OPSA Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2017

R. Chen
L. Zhang
H. Deng
Huawei Technologies
L. Geng
China Mobile
C. Xie
China Telecom

October 24, 2016

YANG Data Model for Composite VPN Service Delivery
draft-chen-opsawg-composite-vpn-dm-00

Abstract

The operator facing data model is valuable to reduce the operations and management. This document describes the YANG data model of the composite VPN for operators to provision, optimize, monitor and diagnose the end to end PE-based VPN services across multiple autonomous systems (ASes). The model from this document are limited to multiple ASes within one service provider.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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

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

This Internet-Draft will expire on April 27, 2017.

Table of Contents

1. Introduction

Internet Service Providers (ISPs) have significant interest on providing Provider Edge (PE) based virtual private network (VPN) services, in which the tunnel endpoints are the PE devices. In this case, the Customer Edge (CE) devices do not need to have any special VPN capabilities. Customers can reduce support costs by outsourcing VPN operations to ISPs and using the obtained connectivity.

Typically, customers require either layer 2 or layer 3 connectivity services to exchange traffic among a collection of sites. The ISP

gets the requirement and deploys the end to end VPN with an orchestrator across multiple autonomous systems (AS) within its administration.

The YANG data model[RFC7950] described in [I-D.ietf-l3sm-l3vpn-service-model] is used for communication between customers and network operators. It facilitates customers to request the layer 3 VPN service while concealing many provider parameters they do not need to know.

However, the network operators have a different view of the managed network. An operator facing data model is required to reduce the operations and management while still having full control on the network. So that the operators can provision, optimize, monitor and diagnose the VPN deployment based on the existing network infrastructure. Standardization of such a operator facing data model can help operators to operate and manage the VPN deployment in a standard way, and facilitate automation of optimization and diagnosis by standardizing the communication among multi-vendor services such as inventory, provisioning, monitoring, analysis and so on.

This document describes the generic VPN data model from the operators' view for the PE-based VPN service configuration. It aims at providing a simplified configuration on how the requested VPN service is to be deployed over the shared network infrastructure. This data model is limited to PE-Based VPNs as described in RFC 4110 [RFC4110] with the combination of layer 2 and layer 3 VPN services across multiple ASes within one ISP.

2. Definitions

- Customer Facing Service: The highly abstracted service which can be easily understood and consumed by the customer. With the emerging of cloud computing, customers require fast, easy to use, and on demand service from network operators.

- Operator Facing Service: The service provided for the operator to analyse, optimize, monitor and diagnose the network. It usually provides more detailed information related to the operations and management.

- Segmental VPN service: The VPN service deployed for one VPN segment within one AS.

- Composite VPN service: The VPN service deployed within the ISP administrative domain across one or more segments. It could be a combination of layer 2 and layer 3 VPN services for each segment.

3. Use Cases and Usage

In practice, ISP may have various scenarios for the end to end VPN service deployment depending on the network infrastructure and the customer sites connectivity requirements. It will consequently generate requirements of the generic Composite VPN service delivery model design. The Composite VPN data model described in this document covers the following scenarios:

- Multi-AS VPN Service: Customer sites are located in different autonomous systems(AS). ISP need to operate and manage the VPN service across multiple ASes. There are several different deployment scenarios in this case, e.g. single provider and multiple providers. This document only considers a single provider has multiple ASes which is less complex than the multi-provider and multi-AS case.

- Composite L2 and/or L3 VPN Service: Although the customer may request either layer 2 or layer 3 VPN service, the network infrastructure among customer sites may require different VPN service in the corresponding AS. So, an end to end VPN service within the ISP domain may be a composition of multiple segmental layer 2 and layer 3 VPN services. For example, a typical use case is that the enterprise need a layer 3 VPN connection to the remote data center. The end to end VPN connection may go across the metro access network and the IP core network. So the actual deployment of the customer’s request in ISP network maybe the Virtual Private Wire Service (VPLS) in the metro access network and the L3VPN in the IP WAN.

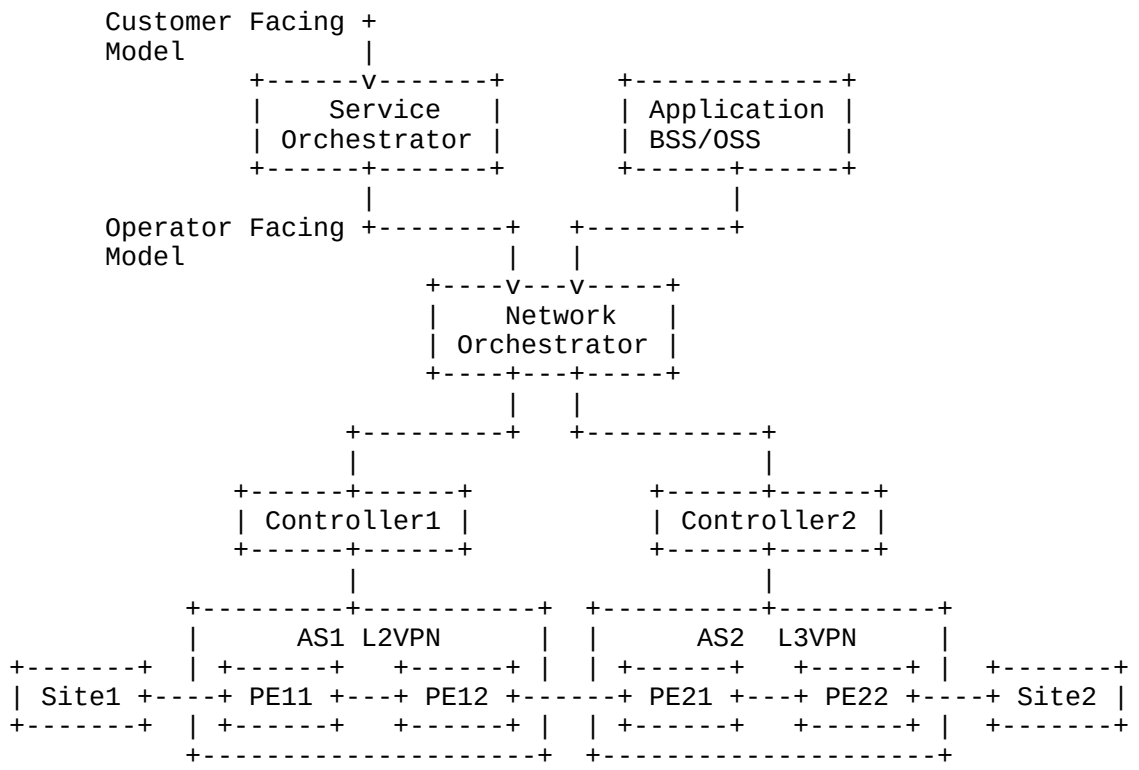
- Dynamic Site Insertion: The customer site that is not in the previously provisioned VPN can be quickly included.

