Abstract

This document defines a YANG model that is used to deliver layer 3 VPN service in ONOS project which is on the controller level. L3SM is focused on the service model which is on the orchestration level to help interaction between customers and network operators and also can be input to automated control and configuration applications. ONOS runs as an open source project for fast deployment. Use L3VPN service model defined in L3SM and deploy L3VPN service in ONOS project can be one step further. On the other hand, push real deployment in ONOS back to IETF is also beneficial.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on Expires June 16, 2016.
1. Introduction

ONOS, Open Networking Operating System, is an open source project that provides a platform for a vendor to build their controller through adding services/applications on top of ONOS. One of the principal motivations of ONOS is to facilitate service deployment, such as VPN service. [L3VPN] in L3SM WG defines a YANG model for layer 3 VPN service to deliver overall service. The description of the L3 service is more focused on the high level, orchestration level, with little information to guide the real configuration. A specific configuration related model that is especially working on controller level is necessary to fulfill the fast VPN provisioning.
This document introduces YANG [RFC6020] [RFC6021] data models for L3VPN configuration in ONOS. Such models can facilitate the standardization for the interface of ONOS, as they are compatible to a variety of protocols such as NETCONF [RFC6241] and [RESTCONF]. Please note that in the context of ONOS, the term "application" refers to an operational and management applications employed, and possibly implemented, by an operator.

2. Conventions used in this document

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]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. L3VPN Deployment in ONOS

In this section, a brief description of ONOS project as well as the L3 VPN deployment demo [ONOS-L3VPN] will be provided and relationship with L3SM will be addressed from an implementer's point of view.

3.1. ONOS-based SDN Controller

ONOS is an open source project that allows users to build their own SDN controller on top of it by adding service or applications. Without building everything from scratch, ONOS platform delivers fast and agile service provisioning. First of all, ONOS has abstracted southbound common API allows single controller to control the devices from different vendors. The southbound details with diversified devices and protocols have been masked so as to support multi-vendor environment. Also this enables controller to control both OF and legacy network devices. Secondly, ONOS is architected as logically centralized but physically distributed clustering system. This design aims to offload users’ development resource on performance and scalability. Lastly, Services provided by ONOS core are designed and implemented as a set of collaborating subsystems. Network control applications can use those services through the NB API.

3.2. L3VPN Deployment in ONOS-based SDN Controller

Deployment of L3VPN in ONOS is based on the requirement that fast and agile VPN provisioning. Using the ONOS given services as collaborating subsystems is the best way to avoid time consuming manual configurations. The overall architecture of L3 VPN deployment in ONOS is shown as figure 1.
More specifically, the deployment of L3VPN on ONOS platform is consisted of following implementations.

1. A user defined application implemented as an external application running on the top of ONOS. In this project, user provides L3VPN service, PCE (Path Calculation Engine) engine, interface to Portal software, etc.

2. The application keeps a snapshot of global view and network resource availability, which is used by PCE for path calculation in the project.

3. Use Producer-Consumer model to provision VPN. In this P-C model, Producer and Consumer are separate applications. Producer provisions network that for Consumer to use. In this project, Producer provisions MPLS tunnel. Consumer is L3VPN service that uses the tunnel.

4. ONOS core is enhanced with WAN service, such as MPLS Label Manager (support global and local label management), Tunnel Manager, and FlowRule extension.

5. South bound provider allows ONOS to talk with both legacy and OF devices and control both Legacy and OF environment/requirement.
4. L3VPN Data Model

This section is mainly focused on the L3VPN service model as mentioned in previous section about implementation.

4.1. Design of the Data Model

The L3SM is focused on the overall service to propose an abstracted interface to manage configuration of components of a L3VPN service. As shown in the document [L3VPN] section 4, it is service oriented and worked on the orchestration layer which does not specify network configuration. The data model in this document works on controller layer to guide the configuration by utilizing the given services in ONOS Core.

The L3VPN service model in [L3VPN] is constructed with two containers, one with some generic VPN information and the other one called site which is composed of a customer edge router (CE) attached to a provider edge router (PE). Meanwhile, the VPN service is sites based and can be treated as connections among multiple sites. However, the L3VPN service in this ONOS project is based on tunnel as mentioned earlier. The L3VPN data model here does not contradict with the one in [L3VPN] since the latter one is working the orchestration level and not designed for specific implementation. Using the [L3VPN] as a reference and use some of its parameters with specific configuration information to design the data model for this deployment in ONOS is primary goal of this document.

