RTWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2019

S. Wood, Ed.
Cisco Systems
B. Wu, Ed.
Q. Wu, Ed.
Huawei
C. Menezes
HPE Aruba

June 29, 2019

YANG Data Model for SD-WAN OSE service delivery
draft-wood-rtgwg-sdwan-ose-yang-01

Abstract

This document defines two SD-WAN OSE Open SD-WAN Exchange(OSE) service YANG modules to enable the orchestrator in the enterprise network to implement SD-WAN inter-domain reachability and connectivity services and application aware traffic steering services.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Software-Defined WAN networking (SDWAN) has become a major new technology in Wide Area Networking. SDWAN architecture is a combination of data and control plane orchestration, proprietary control-plane enhancements as well as single-hop, CE-CE data-planes often referred to as "fabrics". On top of this infrastructure, centralized network policy administration and distribution is provided to achieve a specific set of network outcomes or use-cases.

As a result of the use-case driven approach, SDWAN technology solutions often encode choices about data-plane and protocol operation into associated data-plane, control-plane and controller subsystems. These choices are intended to simplify deployment of SDWAN use-cases, but often result in systems that are not compatible and network elements that cannot interoperate in the manner of traditional, standards-based IP networks.

The Open SD-WAN Exchange (OSE) is an open framework to allow for one vendor SD-WAN domain to communicate with another vendor SD-WAN domain. The goal is to enable interworking between different SDWAN domains via the definition of standard service behaviours as well as standard data models to define those services. The underlying service implementation in each domain is only relevant in that it meets the specified service definition. To create OSE SD-WAN services across domain, a higher layer orchestrator may use generic API calls based on the service models to create the desired SDWAN services within each domain via the serving SDWAN manager.

The services currently defined by specification [OSE] include:

o OSE Gateway Reachability services

o Application Path Management Services

This document defines two SD-WAN service YANG modules to enable the orchestrator in the enterprise network to implement SD-WAN inter-domain reachability and connectivity services and application aware traffic steering services. The SD-WAN OSE Service Model is for enterprise own network.

1.1. Terminology

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

o client

o server

o configuration data

o state data

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

o augment

o data model

o data node

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

1.2. Tree diagram

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

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 RFC2119 [RFC2119].

3. 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.

SDWAN Manager: SDWAN Manager is the domain specific manager and controller required to configure, manage and control a particular SDWAN domain. To create OSE SDWAN services, a higher layer orchestrator may use OSE defined API calls to create the desired SDWAN services within each domain via the serving SDWAN manager.

Client Orchestration: The Client Orchestration layer is an abstraction of a service level orchestrator or software that implements control the the SDWAN through the defined OSE APIs. The OSE service specifications do not specify the functions and procedures within this entity apart from the fact that it would use the OSE APIs. The client orchestration layer is a functional block which would implement OSE API calls to one or more serving SDWAN managers.

SD-WAN controller: The SD-WAN Controller is a reference block that encompasses the network control-plane functions required to operate the SDWAN network. The SD-WAN network controller delivers control-plane/data-plane separation the is the realization of SDN architecture within the SD-WAN usecase. Each SD-WAN network controller is managed and configured by the SD-WAN manager. The interface between SDWAN network controller and SD-WAN network manager for this purpose is currently outside the scope of the document.

4. The SD-WAN OSE Service Model Requirements

This section provides a common definition for service types required across different SD-WAN vendor domains. The Open SD-WAN Exchange (OSE) model focuses on interoperability between domains, rather than specifying standard protocol and operations with each SD-WAN domain.

The OSE interoperability models focus on the definition of a standard set of service models and parameters that can be implemented in an SDWAN management system to configure interoperable services within an SDWAN domain and between SDWAN domains.

4.1. Reachability & Route Exchange Requirements

In [OSE]SD-WAN reference model, it is assumed that communication between sites in different domains happening via the OSE gateway which suggests that traffic spanning the domains will be backhauled to the OSE gateway. The interfaces between the gateways are called NNI interfaces. The interconnection between OSE gateways includes the following:

- OSE gateway interconnection: There may be multiple links between OSE gateways. To mitigate the constraints of the underlying network between the OSE gateway, an IP overlay tunnel needs to be established, and provide simple configuration and operation. It is assumed that GRE and IKE based IPsec can be used.

