Abstract

This document defines a YANG data model that can be used to deliver a Layer 2 Provider Provisioned VPN service. These services include Virtual Private Wire Service (VPWS) and Virtual Private LAN service (VPLS). This model is intended to be instantiated at management system to deliver the L2VPN service, and is not a configuration model to be used directly on network elements.

This model provides an abstracted view of the Layer 2 VPN service configuration components. It will be up to a management system (orchestrator) to take this as an input and use specific configurations models to configure the different network elements to deliver the service. It is called as north bound L2VPN Service YANG data model. How configuration of network elements is out of scope of the document, and is defined in document [I-D.shah-bess-l2vpn-yang].

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 February 4, 2017.
1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG [RFC6020] is a modular language that represents data structures in an XML or JSON tree format, and is used as a data modeling language for the NETCONF.

This document defines a YANG data model for Layer 2 Provider Provisioned VPN service configuration. These services include
Virtual Private Wire Service (VPWS) and Virtual Private LAN service (VPLS).

2. Terminology

3. Typical Scenario

The idea of the Lay 2 VPN service model is to propose an abstracted interface to manage configuration of components of a Lay 2 VPN service. A typical usage is to use this model as an input for an orchestration layer who will be responsible to translate it to orchestrated configuration of network elements which will be part of the service.

Figure 1 is the typical scenario for the SDN based layer 2 VPN Service. The Layer 2 service YANG data model is used between the orchestration and controller 1 and controller 2 and as the input for the orchestrator which is responsible to translate the application service to configure the network elements. The interfaces between controller and orchestration are called north bound interfaces, so the YANG data model defined in this document is also called north bound YANG service data model for Lay 2 VPN. The interfaces between controller and element networks are called south bound interfaces, and the YANG data model as the input of controller to configure the network elements is defined in document [I-D.shah-bess-l2vpn-yang].

There are two network element domains in the figure: domain A and domain C. The network elements in domain A are A11, A12, A21, A22, and A31 and A32. A31 and A32 are ASBRs, which are the edge routers connect to the other domains. Network elements C11, C12, C21, C22, C31, and C32 are in the domain C, and C31 and C32 are ASBRs in domain C.
4. L2VPN Service Common

4.1. PW-Template

The pw-templates container contains a list of pw-template. Each pw-template defines a list of common pseudowire attributes such as PW MTU, control word negotiation, cc, cv etc.

```yang
  +--rw pw-template* [name]
      +--rw name              string
      +--rw mtu?              uint32
      +--rw cw-negotiation?   cw-negotiation-type
      +--rw cc                cc-type
      +--rw cv                cv-type
```

4.2. QoS Policy Template

The QoS policy template defines the mapping policy, car policy and queue-schedule-policy. The QoS policy template is referenced by ac container.
5. ELINE and ELAN Service

5.1. PW List

If the L2VPN Service (ELINE service or ELAN service) needs to cross the AS domain, the PW container is configured as the following tree structure. If the L2VPN service is established in one domain, it is no need to configure PW container.

The asbr-id leaf is used to configure the edge ASBR of the domain. As the figure 1 shows, there are two ASBRs in the domain A, so it is required to indicate which exit ASBR it is when the controller establishes the cross PW.

If the L2VPN service is ELAN service, it is necessary to configure the split-horizon-group for the PW list.
5.2. AC List

Each ELINE and ELAN services define a list of AC that are participating members of the given service instance. The leaf ne-id is used to configure the ingress and egress ac of the end to end L2VPN service tunnel. The access type of ac could be port, dot1q, qinq, etc.