The YANG model in this document has two containers, one follows the vpn-svc in [L3VPN] to specify the parameter for L3VPN service and utilize the given service, and the latter one is to define the service path which contains ingress/egress information. Note that this configuration model utilizes only part of the parameters defined in the generic service model in [L3VPN] to deploy a VPN service. The key information such as "connection", "qos-policy", "connection", "protect-policy" and "oam-policy" all come from [L3VPN]. The top level design of the data model is shown in the following YANG tree.

4.1.1. VPN service topology

The type of topology of the VPN is required for configuration. Current version supports: hub and spoke (where hubs can exchange traffic), full-mesh. New topologies could be added by augmentation. The default service topology is "full-mesh".

4.1.2. L3 VPN access information

The access information of the VPN contains "role" information to denote role of the lsp that it is a master, slave, hub or spoke. Also the qos information is provided with qos policy. Another important container here is the "connection" to define the ip connection between PE and CE. Note that [L3VPN]
has "ip-connection" container defined in "sites", however the L3 VPN here is not site based so the connection information is defined as general parameters in access information. Following the structure in [L3VPN], the "connection" has ip address and the most important thing, "protocols" to connect PE and CE, such as OSPF, BGP or static.

4.1.3. Protect policy

Protect policy defines the protect type as well as revertive type.

4.1.4. Tunnel service

The "tunnel-service" is the most important container in vpn-svc. It tells the ONOS core to utilize the tunnel service with information on "tunnel-mode" and oam policy. Generally, this is very similar to the "sites" container in [L3VPN].
module: ietf-sd-onos-service-l3vpn
+-rw service
  +-rw l3vpn-cfg
    |-rw vpn-cfg* [name]
      |  +--rw name string
      |  +--rw id uint32
      |  +--rw user-label? string
      |  +--rw parent-ncd-id? string
      |  +--rw admin-status? types:admin-status
      |  +--rw operate-status? types:operate-status
      |  +--rw service-topology? enumeration
      |  +--rw access-information
      |     +--rw access-circuit* [id]
      |     |  +--rw id string
      |     |  +--rw name? string
      |     |  +--rw ne-id string
      |     |  +--rw ltp-id string
      |     |  +--rw admin-status? types:admin-status
      |     |  +--rw operate-status? types:operate-status
      |     +--rw 12-access
      |        +--rw access-type enumeration
      |        +--rw dot1q-vlan-bitmap string
      |        +--rw qinq-svlan-bitmap string
      |        +--rw qinq-cvlan-bitmap string
      |        +--rw access-action enumeration
      |        +--rw action-vlan-id? int32
      |        +--rw role? enumeration
      |        +--rw qos-policy
      |          +--rw qos-if-cars
      |          |  +--rw qos-if-car* [direction]
      |          |     +--rw direction enumeration
      |          |     +--rw cir? int32
      |          |     +--rw pir? int32
      |          |     +--rw cbs? int32
      |          |     +--rw pbs? int32
      |          +--rw connection
      |            +--rw ip-address? inet:ip-address
      |            +--rw mask-length? int32
      |            +--rw protocols? enumeration
      |            +--rw static-routes
      |            |  +--rw static-route* [ip-prefix mask-length]
      |            |     +--rw ip-prefix inet:ipv4-address
      |            |     +--rw mask-length uint32
      |            |     +--rw next-hop? inet:ipv4-address
      |            |     +--rw preference? uint32
      |            +--rw protect-policy
      |               +--rw protect-type? enumeration
Internet-Draft            L3VPN Deployment in ONOS               December 2015