- Route exchange: Provides L3 reachability information exchange to facilitate L3 connectivity between SD-WAN domains.

4.2. Network Segmentation Requirements

In addition to the basic connection, the inter-domain interconnection needs to ensure the interworking of network segments. Network segmentation divides an enterprise network into different traffic or routing contexts to provide clear separation of traffic of each segment. These are often referred to as Virtual networks. The most common technology of network segmentation are virtual LANs, or VLANs,

for Layer 2 implementation, and virtual routing and forwarding, or VRF, for Layer 3 implementation. For traffic flowing across SD-WAN domains boundaries, segmentation must be preserved. A method of configuration is required to ensure per segment traffic flow separation while passing through SD-WAN domain boundaries. Such use case is also described in Augmenting RFC4364 Technology to Provide Secure Layer L3VPNs over Public Infrastructure [I-D.rosen-bess-secure-l3vpn]. Therefore, as specified in BGP/MPLS IP VPN [RFC4364]for Multi-AS use cases, it is assumed that MP-BGP with Option B is preferred due to its ease for provisioning, segmentation and operations. For some cases when Option B is not available, separate instances of BGP to be configured on a per VRF basis, which is Option A. This may require more involvement from the provisioning systems.

4.3. Path Management Requirements

As specified in ONUG SD-WAN whitepaper[ONUG], dynamic path selection is one of the core features of the SD-WAN, which site-to-site packets can be distributed across multiple WAN connections in real-time, based on current link metrics such as delay, loss and jitter. In this model, a path is considered to be an access network and not a path within an access network, although the latter is not precluded. For business critical applications traversing SD-WAN domains, policies via standardized APIs need to be provisioned to guarantee end-to-end SLA requirements and each domain is responsible for implementing consistent policy enforcement behaviour. Since inter-domain traffic are all backhauled by the OSE gateways, each part of the traversing path needs to be consistent.

Note: A method needs to be specified for budgeting end-to-end delay across multiple domains - delay/loss/jitter needs to be shared so that each domain can compute the total path, determine who's violating and then execute path change.

5. Service Model Usage

```
             +-------------------------------------+
             |        Client Orchestration         |
             +-------------------------------------+
                 |                               |
    ose-path-svc |          ose-reachability-svc |
       Model     |             Model             |
                 |                               |
         +----------------+              +----------------+
         | SDWAN manager  |              | SDWAN manager  |
         +----------------+              +-------+--------+
                 |                               |
                 | NETCONF/CLI ...               |
                 |                               |
   +-------------------------+            +------+-------------------+
   |     SDWAN Domin #1      |            |     SDWAN Domin #2       |
   |                         |            |                          |
   |++++++++  +++++++++++++  |   NNI      | +++++++++++++   ++++++++ |
   |+Branch+--+OSE Gateway+--+------------+-+OSE Gateway+---+Branch+ |
   |++++++++  +++++++++++++  |            | +++++++++++++   ++++++++ |
   |                         |            |                          |
   |Site A                   |            |                   Site B |
   +-------------------------+            +--------------------------+
```

Figure 1

As shown in figure 1, communication between branch sites sitting in domain#1 and domain#2 happens via a border element referred to as the OSE Gateway. This border element interworks the SDWAN control and data plane of the SDWAN domain to a common, defined NNI carrying routing information to establish reachability between domains. It also carries segmentation identifiers that are mutually agreed and configured within each OSE gateway by the domain serving SDWAN manager. The serving SDWAN manager in each respective domain is configured by the operator with information about which segments in each domain are to be connected.

Segment connections must be 1:1 across each OSE gateway.

Note: The detailed control and data plane specifications for the OSE Gateway NNI will refer to the definition of the relevant SD-WAN protocols in the IETF.