A typical usage of this operator facing model is as a description of the end to end PE based VPN service for a network orchestration layer [I-D.wu-opsawg-service-model-explained] which can then be translated to Segmental VPN information for the configuration of domain controllers. As shown in the following figure, while, for example, users may send highly abstracted layer 3 VPN service requests to the service orchestrator, it cannot provide enough information for operators to operate and manage an end to end VPN service. The operator facing interface enables configuration of VPN deployment by introducing more network knowledge and governance policies. For example :

- Add the operational requirements for operation visualization, monitoring, and diagnosis. This requires more information on the Segmental VPN deployment in each AS.

- Optimize the VPN deployment of the customer's requests based on the exiting networking, e.g. deploy the L3VPN request from the customer to multiple VPN segments (IP Radio Access Network, Packet Transport Network, IP Core) in the end to end environment.

- Manage various policies for different customers.



4. Data Model Design Requirements

In order to describe the operator facing interface for the end to end PE based VPN service, the data model design should address the following requirements:

1. As the operator facing model, the model need to facilitate the operator to provision, optimize, monitor and diagnose the end to end PE-based VPN services across multiple autonomous systems. The model should be described from the operator's view of the VPN, not from the customer's view. For example, when the customer requests a VPN service, this can be expressed as a set of sites with the interconnected VPN. Both the sites and the VPN

model only consist of information that can be provided by the customer. For the same VPN service, the operator need to know information for operations and management, e.g. the access points on the PE, the VPN segment in each AS, and even the inter-connection between two ASes.

2. The model facilitates that the operator can quickly find all the information related to one end-to-end VPN of a particular customer. So that the operator can easily deal with one end to end VPN deployment. For example, the operator can trace how a VPN service request from the customer is really deployed (possibly across several ASes) in the network infrastructure. So when a connection failure happens, the operator can quickly determine where the problem is.

3. The model must be able to express various number of Segmental VPN composition. As described in Section 3, the operator need to operate and manage VPN among customer sites located in different autonomous systems(AS). In this case, an end to end VPN service deployed across multiple ASes, each of which can be described as a Segmental VPN. So the model should have the flexibility to describe both single AS and multi-AS VPN cases.

4. The model should include the basic information about the end to end VPN service. On one hand the operator can easily understand the deployment of one customer request. On the other hand, the overall information contains many parameters that can be referred and reused by many associated Segmental VPN models. This could be an abstract of the customer facing VPN model with information can be reused from operator’s view.

5. The model should allow to define one or multiple Segmental VPN information for each AS among the sites. As described in Section 3, an end to end VPN service within the ISP domain may consist of multiple segmental layer 2 and layer 3 VPN services. So the Segmental VPN description should have the capability to address the type of technology in use, and the corresponding parameters that will be used for the operations and management.

6. The model should facilitate operators to know the Access Point (AP) information for both Composite VPN and Segmental VPN. The AP of a Composite VPN is the interface between the provider network and the customer network. The AP of the Segmental VPN is the interface between two adjacent Segmental VPNs. The AP information is crucial because it has to match the remote peer for connectivity, and it usually contains information that is useful for the operations and management. For example, the AP

may indicate what's the peer connected, and the routing protocol that is used for exchanging routing information, and so on.

7. The VPN provisioning policy is optional but recommended in the model. 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, VLAN, and IP address. So when the customer requests a VPN, the system can automatically generate the end to end VPN planning according to the provisioning policies. To be simple, the model can only have an ID index refers to the VPN provisioning policy defined in other service applications.

8. The capability to describe the various QoS requirements should be supported by the model. There can be two kinds of QoS configuration.

   * The AP based QoS: describes the QoS requirements on the access point. For example, the CAR (committed access rate) definition on the inbound or outbound ports.

   * The flow based QoS: describes the QoS requirements on a flow. This enables the fine grained QoS control with the capability of identifying the flow.

5. Design of the Data Model

The PE-based VPN service is modeled with a recursive pattern. The VPN service deployed within each AS is modeled as a Segmental VPN object including the VPN description information within this AS and the Access Points (AP) that are used to connect to the peered device or AS. As an end to end VPN service within the ISP domain, it's then modeled as a Composite VPN object with the overall VPN information and the APs that are used to connect to the peered customer sites.

5.1. Composite VPN

The composedVPN top container contains the business type, VPN basic information, a list of segment VPN information, a list of access point information and and some overall information. The id MUST be unique for the reference of this composed VPN service. The tenantID associates this VPN service to a dedicated customer. The businessTypeID can associate composed VPN with a business template. The vpnBasicInfo object here includes basic information for this Composite VPN service. I.e., all the properties (e.g., topology, serviceType) in this object describe the overview that the customer want, no matter with any segment VPN information. The

accessPointList describes a list of APs that are used to connect to the peered customer sites. A Composite VPN includes one or more Segmental VPN.

```
+--rw composedVpns* [id]
   +--rw id                 yang:uuid
   +--rw name?              string
   +--rw description?       string
   +--rw tenantId?          yang:uuid
   +--rw businessTypeID?    yang:uuid
   +--rw vpnBasicInfo
   |  ...
   +--ro operStatus?        CommonTypes:OperStatus
   +--ro syncStatus?        CommonTypes:SyncStatus
   +--rw startTime?         yang:date-and-time
   +--rw segVpnList* [index]
   |  ...
   +--rw accessPointList* [id]
      ...
```