+--rw revertive-type?   enumeration
+--rw wtr?              uint32
+-rw tunnel-service
  +--rw signaling-type?   enumeration
  +--rw tunnel-mode?      enumeration
  +--rw protect-policy
    +--rw protect-type?     enumeration
    +--rw revertive-type?   enumeration
    +--rw wtr?              uint32
  +--rw oam-policy
    +--rw detect-type?       enumeration
    +--rw bfd-detect-para
      +--rw ingress-discriminator?   int32
      +--rw egress-discriminator?   int32
      +--rw tx-interval?             int32
      +--rw rx-interval?             int32
      +--rw detect-interval?         int32
    +--rw latency?          int32
 +-rw service-paths
  +--rw service-path* [service-id source-ne-id source-ac-id destination-ne-id destination-ac-id]
    +--rw service-id           string
    +--rw source-ne-id         string
    +--rw source-ac-id         string
    +--rw destination-ne-id    string
    +--rw destination-ac-id    string
    +--rw path-lists
      +--rw path-list* [path-layer path-role]
      +--rw path-layer    enumeration
      +--rw path-role     enumeration
      +--rw paths
        +--rw path* [ne-id]
          +--rw ne-id               string
          +--rw ingress-ltp-id?     string
          +--rw backward-peer-id?   string
          +--rw egress-ltp-id?      string
          +--rw forward-peer-id?    string

rpcs:
  +---x create-l3vpn-instance
    +---w input
      +---w l3vpn-instance
        +---w name                 string
        +---w id                   uint32
        +---w user-label?          string
        +---w parent-ncd-id?       string
        +---w admin-status?        types:admin-status
        +---w operate-status?      types:operate-status
```plaintext
+---w service-topology?  enumeration
+---w access-information
  +---w access-circuit* [id]
    +---w id             string
    +---w name?          string
    +---w ne-id          string
    +---w ltp-id         string
    +---w admin-status?  types:admin-status
    +---w operate-status? types:operate-status
  +---w 12-access
    +---w access-type    enumeration
    +---w dot1q-vlan-bitmap string
    +---w qinq-svlan-bitmap string
    +---w qinq-cvlan-bitmap string
    +---w access-action  enumeration
    +---w action-vlan-id? int32
  +---w role?           enumeration
  +---w qos-policy
    +---w qos-if-cars
      +---w qos-if-car* [direction]
        +---w direction    enumeration
        +---w cir?         int32
        +---w pir?         int32
        +---w cbs?         int32
        +---w pbs?         int32
    +---w connection
      +---w ip-address?   inet:ip-address
      +---w mask-length?  int32
      +---w protocols?    enumeration
      +---w static-routes
        +---w static-route* [ip-prefix mask-length]
          +---w ip-prefix   inet:ipv4-address
          +---w mask-length uint32
          +---w next-hop?   inet:ipv4-address
          +---w preference? uint32
  +---w protect-policy
    +---w protect-type?  enumeration
    +---w revertive-type? enumeration
    +---w wtr?           uint32
  +---w tunnel-service
    +---w signaling-type? enumeration
    +---w tunnel-mode?   enumeration
    +---w protect-policy
      +---w protect-type?  enumeration
      +---w revertive-type? enumeration
      +---w wtr?           uint32
    +---w oam-policy
```

---


[Page 9]
+----w detect-type?    enumeration
   +----w bfd-detect-para
      +----w ingress-discriminator? int32
      +----w egress-discriminator? int32
      +----w tx-interval?        int32
      +----w rx-interval?        int32
      +----w detect-interval?    int32
   +----w latency?          int32
+----x delete-l3vpn-instance
   +----w input
      +----w l3vpn-id?    string
4.2. L3VPN YANG Model

<CODE BEGINS> file "ietf-sd-onos-common-types @2015-12-16.yang"
module ietf-sd-onos-common-types {
    prefix types ;
    /*
    import ietf-inet-types{
        prefix inet;
    }
    import ietf-yang-types {
        prefix yang-types;
    }
    */
    organization "";
    contact "";

    description
        "Defines common basic types of L3VPN.";

    revision "2015-10-14" {
        reference "";
    }

typedef admin-status {
    type enumeration {
        enum admin-up {
            value 0 ;
            description "admin up, the operate-status is depend on the real
running status." ;
        }
        enum admin-down {
            value 1 ;
            description "admin down, the operate-status is forced to down no
matter what the real status is" ;
        }
        enum config-up {
            value 2 ;
            description "the operate-status is forced to up no matter what
the real status is." ;
        }
    }
    default admin-up;
    description
        "The administration status of the service.";

typedef notification-status {
  type enumeration {
    enum up {
      value 0;
      description "up.";
    }
    enum down {
      value 1;
      description "down.";
    }
  }
  default up;
  description "The notification status of the service.";
}

typedef notification-type {
  type enumeration {
    enum ne {
      value 0;
      description "ncd change.";
    }
    enum link {
      value 1;
      description "link change.";
    }
    enum ltp {
      value 2;
      description "ltp change.";
    }
  }
  default ltp;
  description "The notification-type of the service.";
}