The ONUG SD-WAN service YANG model provides an abstracted interface to configure, and manage the components of an SD-WAN service. The components of the SD-WAN service include the OSE Gateway Service component and the Path Management Service component. OSE gateway service component defines Reachability and Route Exchange Segmentation requirements for OSE Gateway devices while path

management service component defines path management policy for domain serving SD-WAN managers.

A typical usage for this model is to generate Restconf[RFC8040] API used between Client Orchestraton layer and SDWAN manager and used by an enterprise operator to provision the inter-domain services. Before configuring the inter-domain path management policy service, the ose-reachability-svc model is used for the following configuration:

- Create one or more OSE gateways in the serving domain.

- Create underlying connections between the OSE gateway and other SD-WAN domain gateways, including control plane and data plane.

- Create overlay tunnels between the OSE gateway and other SD-WAN domain gateways with Tunnel setup parameters, such as IPsec Tunnel related authentication and encryption parameters.

- Create segment mappings between the OSE gateway and other SD-WAN domain gateways with segment related parameters, such as VLAN ID or VRF ID.

For the configuration of network elements may be done using NETCONF [RFC6241] or any other configuration (or "southbound") interface such as Command Line Interface (CLI) in combination with device-specific and protocol-specific YANG data models.

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

6. Design of the Data Model

The SD-WAN OSE service model currently has two YANG modules.

6.1. OSE Gateway Service Model

The aim of OSE Gateway module is to define parameters for connection setup between SD-WAN domains. As specified by RFC4364, this model defines Option A and Option B to interconnect the different domain. The option B allows one to minimize configuration inputs and allows the solution to scale really high because only the BGP RIBs store all the inter-AS / inter-SD-WAN VPN routes. MP-BGP can run a single label stack within the GRE tunnel, between the NNI nodes such that the MPLS label will be used for traffic segmentation. In the cases, where L3VPN Inter-AS Option B is not supported, revert to MP-BGP based Inter-AS VPN Option A while using MPLS labels. The option A

requires Orchestration layer to signal underlying SD-WAN domains to configure and instantiate VRF instances per tenant, as well as MP-BGP based L3VPN configuration and instantiation per tenant. This option can still run on GRE or IPSec tunnels while providing isolation from underlay changes and dependencies and MPLS label within the GRE tunnel will provide per tenant service level separation.

- ose-gateway: Gateway name and Gateway ID are specified for each domain.

- tunnel: describes encap-type in the interconnection points, and authentication and encryption are also specified to secure the interconnection between SD-WAN domains.

- ose-interworking-option: MP-BGP based L3VPN Inter-AS Option B with MPLS labels and Inter-AS Option A are defined.

- ose-gateway control plane peering:Control Plane peering between SD-WAN Edge Nodes which exchanges routes and additional reachability information as well as forward transit traffic. For good HA and resiliency characteristics, it is recommended to establish control plane sessions between each node.

- segment: to guarantee end to end secure traffic, the segment traffic from a specific domain needs to cross connect to the target segment through an OSE gateway.

The complete data hierarchy is presented as follows:

```
   module: ietf-ose-gateway-svc
     +--rw ose-gateways
        +--rw ose-gateway* [gw-id]
           +--rw gw-id           uint32
           +--rw gw-name?        string
           +--rw peer-list* [name]
           |  +--rw name                       string
           |  +--rw peer-gw-id?                uint32
           |  +--rw peer-gw-name?              string
           |  +--rw ose-interworking-option?   enumeration
           |  +--rw encap-type?                enumeration
           |  +--rw auth-type?                 enumeration
           |  +--rw crypto-option?             enumeration
           +--rw segment-list* [segment-name]
              +--rw segment-name     string
              +--rw vlan-id?         uint16 {ose-option-a}?
              +--rw vrf-id?          uint16 {ose-option-b}?
              +--rw segment-type?    enumeration
              +--rw crossconnects* [ccname]
                 +--rw ccname               string
                 +--rw gateway-reference?
                 |       -> ../../../peer-list/peer-gw-id
                 +--rw peer-seg-name?       string
                 +--rw peer-seg-id-vlan?    uint16 {ose-option-a}?
                 +--rw peer-seg-id-vrf?     uint16 {ose-option-b}?
```