5.2. Access Point

The accessPointList models a list of APs including the access or attachment information for the remote peer. AP is provided by the PE either in the Composite VPN view or in the Segmental VPN view. The AP uses working layer and corresponding layer information to describe the different configurations in Composite VPN level and the Segmental VPN level.

```
+--rw accessPointList* [id]
   +--rw id                     yang:uuid
   +--rw name?                  string
   +--rw description?           string
   +--rw neID?                  yang:uuid
   +--rw containingMainTPID?    yang:uuid
   +--rw tpBasicInfo
   |  +--rw edgePointRole?   CommonTypes:EdgePointRole
   |  +--rw topologyRole?    CommonTypes:TopoNodeRole
   |  +--rw Type?            CommonTypes:TpType
   |  +--rw workingLayer?    CommonTypes:LayerRate
   |  +--rw typeSpecList* [layerRate]
   |  |  +--rw layerRate       CommonTypes:LayerRate
   |  |  +--rw (specValue)?
   |  |     +--:(LR_Ethernet)
   |  |     |  +--rw ethernetSpec
   |  |     |     +--rw accessType?    CommonTypes:EthernetEncapType
   |  |     |     +--rw (accessVlanValue)?
   |  |     |     |  +--:(QinQVlan)
```

```
|  |     |     |  |  +--rw qinqVlan
|  |     |     |  |     +--rw cvlanList*   uint64
|  |     |     |  |     +--rw svlanList?   uint64
|  |     |     |  +--:(DOT1Q)
|  |     |     |     +--rw dot1q
|  |     |     |        +--rw dot1qVlanList*   uint64
|  |     |     +--rw vlanAction?    CommonTypes:EthernetAction
|  |     |     +--rw actionValue?   string
|  |     +--:(LR_IP)
|  |     |  +--rw ipSpec
|  |     |     +--rw masterIp?   string
|  |     |     +--rw mtu?        uint64
|  |     +--:(LR_Vxlan)
|  |        +--rw vxlanSpec
|  |           +--rw vni?      uint32
|  |           +--rw vtepIP?   string
|  +--rw adminStatus?     CommonTypes:AdminStatus
|  +--rw tpQosNode
|  |  +--rw qosConfigType?     QosConfigType
|  |  +--rw qosDetailType?     QosDetailType
|  |  +--rw inTpCar* [index]
|  |  |  +--rw index         uint32
|  |  |  +--rw dataKind?     dataKind
|  |  |  +--rw actionType?   ActionType
|  |  |  +--rw action?       string
|  |  +--rw outTpCar* [index]
|  |  |  +--rw index         uint32
|  |  |  +--rw dataKind?     dataKind
|  |  |  +--rw actionType?   ActionType
|  |  |  +--rw action?       string
|  |  +--rw inQosProfileId?    yang:uuid
|  |  +--rw outQosProfileId?   yang:uuid
|  +--rw flowServices* [index]
|  |  +--rw qosConfigType?       QosConfigType
|  |  +--rw flowQosTemplateID?   yang:uuid
|  |  +--rw flowServices* [flowClassifierId]
|  |     +--rw flowClassifierId    yang:uuid
|  |     +--rw flowBehaviors* [index]
|  |        +--rw index         uint32
|  |        +--rw dataKind?     dataKind
|  |        +--rw actionType?   ActionType
|  |        +--rw action?       string
|  +--rw addtionalInfo* [name]
|     +--rw name     string
|     +--rw value?   string
+--rw peerCeTp
|  +--rw ceID?           yang:uuid
|  +--rw ceDirectNeID?   yang:uuid
```

```
|  +--rw ceDirectTPID?   yang:uuid
|  +--rw ceIfmasterIp?   string
|  +--rw location?       string
+--rw routeProtocolSpec* [type]
|  +--rw type                CommonTypes:RouteProtocolType
|  +--rw (para)?
|     +--:(staticRouting)
|     |  +--rw staticRouteItems* [index]
|     |     +--rw index              uint32
|     |     +--rw destinationCidr?   string
|     |     +--rw egressTP?          yang:uuid
|     |     +--rw routePreference?   string
|     |     +--rw nextHopIp?         string
|     +--:(bgp)
|        +--rw bgpProtocols* [index]
|           +--rw index                uint32
|           +--rw peerAsNumber?        uint64
|           +--rw bgpMaxPrefix?        int32
|           +--rw bgpMaxPrefixAlarm?   uint32
|           +--rw peerIp?              string
+--ro operStatus?          CommonTypes:OperStatus
```

5.2.1. Termination Point Basic Information

The tpBasicInfo describes the basic information that is used to express the design intent of the VPN from the Termination Point (TP) of view. That means the information described here is relative static, no matter which exact peer TP is going to connect.

The typeSpecList describes the layered information on the TP. It describes in detail on the ethernet layer information, or the IP layer and VxLan information if any higher layer protocol is enabled.

5.2.2. QoS

This model support two kinds of QoS description as described in the section 4:

- TP based QoS: describes the OoS requirements on a termination point. For example, the CAR (committed access rate) definition on the inbound or outbound ports.

- Flow based QoS: describes the QoS requirements on a flow. This enables the fine grained QoS control with the capability of identifying the flow.

5.2.3. Routing Protocol

The routeProtocolSpec object describes information of the routing protocol that is used to exchange the routing information with the remote peer. This object is extensible with any possible routing protocols. The BGP and static routing listed are examples to show how these two widely used solutions are described.

5.3. Segmental VPN

The segVpnList describes a list of Segmental VPN information which is only from the segment point of view. I.e., the description here takes care about how the Segmental VPN looks like and how it can communicate with peered devices outside this Segmental VPN. The segment information is composed of basic VPN information and a list of APs. The set of APs in the description are interfaces that customer sites or other segment VPNs can attach. In different scenarios, each Segmental VPN could be a layer 2 VPN, or layer 3 VPN, or even a termination point.

```
+--rw segVpnList* [index]
   +--rw index      uint32
   +--rw vpnType?   string
   +--rw vpnRole?   VPNTypes:ProtectionRole
   +--rw vpnInfo
      +--rw (vpnType)?
         +--:(wanVpn)
            +--rw vpn
               +--rw id?                yang:uuid
               +--rw name?              string
               +--rw description?       string
               +--rw vpnBasicInfo
               |  +--rw topology?      CommonTypes:Topology
               |  +--rw serviceType?   VPNTypes:ServiceType
               |  +--rw technology?    VPNTypes:VPNTunnelType
               |  +--rw adminStatus?   CommonTypes:AdminStatus
               +--ro operStatus?        CommonTypes:OperStatus
               +--ro syncStatus?        CommonTypes:SyncStatus
               +--rw accessPointList* [id]
                  ...
```