typedef operate-status {
  type enumeration {
    enum operate-up {
      value 0;
      description "operate up.";
    }
    enum operate-down {
      value 1;
      description "operate down.";
    }
  }
}
{  
  default operate-up;
  description
  "The operation status of the service."
}

grouping command-result {
  description
  "Reusable container of the result of the command.";
  container command-result {
    description
    "The result of the command.";
    leaf result {
      type int8;
      description
      "1 : success, 2 : failed, 3 : partly failed" ;
    }
    container success-resources {
      description
      "The resources those are available." ;
      list success-resource-list {
        description
        "The resource list shows those are available." ;
        leaf resource-id {
          type string;
          description
          "The available resource id." ;
        }
      }
    }
    container failed-resources {
      description
      "The resources those are failed." ;
      list failed-resource-list {
        description
        "The resources list shows those are failed." ;
        leaf resource-id {
          type string;
          description
          "The failed resources ids." ;
        }
        leaf error-code {
          type string;
          description
          "The error code." ;
        }
      }
    }
  }
}

typedef exp {
  type enumeration {
    enum BE {
      value 0 ;
      description "BE." ;
    }
    enum AF1 {
      value 1 ;
      description "AF1." ;
    }
    enum AF2 {
      value 2 ;
      description "AF2." ;
    }
    enum AF3 {
      value 3 ;
      description "AF3." ;
    }
    enum AF4 {
      value 4 ;
      description "AF4." ;
    }
  }
}
enum EF {
  value 5;
  description "EF.";
}
enum CS6 {
  value 6;
  description "CS6.";
}
enum CS7 {
  value 7;
  description "CS7.";
}
default CS7;
description "exp parameter.";

typedef pw-role{
type enumeration {
  enum normal{
    value 0;
    description "normal.";
  }
  enum master{
    value 1;
    description "master.";
  }
  enum slave{
    value 2;
    description "slave.";
  }
  enum DNI-PW{
    value 3;
    description "DNI PW.";
  }
}
default normal;
description "The role of the PW.";

grouping qos-if-car{
description "qos parameter.";
leaf direction{
type enumeration {
enum inbound{
    value 0 ;
    description "inbound." ;
}
enum outbound {
    value 1 ;
    description "outbound." ;
}
description "qos for interface car" ;
}
leaf cir {
    type int32;
    description "forward CIR. unit:Kbps" ;
}
leaf pir {
    type int32;
    description "forward PIR. unit:Kbps" ;
}
leaf cbs {
    type int32;
    description "forward CBS. unit:Kbps" ;
}
leaf pbs {
    type int32;
    description "forward PBS. unit:Kbps" ;
}
}
grouping protect-policy {
    description "The protect policy of the VPN" ;
    leaf protect-type {
        type enumeration {
            enum unprotected {
                value 0 ;
                description "unprotected." ;
            }
            enum protected {
                value 1 ;
                description "protection." ;
            }
        }
        default unprotected ;
        description "protection type" ;
    }
}
leaf revertive-type {
  type enumeration {
    enum no-revertive {
      value 0 ;
      description "unprotected." ;
    }
    enum revertive {
      value 1 ;
      description "protection." ;
    }
  }
  description "revertive mode" ;
}
leaf wtr {
  type uint32;
  default 300;
  description "WTR.unit:s" ;
}
}

grouping oam-policy {
  description "The oam policy of the vpn service." ;
  leaf detect-type {
    type enumeration {
      enum undetect {
        value 0 ;
        description "unprotected." ;
      }
      enum APS {
        value 1 ;
        description "protection." ;
      }
      enum BFD {
        value 2 ;
        description "unprotected." ;
      }
    }
    description "detect type" ;
  }
  container bfd-detect-para {
    description "bfd detect parameters." ;
    leaf ingress-discriminator {
      type int32;
      description "ingress-discriminator" ;
    }
    leaf egress-discriminator {
      type int32;
      description "egress-discriminator" ;
    }
  }
}
leaf tx-interval {
  type int32;
  description "tx-interval" ;
}
leaf rx-interval {
  type int32;
  description "rx-interval" ;
}
leaf detect-interval {
  type int32;
  description "detect-interval" ;
}