6.2. OSE Path Service Model

Path management module defines automatic path selection policy for traffic across the domain. Policy control will take shape in the form of an ordered list. Each item in the list will be evaluated to match the traffic classifier. The first match will result in processing the matched traffic according to the associated link & path policy. In turn, the link & path policy will be framed in the context of the Performance SLA associated to the links and paths.

```
  +------------------+    +------------------+       +------------------+
  |                  |   \|    Link & Path   |/      |    Link&Path     |
  |Traffic Classifier+----+     Policy       +-------+  Performance     |
  |                  |   /|                  |\      |     SLAs         |
  +------------------+    +------------------+       +------------------+
```

figure 2

Traffic classification rules are handled by the "traffic-class" container. The traffic-classification-policy container is an ordered list of rules that match a flow or application and set the

appropriate business-priority and make link or path selection.This business priority can be factored into the path selection decision.

The client orchestrator can define the match using an application reference or a flow definition that is more specific (e.g., based on Layer 3 source and destination addresses, Layer 4 ports, and Layer 4 protocol).

The link or path selection is defined as a list of services properties. Describes the policy for how links should be selected for the specified traffic flow. The properties are as follows:

- mode: Describes the policy for how links should be selected for the specified traffic flow. Values are: 1-Automatic 2-Primary/ preferred 3-Lowest cost

- physical-port:describe the WAN port number

- service-type: Commodity refers to broadband Internet links; Wireless refers to subset of 3G/4G/LTE and upcoming 5G; Private refers to private circuits such as Ethernet, T1, etc

- service-provider: Specifies the name of provider per enumerated list of providers globally.

- path-selection-mode: Describes the policy for how paths should be selected for the specified traffic flow. This includes the policy option for portions of traffic to not be sent across the SD-WAN overlay tunnel. Values are: 1 - "Drop" 2 - "UnderNon overlay" 3 - "Overlay".

A custom SLA profile is defined as a list of services properties. The properties are as follows:

- delay: defines the latency constraint of a specific traffic class .

- jitter: defines the jitter constraint of a specific traffic class.

- loss: defines the loss constraint of a specific traffic class.

The complete data hierarchy is presented as follows:

```
   module: ietf-ose-path-svc
     +--rw path-svc
     |  +--rw path-policy* [policy-name]
     |     +--rw policy-name                   string
     |     +--rw flow-classification* [class-name]
     |     |  +--rw class-name           string
     |     |  +--rw dscp?                inet:dscp
     |     |  +--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 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 protocol-field?      union
     |     +--rw application-classification* [application-class-name]
     |     |  +--rw application-class-name    string
     |     |  +--rw category-id?              uint32
     |     |  +--rw application-id?           uint32
     |     +--rw user* [list-name]
     |     |  +--rw list-name     string
     |     |  +--rw user-id*      string
     |     |  +--rw user-group*   string
     |     +--rw site*                         uint32
     |     +--rw link-path-policy
     |        +--rw business-priority?     enumeration
     |        +--rw link-selection-mode
     |        |  +--rw mode?               enumeration
     |        |  +--rw physical-port?      uint32
     |        |  +--rw service-type?       enumeration
     |        |  +--rw service-provider?   string
     |        +--rw path-selection-mode?   enumeration
     +--rw traffic-sla
        +--rw traffic-sla* [custom_sla_name]
           +--rw custom_sla_name     string
           +--rw direction?          identityref
           +--rw latency?            uint32
           +--rw jitter?             uint32
           +--rw packet-loss-rate?   uint32
```

