string;
            description
              "Key in hexadecimal string format.  When compared
               to ASCII, specification in hexadecimal affords
               greater key entropy with the same number of
               internal key-string octets.  Additionally, it
               discourages usage of well-known words or
               numbers.";
          }
        }
      }
    }
  }

  grouping static-config {
    description
      "StaticRouteItem Grouping.";
    leaf index {
      type uint32;
      description
        "static item index";
    }
    leaf dest-cidr {
      type string;
      description
        "address prefix specifying the set of
         destination addresses for which the route may be
         used. ";
    }
    leaf egress-tp {
      type yang:uuid;
      description
        "egress tp";
    }
    leaf route-preference {
      type string;
      description
```

```
        "route priority. Ordinary, work route have
         higher priority.";
    }
    leaf next-hop {
      type inet:ip-address;
      description
        "Determines the outgoing interface and/or
         next-hop address(es), or a special operation to be
         performed on a packet..";
    }
  }

  grouping ethernet-spec {
    description
      "Ethernet Spec Grouping.";
    leaf access-type {
      type eth-encap-type;
      description
        "access frame type";
    }
    choice accessVlanValue {
      description
        "accessVlanValue";
      case qinq {
        container qinq {
          description
            "qinqVlan";
          uses QinQVlan;
        }
      }
      case dot1q {
        container dot1q {
          description
            "dot1q";
          uses Dot1QVlan;
        }
      }
    }
    leaf vlan-action {
      type ethernet-action;
      description
        "specify the action when the vlan is matched";
    }
    leaf action {
      type string {
        length "0..100";
      }
      description
```

```
        "specify the action value.";
    }
  }

  grouping pwspec {
    description
      "PwSpec Grouping.";
    leaf control-word {
      type boolean;
      default "false";
      description
        "control Word.";
    }
    leaf vlan-action {
      type pwtagmode;
      description
        "pw Vlan Action.";
    }
  }

  grouping ipspec {
    description
      "IpSpec Grouping.";
    leaf ip-address {
      type inet:ip-address;
      description
        "master IP address";
    }
    leaf mtu {
      type uint64;
      description
        "mtu for ip layer,scope:46~9600";
    }
  }

  grouping VPN {
    description
      "VPN Grouping.";
    leaf vpn-id {
      type yang:uuid;
      description
        "VPN Identifier.";
    }
    leaf vpn-name {
      type string {
        length "0..200";
      }
      description
```

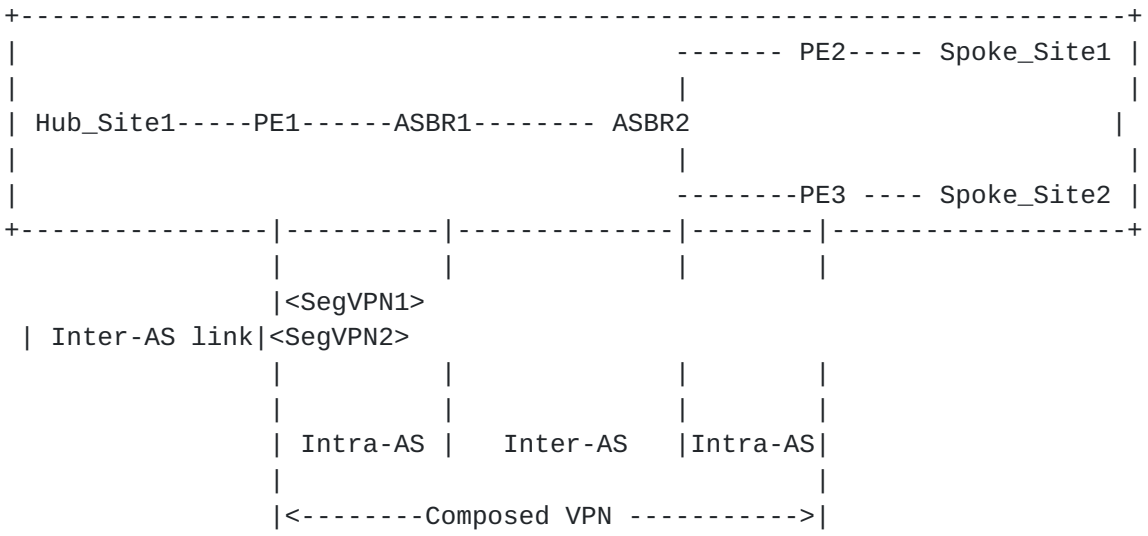
```
        "Human-readable name for the VPN service.";
    }
    leaf service-type {
      type service-type;
      description
        "The service type combines service types from
         RFC8299 (L3SM) and RFC8466 (L2SM),for example L3VPN,VPWS etc.
         It could be augmentated for future extensions.";
    }
    leaf topo {
      type vpn-topology;
      description
        "The VPN topology could be full-mesh,point-to-point
         and hub-spoke, others is reserved for future extensions.";
    }
    leaf tunnel-type {
      type tunnel-type;
      description
        "Tunnel Type:LDP&#65306;LDP Tunnel,RSVP-TE&#65306;RSVP-TE Tunnel
         SR-TE&#65306;SR-TE Tunnel,MPLS-TP&#65306;MPLS-TP
Tunnel,VXLAN&#65306;VXLAN Tunnel
        ";
    }
    leaf admin-state {
      type admin-state;
      description
        "administrative status.";
    }
    leaf oper-state {
      type oper-state;
      config false;
      description
        "Operational status.";
    }
    leaf sync-state {
      type sync-state;
      config false;
      description
        "Sync status.";
    }
    list access-point {
      key "tp-id";
      description
        "TP list of the access links which associated
         with PE and CE or ASBR";
      uses pe-termination-point;
    }
  }
```