grouping ac {
  description "Access information." ;
  leaf id{
    type string;
    mandatory true;
    description "ac id." ;
  }
  leaf name{
    type string;
    config false;
    description "ac name." ;
  }
  leaf ne-id {
    type string ;
    mandatory true;
    description "ne id." ;
  }
  leaf ltp-id {
    type string ;
    mandatory true;
    description "ltp id." ;
  }
  leaf admin-status {
    type types:admin-status;
    description "Administration status." ;
  }
  leaf operate-status {
    type types:operate-status;
    description "Operation status." ;
  }
}
container l2-access {
    description "link layer access information of ac.";
    uses l2-access;
}

leaf role {
    type enumeration {
        enum master {
            value 0;
            description "master.";
        }
        enum slave {
            value 1;
            description "slave.";
        }
        enum hub {
            value 2;
            description "slave.";
        }
        enum spoke {
            value 3;
            description "slave.";
        }
    }
    default master;
    description "role of snc lsp.";
}

container qos-policy {
    description "The qos policy of the vpn service.";
    container qos-if-cars {
        description "qos policy if car.";
        list qos-if-car {
            key "direction";
            uses qos-if-car;
            description "List of qos parameters.";
        }
    }
}

grouping l3-ac {
    description "Access information of l3vpn.";
    uses ac;
    container connection {
        description "connection information of ac.";
        uses connection;
    }
}
grouping l2-access {
  description "Access information of l2vpn.";
  leaf access-type{
    type enumeration {
      enum Port {
        value 0;
        description "master.";
      }
      enum Dot1Q {
        value 1;
        description "slave.";
      }
      enum QinQ {
        value 2;
        description "master.";
      }
    }
    mandatory true;
    description "The access type of the vpn service.";
  }
  leaf dot1q-vlan-bitmap {
    type string;
    mandatory true;
    description "Dot1Q Vlan Bitmap.";
  }
  leaf qinq-svlan-bitmap {
    type string;
    mandatory true;
    description "QinQ svlan Bitmap.";
  }
  leaf qinq-cvlan-bitmap {
    type string;
    mandatory true;
    description "QinQ cvlan Bitmap.";
  }
  leaf access-action {
    type enumeration {
      enum Keep {
        value 0;
        description "keep.";
      }
      enum Push {
        value 1;
        description "push.";
      }
    }
  }
}
enum Pop {
  value 2;
  description "pop.";
}
enum Swap {
  value 3;
  description "swap.";
}

mandatory true;
  description "access type.";
}
leaf action-vlan-id {
  type int32 {
    range "1..4094";
  }
  description "action vlan id.";
}

grouping connection {
  description "Describe the connection of the vpn service.";
  leaf ip-address {
    type inet:ip-address;
    description "ip address of access circuit’s connection.";
  }
  leaf mask-length {
    type int32 {
      range "1..32";
    }
    description "mask length of ip address.";
  }
  leaf protocols {
    type enumeration {
      enum static {
        value 0;
        description "static.";
      }
      enum ospf {
        value 1;
        description "os pf.";
      }
      enum isis {
        value 2;
        description "bgp";
      }
      enum bgp {

value 3;
    description "bgp";
  }
}
description "protocols between PE and CE.";
}
container static-routes {
  description "Defines the static routes.";
  list static-route {
    key "ip-prefix mask-length";
    description "List of static route.";
    leaf ip-prefix {
      type inet:ipv4-address;
      description "ip prefix";
    }
    leaf mask-length {
      type uint32 {
        range "1..32";
      }
      description "mask length";
    }
    leaf next-hop {
      type inet:ipv4-address;
      description "next hop";
    }
    leaf preference {
      type uint32 {
        range "1..65535";
      }
      description "preference of the route.";
    }
  }
}
}
grouping pw{
  description "PW definition ";
  leaf id {
    type string;
    description "ID of pw.";
  }
  uses encapslate-type-grouping;
  leaf ingress-ne-id {
    type string;
    description "ingress ne id.";
  }
}
leaf egress-ne-id {
  type string;
  description "egress ne id.";
}

leaf ctrl-word-support {
  type enumeration {
    enum not-support {
      value 0;
      description "pw doesn't support control word";
    }
    enum support {
      value 1;
      description "pw supports control word";
    }
  }
  default support;
  description "ctrl word support. 0 : not support, 1 : support";
}