7. SD-WAN OSE Gateway Service YANG Module

```
 <CODE BEGINS> file "ietf-ose-gateway-svc@2019-06-10.yang"
 module ietf-ose-gateway-svc {
   namespace "urn:ietf:params:xml:ns:yang:ietf-ose-gateway-svc";
   prefix ose-gw-svc;

   organization
     "IETF foo Working Group.";
   contact
     "WG List: foo@ietf.org
      Editor:  ";
   description
     "The YANG module defines a generic service configuration
      model for interworking between different SD-WAN domains.";

   revision 2019-06-10 {
     description
       "Initial revision";
     reference
       "A YANG Data Model for SD-WAN service configuration of
        gateway-svc.";
   }

   feature ose-option-a {
     description
       "This feature means that ose reachability service option-A is
        supported by the Serving SDWAN manager";
     reference "ONUG-OSE-2 SDWAN Reachability and Segmentation
               Specification";
   }

   feature ose-option-b {
     description
       "This feature means that ose reachability service option-B
        is supported by the Serving SDWAN manager";
     reference "ONUG-OSE-2 SDWAN Reachability and Segmentation
               Specification";
   }

   container ose-gateways {
     list ose-gateway {
       key "gw-id";
       leaf gw-id {
         type uint32;
         description
           "Identifier for Gateway.";
       }
```

```
       leaf gw-name {
         type string;
         description
           "OSE gateway name.";
       }
       list peer-list {
         key "name";
         leaf name {
           type string;
           description
             "Peer Name.";
         }
         leaf peer-gw-id {
           type uint32;
           description
             "Identifier for the remote peer gateway.";
         }
         leaf peer-gw-name {
           type string;
           description
             "Name of remote peer gateway. ";
         }
         leaf ose-interworking-option {
           type enumeration {
             enum ose-option-a {
               description
                 "MP-BGP based Inter-AS VPN Option A with MPLS labels.";
             }
             enum ose-option-b {
               description
                 "MP-BGP based L3VPN Inter-AS Option B with MPLS
                  labels.";
             }
           }
           default "ose-option-b";
           description
             "OSE Gateway interworking options.";
         }
         leaf encap-type {
           type enumeration {
             enum ipsec_tunnel {
               description
                 "The encapsulation option is IPSec Tunnel mode per
                  RFC4303.";
             }
             enum ipsec_transport {
               description
                 "The encapsulation option is IPSec Transport mode
```

```
                   per RFC4303.";
             }
             enum gre {
               description
                 "The encapsulation option is GRE tunnel per.";
             }
           }
           description
             "The encapsulation options of OSE Gateway interworking.";
         }
         leaf auth-type {
           type enumeration {
             enum psk {
               description
                 "Pre-Shared Key(PSK).";
             }
             enum pki {
               description
                 "Public Key Infrastructure.";
             }
           }
           description
             "authentication type.";
         }
         leaf crypto-option {
           type enumeration {
             enum aes-128 {
               description
                 "crypto algorithm.";
             }
             enum aes-256 {
               description
                 "crypto algorithm.";
             }
             enum aes-256-gcm {
               description
                 "crypto algorithm.";
             }
           }
           description
             "Crypto algorithm selection. Others to be added.";
         }
         description
           "OSE Gateway peer gateway list.";
       }
       list segment-list {
         key "segment-name";
         leaf segment-name {
```

```
           type string;
           description
             "segment name.";
         }
         leaf vlan-id {
           if-feature "ose-option-a";
           type uint16;
           description
             "vlan ID.";
         }
         leaf vrf-id {
           if-feature "ose-option-b";
           type uint16;
           description
             "vrf ID.";
         }
         leaf segment-type {
           type enumeration {
             enum overlay {
               description
                 "overlay NNI interworking.";
             }
             enum nsw {
               description
                 "underlay NNI interworking.";
             }
           }
           description
             "segment type.";
         }
         list crossconnects {
           key "ccname";
           leaf ccname {
             type string;
             description
               "cross connection name.";
           }
           leaf gateway-reference {
             type leafref {
               path "../../../peer-list/peer-gw-id";
             }
             description
               "Specify the OSE gateway to be cross-connected
                with the segment.";
           }
           leaf peer-seg-name {
             type string;
             description
```

```
               "Peer segment name.";
           }
           leaf peer-seg-id-vlan {
             if-feature "ose-option-a";
             type uint16;
             description
               "Peer segment vlan ID.";
           }
           leaf peer-seg-id-vrf {
             if-feature "ose-option-b";
             type uint16;
             description
               "Peer Segment vrf ID.";
           }
           description
             "Cross connection List";
         }
         description
           "Segment List";
       }
       description
         "OSE gateway list.";
     }
     description
       "OSE gateway container.";
   }
 }

 <CODE ENDS>
```