```
  grouping pe-termination-point {
    description
      "grouping for termination points.";
    uses tp-common-attribute;
    container peer-remote-node {
      description
        "TP Peering Information, including CE
         peering and ASBR peering.";
    uses remote-peer-tp;
    }
    container tp-connection-specific-attribute {
      description
        "Termination point basic info.";
      uses tp-connection-specific-attribute;
    }
    list routing-protocol {
      key "type";
      description
        "route protocol spec.";
      uses routing-protcol;
    }
  }

  grouping segment-service-basic {
    leaf svc-input-bandwidth {
      type uint64;
      units "bps";
      mandatory true;
      description
        "From the customer site's perspective, the service
         input bandwidth of the connection or download
         bandwidth from the SP to the site.";
    }
    leaf svc-output-bandwidth {
      type uint64;
      units "bps";
      mandatory true;
      description
        "From the customer site's perspective, the service
         output bandwidth of the connection or upload
         bandwidth from the site to the SP.";
    }
    leaf svc-mtu {
      type uint16;
      units "bytes";
      mandatory true;
      description
        "MTU at service level.  If the service is IP,
```

```
         it refers to the IP MTU.  If CsC is enabled,
         the requested 'svc-mtu' leaf will refer to the
         MPLS MTU and not to the IP MTU.";
    }
    description
      "Defines basic service parameters for a site.";
  }

  container segment-vpns {
    list segment-vpn {
      key "index";
      description
        "Segment Vpn list.";
      leaf index {
        type uint32;
        description
          "index of segment VPN in a composed VPN.";
      }
      uses VPN;
    }
    description
      "Container for Segment VPN.";
  }
}
<CODE ENDS>
```

8. Service Model Usage Example

This section provides an example of how a management system can use this model to configure an IP VPN service on network elements.



Composed VPN Service Model Usage Example

In this example, we want to achieve the provisioning of an end to end VPN service for three sites using a Hub-and-Spoke VPN service topology. The end to end VPN service is stitched by two segmented VPN.

The following XML snippet describes the overall simplified service configuration of this composed VPN.

```
<?xml version="1.0"?>
<composed-vpns xmlns="urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc">
     <composed-vpn>
      <vpn-id>12456487</vpn-id>
      <topo>hub-spoke</topo>
      <service-type>hybrid</service-type>
      <segment-vpn>
        <index>1</index>
          <vpn-id>111<vpn-id>
          <topo>hub-spoke</topo>
          <service-type>l2vpn</service-type>
          <access-point>
             <tp-id>ap1-tp1</tp-id>
             <node-id>PE1</node-id>
             <topology-role>hub</topology-role>
             <peer-remote-node>
               <remote-node-id>Hub_Site1</remote-node-id>
             </peer-remote-node>
              <tp-connection-specific-attribute>
               <qos-attribute>
               <svc-mtu>1514</svc-mtu>
               <svc-input-bandwidth>10000000</svc-input-bandwidth>
               <svc-output-bandwidth>10000000</svc-output-bandwidth>
                </qos-attribute>
             </tp-connection-specific-attribute>
             <routing-protocol>
               <type>bgp</type>
               <bgp>
               <as-no>AS1</as-no>
               </bgp>
            </routing-protocol>
            </access-point>
            <access-point>
             <tp-id>ap1-tp2</tp-id>
             <node-id>ASBR1</node-id>
             <topo-role>hub</topo-role>
             <peer-remote-node>
               <remote-node-id>ASBR2</remote-node-id>
             </peer-remote-node>
             <inter-AS-option>Option A</inter-AS-option>
```