leaf sn-support {
  type enumeration {
    enum not-support {
      value 0;
      description "pw doesn't support control word";
    }
    enum support {
      value 1;
      description "pw supports control word";
    }
  }
  default not-support;
  description "serial number support. 0 : not support, 1 : support";
}

leaf vccv-type {
  type enumeration {
    enum not-support {
      value 0;
      description "pw doesn't support vccv";
    }
    enum CWD {
      value 1;
      description "pw supports vccv";
    }
    enum Label-alert {

enum TTL {
    value 2 ;
    description "pw supports vccv" ;
}

enum CWD&Label-alert {
    value 4 ;
    description "pw supports vccv" ;
}

enum CWD&TTL {
    value 5 ;
    description "pw supports vccv" ;
}

enum Label-alert&TTL {
    value 6 ;
    description "pw supports vccv" ;
}

enum CWD&Label-alert&TTL {
    value 7 ;
    description "pw supports vccv" ;
}

default not-support;
description "vccv type" ;

leaf conn-ack-type {
    type enumeration {
        enum not-support {
            value 0 ;
            description "pw doesn’t support connection ack" ;
        }
        enum support {
            value 1 ;
            description "pw supports connection ack" ;
        }
    }

default not-support;
description "Connectivity test type" ;
}

container tunnels {
    description "Define tunnels." ;
    list tunnel{
        key "tunnel-id";
        description "The list of tunnel id." ;
    }
}
uses tunnel;
}
}
}

grouping tunnel {
    description "Reusable entity of tunnel.";
    leaf tunnel-id {
        type string;
        description "ID of tunnel.";
    }
}

grouping encaplate-type-grouping {
    description "encaplate type";
    leaf encaplate-type {
        type enumeration {
            enum NONE {
                value 0;
                description "none.";
            }
            enum fr-dlci-martini {
                value 1;
                description "fr-dlci-martini.";
            }
            enum atm-aal5-sdu {
                value 2;
                description "atm-aal5-sdu.";
            }
            enum atm-transparent {
                value 3;
                description "atm-transparent.";
            }
            enum ethernet-vlan {
                value 4;
                description "ethernet-vlan.";
            }
            enum ethernet {
                value 5;
                description "ethernet.";
            }
            enum hdlc {
                value 6;
                description "hdlc.";
            }
            enum ppp {
                value 7;
                description "ppp.";
            }
        }
    }
}
enum cep-mpls {
    value 8;
    description "cep-mpls.";
}
enum atm-ntol {
    value 9;
    description "atm-ntol.";
}
enum atm-ntol-vpc {
    value 10;
    description "atm-ntol-vpc.";
}
enum ip-layer2 {
    value 11;
    description "ip-layer2.";
}
enum atm-1tol-vcc {
    value 12;
    description "atm-1tol-vcc.";
}
enum atm-1tol-vpc {
    value 13;
    description "atm-1tol-vpc.";
}
enum atm-aal5-pdu {
    value 14;
    description "atm-aal5-pdu.";
}
enum fr-port {
    value 15;
    description "fr-port.";
}
enum cep-packet {
    value 16;
    description "cep-packet.";
}
enum e1 {
    value 17;
    description "e1.";
}
enum t1 {
    value 18;
    description "t1.";
}
enum e3 {
    value 19;
    description "e3.";}
enum t3 {
    value 20;
    description "t3.";
}

enum cesopsn-basic {
    value 21;
    description "cesopsn-basic.";
}

enum tdmoip-aal1 {
    value 22;
    description "tdmoip-aal1.";
}

enum cesopsn-tdm {
    value 23;
    description "cesopsn-tdm.";
}

enum tdmoip-aal2 {
    value 24;
    description "tdmoip-aal2.";
}

enum fr-dlci {
    value 25;
    description "fr-dlci.";
}

description "encaplate type.";
}

grouping pw-trail{
    description "pw trail information.";
    leaf id {
        type string;
        description "ID of pw-trail.";
    }