8. SD-WAN OSE Path Service YANG Module

```
<CODE BEGINS> file "ietf-ose-path-svc@2019-06-10.yang"
module ietf-ose-path-svc {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ose-path-svc";
  prefix ose-path-svc;

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF foo Working Group.";
  contact
    "WG List: foo@ietf.org
     Editor:  ";
  description
    "The YANG module defines a generic service configuration
```

```
     model for interworking between different SD-WAN domains.";

  revision 2019-06-10 {
    description
      "Initial revision";
    reference
      "A YANG Data Model for SD-WAN service configuration of
       path-svc.";
  }

  identity traffic-direction {
    description
      "Base identity for traffic direction.";
  }

  identity upstream {
    base traffic-direction;
    description
      "Identity for Site-to-WAN direction.";
  }

  identity downstream {
    base traffic-direction;
    description
      "Identity for WAN-to-Site direction.";
  }

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

  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.";
  }

  container path-svc {
    description
      "Container for application aware path selection policy.";
    list path-policy {
      key "policy-name";
      description
        "List for path selection policy.";
      leaf policy-name {
        type string;
        description
          "Policy name.";
      }
      list flow-classification {
        key "class-name";
        description
          "List for traffic classification.";
        leaf class-name {
          type string;
          description
            "Traffic classification name.";
        }
        leaf dscp {
          type inet:dscp;
          description
            "DSCP value.";
        }
        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.";
        }
        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 protocol-field {
          type union {
            type uint8;
            type identityref {
              base protocol-type;
            }
          }
          description
            "Match on IPv4 protocol or IPv6 Next Header field.";
        }
      }
      list application-classification {
        key "application-class-name";
        description
          "List for application.";
        leaf application-class-name {
          type string;
          description
            "Application classification name.";
        }
        leaf category-id {
```

```
          type uint32;
          description
            "Describe the application category, e.g. Media,
             Peer2Peer.";
        }
        leaf application-id {
          type uint32;
          description
            "Describe the application and sub-application flows
              as well.";
        }
      }
      list user {
        key "list-name";
        description
          "List for User.";
        leaf list-name {
          type string;
          description
            "User list name.";
        }
        leaf-list user-id {
          type string;
          description
            "User list.";
        }
        leaf-list user-group {
          type string;
          description
            "User group list.";
        }
      }
      leaf-list site {
        type uint32;
        description
          "Describe the enterprise site or set of sites.";
      }
      container link-path-policy {
        description
          "Container for path selection policy.";
        leaf business-priority {
          type enumeration {
            enum high {
              description
                "Refers to high priority.";
            }
            enum normal {
              description
```

```
                "Refers to normal priority.";
            }
            enum low {
              description
                "Refers to low priority..";
            }
            enum voice {
              description
                "Refers to voice priority.";
            }
            enum critical_data {
              description
                "Refers to critical_data priority.";
            }
            enum transactional {
              description
                "Refers to transactional priority.";
            }
            enum user-defined {
              description
                "Refers to user-defined priority.";
            }
          }
          description
            "Describes the business priority for the matched traffic or
             application.";
        }
        container link-selection-mode {
          description
            "Describes the policy for how links should be selected for
            the specified traffic flow.";
          leaf mode {
            type enumeration {
              enum automatic {
                description
                  "Refers to automatic mode with all the WAN link
                  service.";
              }
              enum primary {
                description
                  "For certain traffic requiring high security or to
                  use a limited usage based circuit.";
              }
              enum lowest-cost {
                description
                  "For certain traffic only low cost WAN link could
                   be used.";
              }
```

```
            }
            description
              "Automatic option needs to take the SLA profile into
               consideration; Primary and lowest-cost are NOT
               automatic.";
          }
          leaf physical-port {
            type uint32;
            description
              "When in NOT automatic mode, specify the physical-port.";
          }
          leaf service-type {
            type enumeration {
              enum commodity {
                description
                  "Refers to broadband Internet links.";
              }
              enum wireless {
                description
                  "Refers to subset of 3G/4G/LTE and upcoming 5G.";
              }
              enum private {
                description
                  "Refers to private circuits such as Ethernet, T1,
                   etc.";
              }
            }
            description
              "When in NOT automatic mode, specify the physical-port,
               service-type.";
          }
          leaf service-provider {
            type string;
            description
              "When in NOT automatic mode, specify the name of
              provider.";
          }
        }
        leaf path-selection-mode {
          type enumeration {
            enum drop {
              description
                "Specify to drop the traffic.";
            }
            enum underlay {
              description
                "Specify the underlay path.";
            }
```

```
            enum overlay {
              description
                "Specify the overlay path.";
            }
          }
          default "overlay";
          description
            "Describes the policy for how paths should be selected for
             the specified traffic flow. If a destination for a traffic
             flow can be reached through both the overlay as well as
             the underlay,specify a preference .";
        }
      }
    }
  }
  container traffic-sla {
    description
      "Container for traffic SLA measurement.";
    list traffic-sla {
      key "custom_sla_name";
      description
        "List for traffic sla profile";
      leaf custom_sla_name {
        type string;
        description
          "customer traffic sla name";
      }
      leaf direction {
        type identityref {
          base traffic-direction;
        }
        default "both";
        description
          "The direction to which the QoS profile
           is applied: upstream or downstream.";
      }
      leaf latency {
        type uint32;
        units "msec";
        description
          "Downstream or upstream latency observed on the path in msec";
      }
      leaf jitter {
        type uint32;
        units "msec";
        description
          "Jitter observed on the path in msec";
      }
```

```
      leaf packet-loss-rate {
        type uint32 {
          range "0..100";
        }
        units "percent";
        description
          "Percentage of packet loss observed on the  path for the
           upstream and downstream";
      }
    }
  }
}

<CODE ENDS>
```