6. YANG Module

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

```
import ietf-yang-types {
    prefix yang;
}

import ietf-nvo-common-types {
    prefix CommonTypes;
}
import ietf-nvo-tp {
    prefix TP;
}

import ietf-nvo-vpn-types {
    prefix VPNTypes;
}

organization "" ;
contact "";
description "ietf-nvo-vpn";
revision 2016-10-24 {
    reference "draft-chen-opsawg-composite-vpn-dm-00";
}

container nvoVPNMgr{
    description "";
    list composedVPNs {
        key "id";
        description "";
        uses VPN:ComposedVPN;
    }
}

grouping ComposedVPN {
    description "ComposedVPN Grouping.";

    leaf id {
        type yang:uuid ;
        description "UUID-STR for service ." ;
    }

    leaf name {
        type string  {length "0..200";}
        description "Human-readable name for the service." ;
    }
    leaf description {
        type string  {length "0..200";}
        description "Detailed specification for the servcie." ;
    }
    leaf tenantId {
```

```
            type yang:uuid;
            description "UUID-STR for tenant." ;
        }
        leaf businessTypeID {
            type yang:uuid;
            description "business Type Name" ;
        }

        container vpnBasicInfo {
            description "VPN BASIC INFO";
            uses VPNTypes:VPNBasicInfo;
        }

        leaf operStatus {
            type CommonTypes:OperStatus;
            config false;
            description "Operational status." ;
        }

        leaf syncStatus {
            type CommonTypes:SyncStatus;
            config false;
            description "Sync status." ;
        }

        leaf startTime {
            type yang:date-and-time;
            description "Service lifecycle: request for service start
            time." ;
        }

        list segVpnList    {
            key "index";
            description "SegVpn list ";
            uses VPN:SegmentVPN;
        }

        list accessPointList {
            key "id";
            description "TP list of the access links which associated
            with CE and PE";
            uses TP:Tp;
        }
    }

    grouping SegmentVPN {
        description "SegmentVPN Grouping.";
```

```
            leaf index {
                type uint32;
                description "index of segment VPN in a composed VPN.";
            }

            leaf vpnType {
                type string  {length "0..30";}
                description "value: nop/wanVpn";
            }

            leaf vpnRole {
                type VPNTypes:ProtectionRole;
                description "value: nop|vpn";
            }

            container vpnInfo {
                description "vpn information";
                choice vpnType {
                      description "vpn type.";
                      case wanVpn {
                          container vpn {
                              description "vpn.";
                              uses VPN:VPN;
                            }
                      }
                  }
            }
        }

        grouping VPN {
            description "VPN Grouping.";

            leaf id {
                type yang:uuid ;
                description "UUID-STR for VPN." ;
            }
            leaf name {
                type string  {length "0..200";}
                description "Human-readable name for the service." ;
            }

            leaf description {
                type string  {length "0..200";}
                description "Detailed specification for the servcie." ;
            }

            container vpnBasicInfo {
                description "vpn basic info" ;
```

```
            uses VPNTypes:VPNBasicInfo;
        }

        leaf operStatus {
            type CommonTypes:OperStatus;
            config false;
            description "Operational status." ;
        }

        leaf syncStatus {
            type CommonTypes:SyncStatus;
            config false;
            description "Sync status." ;
        }

        list accessPointList {
            key "id";
            description "TP list of the access links which associated
            with CE and PE";
            uses TP:Tp;
        }
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-nvo-common-types.yang"
module ietf-nvo-common-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-common-types" ;
    prefix CommonTypes ;
    import ietf-yang-types {
        prefix yang;
    }

    organization "" ;
    contact "";
    description "ietf-nvo-common-types";
    revision 2016-10-24 {
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    typedef AdminStatus {
        type enumeration {
            enum active {
                description "Active status";
            }
            enum inactive {
                description "Inactive status";
            }
```

```
                enum partial {
                    description "Partial status";
                }
            }
            description "AdminStatus";
        }

        typedef OperStatus {
            type enumeration {
                enum up {
                    description "Up status";
                }
                enum down {
                    description "Down status";
                }
                enum degrade {
                    description "Degrade status";
                }
            }
            description "OperStatus";
        }

        typedef SyncStatus {
            type enumeration {
                enum SYNC {
                    description "Sync status";
                }
                enum OUT-SYNC {
                    description "Out sync status";
                }
            }
            description "SyncStatus";
        }

        typedef Topology {
            type enumeration {
                enum full-mesh {
                    description "full-mesh";
                }
                enum point_to_multipoint {
                    description "point_to_multipoint";
                }
                enum point_to_point {
                    description "point_to_point";
                }
                enum complex {
                    description "complex";
                }
```

```
            }
            description "Topology";
        }

        typedef Technology {
            type enumeration {
                enum mpls {
                    description "mpls";
                }
                enum rosen_multivpn {
                    description "rosen_multivpn";
                }
                enum ng_multivpn {
                    description "ng_multivpn";
                }
                enum vxlan_overlay_l3vpn {
                    description "vxlan_overlay_l3vpn";
                }
                enum eth_oversdh {
                    description "eth_oversdh";
                }
            }
            description "Technology";
        }

        typedef TopoNodeRole {
            type enumeration {
                enum other {
                    description "other";
                }
                enum hub {
                    description "hub";
                }
                enum spoke {
                    description "spoke";
                }
            }
            description "TopoNodeRole";
        }

        typedef TpType{
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum PTP {
                    description "PTP";
                }
```

```
                enum CTP {
                    description "CTP";
                }
                enum TRUNK {
                    description "TRUNK";
                }
                enum LoopBack {
                    description "LoopBack";
                }
                enum TPPool {
                    description "TPPool";
                }
            }
            description "TpType";
        }

        typedef LayerRate{
            type enumeration {
                enum LR_UNKNOW {
                    description "LR_UNKNOW";
                }
                enum LR_IP {
                    description "LR_IP";
                }
                enum LR_ETHERNET {
                    description "LR_ETHERNET";
                }
                enum LR_VXLAN {
                    description "LR_VXLAN";
                }
            }
            description "LayerRate";
        }

        typedef EthernetEncapType {
            type enumeration {
                enum DEFAULT {
                    description "DEFAULT";
                }
                enum DOT1Q {
                    description "DOT1Q";
                }
                enum QINQ {
                    description "QINQ";
                }
                enum UNTAG {
                    description "UNTAG";
                }
```

```
            }
            description "EthernetEncapType";
        }
        typedef EthernetAction {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum UNTAG {
                    description "UNTAG";
                }
                enum STACKING {
                    description "STACKING";