    leaf role {
        type pw-role;
        description "role of pw-trail.";
    }

    container pw-lists {
        description "List of pw information.";
        list pw-list {
            key id;
            description "List of pw information.";
            uses pw;
grouping tunnel-service {
    description "Reusable entity of tunnel service.";
    leaf signaling-type {
        type enumeration {
            enum RSVP-TE {
                value 0;
                description "RSVP-TE";
            }
            enum LDP {
                value 1;
                description "LDP";
            }
            enum GRE {
                value 2;
                description "GRE";
            }
            enum SR-BE {
                value 3;
                description "SR-BE";
            }
            enum SR-TE {
                value 4;
                description "SR-TE";
            }
        } default RSVP-TE;
        description "signaling type.";
    }
    leaf tunnel-mode {
        type enumeration {
            enum Nto1 {
                value 0;
                description "multi service one tunnel";
            }
            enum 1to1 {
                value 1;
                description "oner service one tunnel";
            }
        } default Nto1;
        description "service to tunnel mode.";
    }
    container protect-policy {
        description "Protect policy.";
    }
}
uses protect-policy;
}
}
}

container oam-policy {
  description "oam policy.";
  uses oam-policy;
}

leaf latency {
  type int32;
  description "tunnel latency requirement.";
}

}

grouping service-path {
  description "Service path of l3vpn.";
  leaf service-id {
    type string;
    description "l2vpn or l3vpn service id.";
  }
  leaf source-ne-id {
    type string;
    description "source ne id.";
  }
  leaf source-ac-id {
    type string;
    description "source ltp id.";
  }
  leaf destination-ne-id {
    type string;
    description "destination ne id.";
  }
  leaf destination-ac-id {
    type string;
    description "destination ltp id.";
  }
  container path-lists{
    description "The path list of service path.";
    list path-list {
      key "path-layer path-role";
      description "The path list of service path.";
      leaf path-layer {
        type enumeration {
          enum PW {
            value 0;
            description "pw path.";
          }
          enum BGP-LSP {
        }
      }
    }
  }
value 1 ;
  description "BGP-LSP Path." ;
}
enum LSP {
  value 2 ;
  description "LSP Path." ;
}

description "path type. 0 : PW, 1 : BGP-LSP, 2 : PW" ;
}
leaf path-role {
  type enumeration {
    enum Primary {
      value 0 ;
      description "master path." ;
    }
    enum Backup {
      value 1 ;
      description "backup path." ;
    }
    enum Active {
      value 2 ;
      description "active path." ;
    }
  }
  description "path role. 0 : master, 1 : backup, 2 : Bypass." ;
}
container paths {
  description "path definition." ;
  list path {
    key "ne-id";
    description "Network element id list." ;
    leaf ne-id {
      type string;
      description "Network element id." ;
    }
    leaf ingress-ltp-id {
      type string;
      description "ingress ltd id." ;
    }
    leaf backward-peer-id {
      type string;
      description "backward peer id." ;
    }
    leaf egress-ltp-id {
      type string;
    }
  }
}
description "egress ltd id." ;
}
leaf forward-peer-id {
  type string;
  description "forward peer id." ;
}

</CODE ENDS>

<CODE BEGINS> file "ietf-sd-onos-service-l3vpn @2015-12-16.yang"
module ietf-sd-onos-service-l3vpn {
  prefix l3vpn ;
  /*
   import ietf-inet-types{
     prefix inet;
   }
   import ietf-yang-types {
     prefix yang-types;
   }
   */
  import ietf-sd-onos-service-types {
    prefix service-types;
  }
  import ietf-sd-onos-common-types {
    prefix types;
  }

  organization "";
  contact "";

  description
    "L3vpn configuration model in ONOS.";

  revision "2015-12-16" {
    reference "";
  }

  container service {
    description
      "Root level of L3vpn service module.";
    container l3vpn-cfg {

description
"l3vpn configuration model in ONOS.";
list vpn-cfg {
  key name;
  description
  "vpn configuration parameter list.";
  uses l3vpn;
}
}
container service-paths {
  description
  "The service path of the l3 vpn.";
  list service-path {
    key "service-id source-ne-id source-ac-id destination-ne-id destination-ac-id";
    description
    "The list of service path.";
    uses service-types:service-path;
  }
}
}

grouping l3vpn {
  description
  "The configuration module of l3 vpn.";
  leaf name {
    type string ;
    mandatory true;
    description "name of snc eline." ;
  }
  leaf id {
    type uint32 ;
    mandatory true;
    description "ID of snc eline." ;
  }
  leaf user-label {
    type string ;
    description "user label of snc eline." ;
  }
  leaf parent-ncd-id {
    type string ;
    description "parent ncd id." ;
  }
  leaf admin-status {
    type types:admin-status;
    description "administration status." ;
  }
}
leaf operate-status {
    type types:operate-status;
    description "operation status." ;
}