9. Security Considerations

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

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

o /ose-path/service

   The entries in the list above include the whole ose path service configurations which the customer subscribes, and indirectly create or modify the path selection configurations. Unexpected changes to these entries could lead to service disruption and/or network misbehavior.

o /ose-gateways/ose-gateway

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

- /ose-gateways/ose-gateway/peer-list

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

- /ose-gateways/ose-gateway/segment-list

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

10. 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-ose-path-svc
           Registrant Contact: The IESG
           XML: N/A; the requested URI is an XML namespace.

           URI: urn:ietf:params:xml:ns:yang:ietf-ose-gateway-svc
           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-ose-path-svc
          Namespace: urn:ietf:params:xml:ns:yang:ietf-ose-path-svc
          Prefix: path-svc
          Reference: RFC xxxx
          Name: ietf-ose-gateway-svc
          Namespace: urn:ietf:params:xml:ns:yang:ietf-ose-gateway-svc
          Prefix: reach-vpn
          Reference: RFC xxxx
---------------------------------------------------------------------
```

11. References

11.1. 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>.

[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>.

11.2. Informative References

[I-D.rosen-bess-secure-l3vpn] Rosen, E. and R. Bonica, "Augmenting RFC 4364 Technology to Provide Secure Layer L3VPNs over Public Infrastructure", draft-rosen-bess-secure-l3vpn-01 (work in progress), June 2018.

[ONUG] Group, O. S. W., Ed., "ONUG Software-Defned WAN Use Case: A white paper from the ONUG SD-WAN Working Group", October 2014.

[OSE] Group, O. O. W., Ed., "ONUG SOFTWARE DEFINED WAN (SD-WAN): NETWORK ARCHITECTURE FRAMEWORK".

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

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

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

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

[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>.

[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>.

[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>.

Appendix A. Acknowledges

Authors' Addresses

Steve Wood (editor)
Cisco Systems

Email: swood1@cisco.com

Bo Wu (editor)
Huawei Technologies, Co., Ltd

Email: lana.wubo@huawei.com

Qin Wu (editor)
Huawei Technologies, Co., Ltd

Email: bill.wu@huawei.com

Conrad Menezes
HPE Aruba

Email: conrad.menezes@hpe.com