                }
            }
            description "EthernetAction";
        }

        typedef RouteProtocolType {
            type enumeration {
                enum staticRouting {
                    description "staticRouting";
                }
                enum bgp {
                    description "bgp";
                }
                enum rip {
                    description "rip";
                }
                enum ospf {
                    description "ospf";
                }
                enum isis {
                    description "isis";
                }
            }
            description "RouteProtocolType";
        }


        typedef QosPriorityType {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum 802dot1p {
                    description "802dot1p";
                }
```

```
                enum dscp {
                    description "dscp";
                }
                enum mplsExp {
                    description "mplsExp";
                }
                enum cos {
                    description "cos";
                }
                enum ipPrecedence {
                    description "ipPrecedence";
                }
            }
            description "QosPriorityType";
        }


        typedef ColorType {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum green {
                    description "green";
                }
                enum yellow {
                    description "yellow";
                }
                enum red {
                    description "red";
                }
            }
            description "ColorType";
        }


        grouping nvStringList{
            description "nvStringList Grouping.";

            list nvstringList {
                key "name";
                uses CommonTypes:nvstring;
                description "nvStringList";
            }
        }

        grouping nvstring {
            description "nvstring Grouping.";
```

```
            leaf name {
                type string;
                description "string name ";
            }
            leaf value {
                type string;
                description "string value";
            }
        }

        typedef FlowClassifierType {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum 802dot1p {
                    description "802dot1p";
                }
                enum dscp {
                    description "dscp";
                }
                enum cos {
                    description "cos";
                }
                enum mpls-exp {
                    description "mpls-exp";
                }
                enum sourceIP {
                    description "sourceIP";
                }
                enum destinationIP {
                    description "destinationIP";
                }
            }
            description "FlowClassifierType";
        }

        grouping ObjectIdentifiers {
            description "ObjectIdentifiers Grouping.";

            list ObjectIdentifiers {
                key "obejctId";
                uses CommonTypes:ObjectIdentifier;
                description "ObjectIdentifiers";
            }
        }

        grouping ObjectIdentifier {
```

```
            description "ObjectIdentifier Grouping.";

            leaf objectType {
                type CommonTypes:ObjectType;
                description "objectType";
            }

            leaf obejctId {
                type yang:uuid;
                description "obejctId";
            }

            leaf roleLable {
                type string {length "0..200";}
                description "here is the route role";
            }
        }

        typedef ObjectType {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum SEG-VPN {
                    description "SEG-VPN";
                }
                enum TP {
                    description "TP";
                }
                enum TPL {
                    description "TPL";
                }
                enum NE {
                    description "NE";
                }
                enum BUSINESSTYPE {
                    description "BUSINESSTYPE";
                }
                enum COMPOSED-VPN {
                    description "COMPOSED-VPN";
                }
                enum SUBNETWORK {
                    description "SUBNETWORK";
                }
            }
            description "ObjectType";
        }
```

```
        typedef EdgePointRole {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum PE {
                    description "PE";
                }
                enum P {
                    description "P";
                }
                enum UNI {
                    description "UNI";
                }
                enum NNI {
                    description "NNI";
                }
                enum AsbTP {
                    description "AsbTP";
                }
            }
            description "EdgePointRole";
        }

        typedef DomainRole {
            type enumeration {
                enum nop {
                    description "nop";
                }
                enum external {
                    description "external";
                }
                enum internal {
                    description "internal";
                }
                enum asb {
                    description "asb";
                }
            }
            description "DomainRole";
        }

        grouping CommandResult {
            description "this is the common result which is send back by
            server by RPC response";
            leaf resultCode {
                type int32;
                description "resultCode";
```

```
            }
            leaf-list successResourceList {
                type yang:uuid;
                description "successResourceList";
            }
            list failedResourceList {
                uses CommonTypes:FailedNode;
                description "failedResourceList";
            }
            leaf errorReason {
                type string {length "0..200";}
                description "errorReason";
            }
        }

        grouping FailedNode {
            description "fail reason for each object.";
            leaf resourceId {
                type yang:uuid;
                description "failed object.";
            }
            leaf errorReason {
                type string {length "0..200"; }
                description "errorReason";
            }
        }

        grouping SLA {
            description "SLA";
            leaf latency {
                type uint32;
                description "delay,unit:ms.";
            }
        }


        typedef ConnectionDirection {
            type enumeration {
                enum CD-UNI {
                    description "CD-UNI";
                }
                enum CD-BI {
                    description "CD-BI";
                }
            }
            description "ConnectionDirection";
        }
```

```
       typedef ObjectDirection {
           type enumeration {
               enum IN {
                   description "IN";
               }
               enum OUT {
                   description "OUT";
               }
               enum BI-DIRECTION {
                   description "BI-DIRECTION";
               }
           }
           description "ObjectDirection";
       }

       typedef DiversityType {
           type enumeration {
               enum NOP {
                   description "NOP";
               }
               enum PE-DIFF {
                   description "PE-DIFF";
               }
               enum MAIN-TP-DIFF {
                   description "MAIN-TP-DIFF";
               }
           }
           description "DiversityType";
       }
   }
   <CODE ENDS>

  <CODE BEGINS> file "ietf-nvo-qos-types.yang"
  module ietf-nvo-qos-types {
      namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-qos-types";
      prefix QosTypes;

      import ietf-yang-types { prefix yang; }

      organization "";
      contact "";
      description "ietf-nvo-qos-types";
      revision 2016-10-24 {
          reference "draft-chen-opsawg-composite-vpn-dm-00";
      }

      /*****************************************************************
      * Type definitions
```

```
      **************************************************************/
      typedef qosPriorityType {
          type enumeration {
              enum nop{
                  description "nop";
              }
              enum 802dot1p{
                  description "802dot1p";
              }
              enum dscp{
                  description "dscp";
              }
              enum mplsExp{
                  description "mplsExp";
              }
              enum cos{
                  description "cos";
              }
              enum ipPrecedence{
                  description "ipPrecedence";
              }
          }