uses service-type-grouping;
container acess-information {
    description "access information of the l3 vpn." ;
    list access-circuit {
        key "id";
        description "list of access circuit." ;
        uses service-types:l3-ac;
    }
}

container protect-policy{
    description "L3VPN Service protect policy." ;
    uses service-types:protect-policy;
}
container tunnel-service {
    description "tunnel service." ;
    uses service-types:tunnel-service;
}

grouping service-type-grouping {
    description "Basic service type" ;
    leaf service-topology {
        type enumeration {
            enum full-mesh {
                value 1 ;
                description "full-mesh." ;
            }
            enum hub-spoke {
                value 2 ;
                description "hub-spoke." ;
            }
        }
        default full-mesh;
        description "service topology type." ;
    }
}

rpc create-l3vpn-instance {
    description "Create l3vpn instance." ;
}
input {
  container l3vpn-instance {
    description "Create l3vpn instance.";
    uses l3vpn;
  }
}
rpc delete-l3vpn-instance {
  description "Delete l3vpn instance.";
  input {
    leaf l3vpn-id {
      type string;
      description "vpn id.";
    }
  }
}
rpc close-l3vpn {
  input {
    leaf l3vpn-id {
      type string;
      description "vpn id.";
    }
    container ac-status {
      description "Access status of the vpn.";
      list acs {
        key "id";
        description "Access information.";
        leaf id {
          type string;
          description "Access id.";
        }
        leaf admin-status {
          type types:admin-status;
          description "Administration status.";
        }
      }
    }
  }
}
rpc modify-l3vpn-instance-basic {
  input {
    leaf l3vpn-id {
      type string;
      description "vpn id.";
    }
  }
}
leaf user-label {
    type string;
}

rpc modify-l3vpn-instance-ac-qos {
    input {
        leaf l3vpn-id {
            type string;
        }
        container ac-qos {
            list acs{
                key "id";
                leaf id {
                    type string;
                }
                container qos-policy {
                    container qos-if-cars {
                        list qos-if-car {
                            key "direction";
                            uses service-types:qos-if-car;
                        }
                    }
                }
            }
        }
    }
}

rpc modify-l3vpn-instance-connection {
    input {
        leaf l3vpn-id {
            type string;
        }
        container ac-connection {
            list acs{
                leaf id {
                    description "ac id.";
                    type string;
                }
                container connection {
                    description "CE to PE connection.";
                    uses service-types:connection;
                }
            }
        }
    }
}
rpc inquire-l3vpn-instance-work-path {
  input {
    leaf service-id {
      type string;
    }
    leaf ingress-ne-id {
      type string;
    }
    leaf destination-ne-id {
      type string;
    }
    leaf ingress-ac-id {
      type string;
    }
    leaf destination-ac-id {
      type string;
    }
    leaf path-layer {
      type string;
    }
    leaf path-role {
      type string;
    }
  }
  output {
    container service-path {
      leaf service-id {
        type string;
      }
      leaf ingress-ne-id {
        type string;
      }
      leaf destination-ne-id {
        type string;
      }
      leaf ingress-ac-id {
        type string;
      }
      leaf destination-ac-id {
        type string;
      }
      leaf path-layer {
        type string;
      }
    }
  }
}
leaf path-role {
  type string;
}
list path-list {
  key "ne-id";
  leaf ne-id {
    type string;
  }
  leaf ingress-ltp-id {
    type string;
  }
  leaf backward-peer-id {
    type string;
  }
  leaf egress-ltp-id {
    type string;
  }
  leaf forward-peer-id {
    type string;
  }
}

5. Security Considerations

TBD

6. IANA Considerations

This document has no actions for IANA.

7. Contributors and Acknowledgments

Jan Lindblad
Tail-F
Email: janl@tail-f.com
8. References

8.1. Normative References


8.2. Informative References


Authors’ Addresses

Yiyong Zha
Huawei Technologies
zhayiyong@huawei.com

Dacheng Zhang
Chaoyang Dist
Beijing 100000
P.R. China
Dacheng.zhang@gmail.com

Ying Cheng (Editor)
China Unicom
P.R. China
Email: chengying10@chinaunicom.cn

Maxim Klyus
NetCracker
Email: klyus@NetCracker.com