```
                    <tp-connection-specific-attribute>
                     <qos-attribute>
                     <svc-mtu>1514</svc-mtu>
                     <svc-input-bandwidth>10000000</svc-input-bandwidth>
                     <svc-output-bandwidth>10000000</svc-output-bandwidth>
                      </qos-attribute>
                    </tp-connection-specific-attribute>
                    <routing-protocol>
                      <type>bgp</type>
                      <bgp>
                      <as-no>AS1</as-no>
                      </bgp>
                   </routing-protocol>
                 </access-point>
               </segment-vpn>
               <segment-vpn>
                 <index>2</index>
                   <vpn-id>222<vpn-id>
                   <topo>hub-spoke</topo>
                   <service-type>l3vpn</service-type>
                   <access-point>
                       <tp-id>ap2-tp2</tp-id>
                       <node-id>PE2</node-id>
                       <topo-role>spoke</topo-role>
                      <peer-remote-node>
                        <remote-node-id>Spoke_Site1</remote-node-id>
                      </peer-remote-node>
                        <qos-attribute>
                        <svc-mtu>1514</svc-mtu>
                        <svc-input-bandwidth>10000000</svc-input-bandwidth>
                        <svc-output-bandwidth>10000000</svc-output-bandwidth>
                         </qos-attribute>
                        <routing-protocol>
                         <type>bgp</type>
                          <bgp>
                          <as-no>ASXXX</as-no>
                          </bgp>
                        </routing-protocol>
                      </access-point>
                      <access-point>
                      <tp-id>ap2-tp1</tp-id>
                      <node-id>PE3</node-id>
                       <topo-role>spoke</topo-role>
                       <peer-remote-node>
                        <remote-node-id>Spoke_Site2</remote-node-id>
                      </peer-remote-node>
                        <qos-attribute>
                        <svc-mtu>1514</svc-mtu>
```

```
                    <svc-input-bandwidth>10000000</svc-input-bandwidth>
                    <svc-output-bandwidth>10000000</svc-output-bandwidth>
                     </qos-attribute>
                  <routing-protocol>
                    <type>bgp</type>
                    <bgp>
                    <as-no>ASXXX</as-no>
                    </bgp>
                 <routing-protocol>
                  </access-point>
                  <access-point>
                  <tp-id>ap2-tp3</tp-id>
                  <node-id>ASBR2</node-id>
                   <topo-role>hub</topo-role>
                  <peer-remote-node>
                    <remote-node-id>ASBR1</remote-node-id>
                  </peer-remote-node>
                   <qos-attribute>
                    <svc-mtu>1514</svc-mtu>
                    <svc-input-bandwidth>10000000</svc-input-bandwidth>
                    <svc-output-bandwidth>10000000</svc-output-bandwidth>
                     </qos-attribute>
                  <routing-protocol>
                    <type>bgp</type>
                    <bgp>
                    <as-no>interAS-1</as-no>
                    </bgp>
                <routing-protocol>
              </access-point>
          </segment-vpn>
        </composed-vpn>
    </composed-vpns>
```

9. Interaction with other YANG models

As expressed in Section 4, this composed VPN service model is intended to be instantiated in a management system and not directly on network elements.

The management system's role will be to configure the network elements. The management system may be modular and distinguish the component instantiating the service model (let's call it "service component") from the component responsible for network element configuration (let's call it "configuration component"). The service is built from a combination of network elements and protocols configuration which also include various aspects of the underlying network infrastructure, including functions/devices and their subsystems, and relevant protocols operating at the link and network

layers across multiple device. Therefore there will be a strong relationship between the abstracted view provided by this service model and the detailed configuration view that will be provided by specific configuration models for network elements.

The service component will take input from customer service model such as L3SM service model [RFC8299] or composed VPN service model and translate it into segment VPN in each domain and then further break down the segment VPN into detailed configuration view that will be provided by specific configuration models for network elements.

10. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] 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:

- /composed-vpns/composed-vpn

  The entries in the list above include the whole composed vpn service configurations which the customer subscribes, and indirectly create or modify the PE,CE and ASBR device configurations. Unexpected changes to these entries could lead to service disruption and/or network misbehavior.

- /composed-vpns/composed-vpn/segment-vpn

  The entries in the list above include the access points configurations. As above, unexpected changes to these entries could lead to service disruption and/or network misbehavior.

- /composed-vpns/composed-vpn/access-point

  The entries in the list above include the access points configurations. As above, unexpected changes to these entries could lead to service disruption and/or network misbehavior.

11. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

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

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

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
-------------------------------------------------------------------
          Name: ietf-composite-vpn-svc
          Namespace: urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc
          Prefix: composed-svc
          Reference: RFC xxxx
          Name: ietf-segmented-vpn
          Namespace: urn:ietf:params:xml:ns:yang:ietf-segment-vpn
          Prefix: segment-vpn
          Reference: RFC xxxx
-------------------------------------------------------------------
```

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[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>.

[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>.

[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>.

12.2. Informative References

[RFC1136] Hares, S. and D. Katz, "Administrative Domains and Routing Domains: A model for routing in the Internet", RFC 1136, DOI 10.17487/RFC1136, December 1989, <https://www.rfc-editor.org/info/rfc1136>.

[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>.

Appendix A. Acknowledges

Geng Liang,Congfeng Xie, Chen Rui, LiYa Zhang,Hui Deng contributed to an earlier version of [I-D.chen-opsawg-composite-vpn-dm]. We would like to thank the authors of that document on the operators' view for the PE-based VPN service configuration for material that assisted in thinking about this document.

Authors' Addresses

Roni Even
Huawei Technologies,Co.,Ltd
Tel Aviv
Israel

Email: roni.even@huawei.com

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

Email: lana.wubo@huawei.com

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

Email: bill.wu@huawei.com

YingCheng
China Unicom
No.21 Financial Street, XiCheng District
Beijing 100033
China

Email: chengying10@chinaunicom.cn