          description "qosPriorityType";
      }

      typedef classifierDetailType {
          type enumeration {
              enum nop{
                  description "nop";
              }
              enum ipPrefixList{
                  description "ipPrefixList";
              }
          }
          description "classifierDetailType";
      }

      typedef ActionType {
          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 "ActionType";
    }

    typedef QosConfigType {
        type enumeration {
            enum nop{
                description "nop";
            }
            enum template{
                description "template";
            }
            enum agile{
                description "agile";
            }
        }
        description "QosConfigType";
    }

    typedef flowClassifierType {
        type enumeration {
            enum nop{
                description "nop";
            }
            enum aluFlowClassifier{
                description "aluFlowClassifier";
            }
        }
        description "flowClassifierType";
    }

    typedef QosDetailType {
        type enumeration {
```

```
            enum nop{
                description "nop";
            }
            enum car{
                description "car";
            }
            enum qosProfile{
                description "qosProfile";
            }
            enum diffServDomain{
                description "diffServDomain";
            }
            enum diffServ{
                description "diffServ";
            }
            enum aluDiffServ{
                description "aluDiffServ";
            }
        }
        description "QosDetailType";
    }

    typedef ClassifierType {
        type enumeration {
            enum 802dot1p{
                description "802dot1p";
            }
            enum dscp{
                description "dscp";
            }
            enum cos{
                description "cos";
            }
            enum mpls-exp{
                description "mpls-exp";
            }
            enum sourceIP{
                description "sourceIP";
            }
            enum destinationIP{
                description "destinationIP";
            }
        }
        description "ClassifierType";
    }

    typedef OrchPermitType {
        type enumeration {
```

```
            enum readUse{
                description "readUse";
            }
            enum crud{
                description "crud";
            }
        }
        description "OrchPermitType";
    }

    typedef ruleType {
        type enumeration {
            enum 802dot1p{
                description "802dot1p";
            }
            enum dscp{
                description "dscp";
            }
            enum cos{
                description "cos";
            }
            enum mpls_exp{
                description "mpls_exp";
            }
            enum source_ip{
                description "source_ip";
            }
            enum destination_ip{
                description "destination_ip";
            }
        }
        description "ruleType";
    }

    typedef MatchModeType {
        type enumeration {
            enum nop{
                description "nop";
            }
            enum match{
                description "match";
            }
            enum unmatch{
                description "unmatch";
            }
        }
        description "MatchModeType";
    }
```

```
      typedef qosBehaviorType {
          type enumeration {
              enum qosCarBehavior{
                  description "qosCarBehavior";
              }
              enum qosServiceChainBehavior{
                  description "qosServiceChainBehavior";
              }
          }
          description "qosBehaviorType";
      }

      typedef qosBehaviorDirectionType {
          type enumeration {
              enum upstream{
                  description "upstream";
              }
              enum downstream{
                  description "downstream";
              }
              enum bidirectional{
                  description "bidirectional";
              }
          }
          description "qosBehaviorDirectionType";
      }

      typedef dataKind {
          type enumeration {
              enum green{
                  description "green";
              }
              enum yellow{
                  description "yellow";
              }
              enum red{
                  description "red";
              }
              enum all{
                  description "all";
              }
          }
          description "dataKind";
      }

      typedef profileType {
          type enumeration {
              enum profile{
```

```
                description "profile";
            }
        }
        description "profileType";
    }

    typedef ruleOperatorType {
        type enumeration {
            enum and{
                description "and";
            }
            enum or{
                description "or";
            }
        }
        description "ruleOperatorType";
    }

    /*****************************************************************
    * Groupings
    *****************************************************************/
    grouping TPQosNode {
        description "TPQosNode Grouping.";

        leaf qosConfigType {
            type QosConfigType;
            description "qosConfigType";
        }
        leaf qosDetailType {
            type QosDetailType;
            description "qosDetailType";
        }
        list inTpCar {
            key index;
            uses FlowBehavior;
            description "inTpCar";
        }
        list outTpCar {
            key index;
            uses FlowBehavior;
            description "outTpCar";
         }
        leaf inQosProfileId {
            type yang:uuid;
            description "inQosProfileId";
        }
        leaf outQosProfileId {
            type yang:uuid;
```

```
            description "outQosProfileId";
        }
    }

    grouping FlowAndBehavior {
        description "FlowAndBehavior Grouping.";

        leaf flowClassifierId {
            type yang:uuid;
            description "flowClassifierId";
        }
        list flowBehaviors {
            key index;
            uses FlowBehavior;
            description "flowBehaviors";
        }
    }

    grouping FlowBehavior {
        description "FlowAndBehavior Grouping.";

        leaf index {
            type uint32;
            description "index";
        }
        leaf dataKind {
            type dataKind;
            description "dataKind";
        }
        leaf actionType {
            type ActionType;
            description "actionType";
        }
        leaf action {
            type string;
            description "action";
        }
    }

    grouping FlowClassifierRule {
        description "FlowClassifierRule Grouping.";

        leaf index {
            type uint32;
            description "index";
        }
        leaf matchMode {
            type MatchModeType;
```

```
            description "matchMode";
        }
        leaf type {
            type ClassifierType;
            description "type";
        }
        leaf-list flowClassifierValue {
            type string;
            description "flowClassifierValue";
        }
        leaf appendix {
            type string;
            description "appendix";
        }
    }

    grouping BandWidthNode {
        description "BandWidthNode Grouping.";

        leaf cir {
            type uint32;
            description "cir";
        }
        leaf pir {
            type uint32;
            description "pir";
        }
        leaf cbs {
            type uint32;
            description "cbs";
        }
        leaf pbs {
            type uint32;
            description "pbs";
        }
    }

    grouping FlowServices {
        description "FlowServices Grouping.";

        leaf qosConfigType {
            type QosConfigType;
            description "qosConfigType";
        }
        leaf flowQosTemplateID {
            type yang:uuid;
            description "flowQosTemplateID";
        }
```

```
          leaf qosDetailType {
              type QosTypes:QosDetailType;
              description "qosDetailType";
          }
          leaf inFlowQosTemplateID {
              type yang:uuid;
              description "inFlowQosTemplateID";
          }
          leaf outFlowQosTemplateID {
              type yang:uuid;
              description "outFlowQosTemplateID";
          }

          list flowServices {
              key flowClassifierId;
              description "default in flow and behaviors";
              uses FlowAndBehavior;
          }
      }
  }
  <CODE ENDS>

  <CODE BEGINS> file "ietf-nvo-tp.yang"
  module ietf-nvo-tp {
      namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-tp";
      prefix TP;

      import ietf-yang-types {
          prefix yang;
      }

      import ietf-nvo-common-types {
          prefix  CommonTypes;
      }

      import ietf-nvo-vpn-routeprotocol {
          prefix  RouteProtocol;
      }

      import ietf-nvo-tp-types {
          prefix TPTypes;
      }

      organization  "";
      contact   "";
      description "This module contains a collection of YANG definitions
      for tp";
      revision 2016-10-24 {
```

```
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    container nvoTPMgr{
        description "nvo tp management";
        list tps {
            key "id";
            uses TP:Tp;
            description "tp retrieve functions";
        }
    }

    grouping Tp {
        description "model of TP";
        leaf id {
            type yang:uuid;
            description "yang:uuid-str for TP";
        }
        leaf name {
            type string {length "0..200";}
            description "Must abbey to name rule defined in system.
            Example FE0/0/1, GE1/2/1.1, Eth-Trunk1.1, etc";
        }
        leaf description {
            type string {length "0..200";}
            description "description for this tp.";
        }

        leaf neID {
            type yang:uuid;
            description "yang:uuid-str for NE ";
        }

        leaf containingMainTPID {
            type yang:uuid;
            description "uuid-str for main interface";
        }
        container tpBasicInfo {
            description "Tp non-instance basic info";
            uses TPTypes:TPBasicInfo;
        }
        container peerCeTp {
            description "CE TP Information";
            uses TPTypes:CeTp;
        }

        list routeProtocolSpec {
            key "type";
```

```
            description "route protocol spec";
            uses RouteProtocol:RouteProtocolSpec;
        }

        leaf operStatus {
            type CommonTypes:OperStatus;
            config false;
            description "Operational status." ;
        }
    }

    grouping TpInventory {
        description "inventory model of TP";
        leaf tpID {
            type yang:uuid;
            description "uuid of tp";
        }
        leaf detailType {
            type string {length "0..100";}
            description "tp detail type. reported by controller";
        }
        leaf maxBandWidth {
            type uint64;
            description "max bandwidth";
        }
        leaf-list potentialLayers {
            type CommonTypes:LayerRate;
            description "capability of tp to create VPN, reported by
            controller";
        }
    }
}
<CODE ENDS>

 <CODE BEGINS> file "ietf-nvo-tp-types.yang"
 module ietf-nvo-tp-types {
     namespace "urn:ietf:params:xml:ns:yang:ietf-nvo-tp-types";
     prefix TPTypes;

     import ietf-nvo-common-types {prefix  CommonTypes;}
     import ietf-yang-types { prefix yang;     }
       import ietf-nvo-qos-types { prefix QosTypes;}

     organization "";
     contact "";
     description "ietf-nvo-tp-types";
     revision 2016-10-24 {
         reference "draft-chen-opsawg-composite-vpn-dm-00";
```

```
    }


    //LayerRate parameters.
    grouping PwSpec {
        description "PwSpec Grouping.";

        leaf controlWord {
            type boolean;
            default false;
            description "controlWord";
        }
        leaf pwVlanAction {
            type TPTypes:PWTagMode;
            description "pwVlanAction";
        }
    }

    grouping IpSpec {
        description "IpSpec Grouping.";

        leaf masterIp {
            type string;
            description "master IP address";
        }
        leaf mtu {
            type uint64;
            description "mtu for ip layer,scope:46~9600";
        }
    }

    grouping EthernetSpec {
        description "EthernetSpec Grouping.";

        leaf accessType {
            type CommonTypes:EthernetEncapType;
            description "access frame type";
        }

        choice accessVlanValue {
            description "accessVlanValue";
            case QinQVlan {
                container qinqVlan {
                    description "qinqVlan";
                    uses TPTypes:QinQVlan;
                }
            }
```

```
                case DOT1Q {
                    container dot1q {
                        description "dot1q";
                        uses TPTypes:Dot1QVlan;
                    }
                }
            }


            leaf vlanAction {
                type CommonTypes:EthernetAction;
                description "Frame type that can be accepted. not needed
                now";
            }

            leaf actionValue{
                type string{length "0..100";}
                description "action value";
            }
        }

        grouping QinQVlan {
            description "QinQVlan Grouping.";

            leaf-list cvlanList {
                type uint64;
                description "cvlanList";
            }
            leaf svlanList {
                type uint64;
                description "svlanList";
            }
        }

        grouping Dot1QVlan {
            description "Dot1QVlan Grouping.";

            leaf-list dot1qVlanList {
                type uint64;
                description "dot1qVlanList";
            }
        }

        grouping VxlanSpec {
            description "VxlanSpec Grouping.";

            leaf vni {
                type uint32;
```

```
            description "vni";
        }

        leaf vtepIP {
            type string;
            description "vtep ip";
        }
    }

    //CE spec
    grouping CeTp {
        description "CeTp Grouping.";

        leaf ceID {
            type yang:uuid;
            description "Site router ID";
        }

        leaf ceDirectNeID {
            type yang:uuid;
            description "direction connected NE ID, only valid in
            asbr ";
        }

        leaf ceDirectTPID {
            type yang:uuid;
            description "ce Direct TP id, only valid in asbr";
        }

        leaf ceIfmasterIp {
            type string;
            description "ceIfmasterIp";
        }

        leaf location {
            type string {length "0..400";}
            description "CE device location ";
        }
    }

    //TPBasicInfo
    grouping TPBasicInfo {
        description "TPBasicInfo Grouping.";

        leaf edgePointRole {
            type CommonTypes:EdgePointRole;
            description "edge role for TP, for example:UNI/NNI ";
        }
```

```
            leaf topologyRole {
                type CommonTypes:TopoNodeRole;
                description "hub/spoke role, etc";
            }

            leaf Type
            {
                type CommonTypes:TpType;
                description "Type";
            }

            leaf workingLayer {
                type CommonTypes:LayerRate;
                description "working layer";
            }

            list typeSpecList {
                key "layerRate";
                uses TPTypes:TpTypeSpec;
                description "typeSpecList";
            }

            leaf adminStatus {
                type CommonTypes:AdminStatus;
                description "administrative status.";
            }

            container tpQosNode {
                description "tpQosNode";
                uses QosTypes:TPQosNode;
            }

            container flowServices{
                description "flow services in one TP";
                uses QosTypes:FlowServices;
            }

            list addtionalInfo {
                key "name";
                uses CommonTypes:nvstring;
                description "addtionalInfo";
            }
        }

        //TpTypeSpec
        grouping TpTypeSpec{
            description "TpTypeSpec Grouping.";
```

```
            leaf layerRate {
                type CommonTypes:LayerRate;
                description "layerRate";
            }

            choice specValue {
                description "specValue";
                case LR_Ethernet {
                    container ethernetSpec {
                        description "ethernetSpec";
                        uses TPTypes:EthernetSpec;
                    }
                }
                case LR_IP {
                    container ipSpec {
                        description "ipSpec";
                        uses TPTypes:IpSpec;
                    }
                }
                case LR_Vxlan {
                    container vxlanSpec {
                        description "vxlanSpec";
                        uses TPTypes:VxlanSpec;
                    }
                }
            }
        }

        //-----------------------------------------------//
        //typedef
        //-----------------------------------------------//
        typedef PWTagMode {
            type enumeration {
                enum RAW{
                    description "RAW";
                }
                enum TAGGED{
                    description "TAGGED";
                }
            }
            description "PWTagMode";
        }
    }
    <CODE ENDS>

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

```
    prefix RouteProtocol;
    import ietf-yang-types { prefix yang; }
    import ietf-nvo-common-types {
        prefix  CommonTypes;
    }

    organization "";
    contact "";
    description "ietf-nvo-vpn-routeprotocol";
    revision 2016-10-24 {
        reference "draft-chen-opsawg-composite-vpn-dm-00";
    }

    grouping RouteProtocolSpec {
        description "RouteProtocolSpec Grouping.";

        leaf type {
            type CommonTypes:RouteProtocolType;
            description "Protocol type" ;
        }
        choice para {
            description "para" ;
            case staticRouting {
                list staticRouteItems {
                    key "index";
                    uses RouteProtocol:StaticRouteItem;
                    description "staticRouteItems" ;
                }
            }
            case bgp {
                    list bgpProtocols {
                            key "index";
                            uses RouteProtocol:BGPProtocolItem;
                            description "bgpProtocols" ;
                    }

            }
        }
    }

    grouping StaticRouteItem {
        description "StaticRouteItem Grouping.";

        leaf index {
            type uint32;
            description "static item index";
        }
        leaf destinationCidr {
```

```
            type string;
            description "destination ip cidr. ";
        }
        leaf egressTP {
            type yang:uuid;
            description "egress tp";
        }
        leaf routePreference {
            type string;
            description "route priority. Ordinary, work route have
            higher priority.";
        }
        leaf nextHopIp {
            type string {length "0..200";}
            description "nextHopIp";
        }
    }

    grouping BGPProtocolItem {
        description "BGPProtocolItem Grouping.";

        leaf index {
            type uint32;
            description "index of BGP protocol item";
        }
        leaf peerAsNumber {
            type uint64;
            description "";
        }
        leaf bgpMaxPrefix {
            type int32;
            description "";
        }
        leaf bgpMaxPrefixAlarm {
            type uint32;
            description "alarm threshold of BGP rout";
        }
        leaf peerIp {
            type string;
            description "peerIp";
        }
    }
}
<CODE ENDS>

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

```
prefix VPNTypes ;

import ietf-nvo-common-types {
    prefix  CommonTypes;
}

organization "";
contact "";
description "ietf-nvo-vpn-types";
revision 2016-10-24 {
    reference "draft-chen-opsawg-composite-vpn-dm-00";
}

typedef ProtectionRole {
    type enumeration {
        enum NOP{
            description "NOP";
        }
        enum MAIN{
            description "MAIN";
        }
    }
    description "ProtectionRole";
}

grouping VPNBasicInfo {
    description "VPNBasicInfo Grouping.";

    leaf topology {
        type CommonTypes:Topology;
        description "current support for full-mesh and
        point_to_multipoint(hub-spoke), others is reserved for
        future extensions." ;
    }

    leaf serviceType {
        type VPNTypes:ServiceType;
        description "current support for mpls l3vpn/vxlan/L2VPN
        overlay, others is reserved for future extensions." ;
    }

    leaf technology {
        type VPNTypes:VPNTunnelType;
        description "mpls|vxlan overlay l3vpn|eth over sdh|nop";
    }

    leaf adminStatus {
        type CommonTypes:AdminStatus;
```

```
                description "administrative status." ;
             }
        }

        typedef VPNTunnelType {
            type enumeration {
                enum NOP{
                    description "NOP";
                }
                enum MPLS{
                    description "MPLS";
                }
                enum MPLS-TP{
                    description "MPLS-TP";
                }
            }
            description "VPNTunnelType";
        }

        typedef ServiceType {
            type enumeration {
                enum l3vpn {
                    description "l3vpn" ;
                }
                enum l2vpn {
                    description "l2vpn" ;
                }
            }
            description "ServiceType";
        }
    }
    <CODE ENDS>
```

7. IANA Considerations

TBD

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <http://www.rfc-editor.org/info/rfc4110>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

10.2. Informative References

[I-D.ietf-l3sm-l3vpn-service-model] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model-18 (work in progress), October 2016.

[I-D.wu-opsawg-service-model-explained] Wu, Q., LIU, S., and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained-03 (work in progress), September 2016.

Authors' Addresses

Rui Chen
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
China

Email: chenrui@huawei.com

Liya Zhang
Huawei Technologies
Wuhan
China

Email: zhangliya@huawei.com

Hui Deng
Huawei Technologies
Beijing
China

Email: denghui02@hotmail.com

Liang Geng
China Mobile
No.32 Xuanwumen West Street
Beijing 100053
China

Email: liang.geng@hotmail.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@ctbri.com.cn