5.3. Service Policy

Service policy is only used in a single AS domain. The service type, signaling type, pw type and oam policy are defined in the service policy container. The tree structure of service policy is designed as following:

```plaintext
    +--rw service-policy
         |  +--rw service-type?  l2vpn-service-type
         |  +--rw signaling-type  l2vpn-signaling-type
         |  +--rw type?          pw-type
         |  +--rw pw-template?    string
         |  +--rw network?        network-type
         |  +--rw oam-policy
         |     |  +--rw (detect-type)?
         |     |      +--:(undetect)
         |     |      +--:(BFD)
         |     |          |  +--rw local-discriminator?  int32
         |     |          |  +--rw remote-discriminator? int32
         |     |          |  +--rw tx-interval?         int32
         |     |          |  +--rw rx-interval?         int32
         |     |          |  +--rw multiplier?          int32
```
5.4. Tunnel Policy

The tunnel policy is used to configure the L2VPN underlay network’s parameters. The signal type, tunnel mode and protect type are defined in the container. The tree structure of tunnel policy is designed as following:

```
  +--rw tunnel-config
    |  +--rw tunnel-policy
    |     |  +--rw tunnel-signaling-type? tunnel-signaling-type
    |     |  +--rw tunnel-mode? tunnel-mode
    |     |  +--rw protect-type? protect-type
    |     |  +--rw receive-mode? receive-mode
    |     |  +--rw (revertive-type)?
    |     |     |  +--:(never)
    |     |     |  +--:(wtr)
    |     |     |  +--rw revert-delay? uint16
```

5.5. Tree Design for ELINE Service YANG Data Model

The tree design for VPWS instance is as following:

```
  +--rw eline-service
    |  +--rw eline-instance* [name]
    |     |  +--rw name string
    |     |  +--rw description? string
    |     |  +--rw pw* [name]
    |     |     |  +--rw name string
    |     |     |  +--rw asbr-id? string
    |     |     |  +--rw node-role? node-role
    |     |     |  +--rw peer? inet:ip-address
    |     |     |  +--rw vcid? uint32
    |     |     |  +--rw type? pw-type
    |     |     |  +--rw tunnel-policy? string
    |     |     |  +--rw request-vlanid? uint16
    |     |     |  +--rw vlan-tpid? string
    |     |     |  +--rw pw-template? string
    |     |     |  +--ro cw-negotiation? cw-negotiation-type
    |     |     |  +--ro cc? cc-type
    |     |     |  +--ro cv? cv-type
    |     |     |  +--rw pw-oam-policy
    |     |     |     |  +--rw (detect-type)?
    |     |     |     |     |  +--:(undetect)
    |     |     |     |     |  +--:(BFD)
```
++rw local-discriminator? int32
++rw remote-discriminator? int32
++rw tx-interval? int32
++rw rx-interval? int32
++rw multiplier? int32

++rw ac* [name]
  ++rw name string
++rw qos-input-templates
    ++rw templates* [template-name]
      ++rw template-name string
      ++rw template-type? template-type
++rw qos-output-templates
    ++rw templates* [template-name]
      ++rw template-name string
      ++rw template-type? template-type
++rw node-role? node-role
++rw pipe-type? pipe-mode
++rw link-discovery-protocol-type? link-discovery-protocol-type
++rw (access-type)?
    +=: (port)
    +=: (dot1q)
      ++rw dot1q-vlan-bitmap? int32
    +=: (qinq)
      ++rw qinq-svlan-bitmap? int32
      ++rw qinq-cvlan-bitmap? int32
++rw (access-action)?
    +=: (keep)
    +=: (push)
    +=: (pop)
    +=: (swap)
  ++rw swap-vlan-id? int32
++rw service-redundancy
  ++rw primary-pw
    ++rw nodeid? string
    ++rw pw-name? string
  ++rw backup-pw
    ++rw nodeid? string
    ++rw pw-name? string
  ++rw communicate-unit? int32
++rw service-policy
  ++rw service-type? 12vpn-service-type
5.6. Tree Design for ELAN Service YANG Data Model

The tree design for VPLS instance is as following:

```yang
++-rw elan-service
  ++-rw elan-instance* [name]
    ++-rw name                   string
    ++-rw description?           string
    ++-rw mac-withdraw?          boolean
```
6. L2VPN Service YANG Data Model

<CODE BEGINS> file "ietf-mpls-l2vpn-svc@2016-08-03.yang"

module ietf-mpls-l2vpn-svc {
    prefix "mpls-l2vpn-svc";
    import ietf-inet-types
    {prefix inet;}
    organization "ietf";
    contact "ietf";
    description "mpls-l2vpn-svc";
    revision "2016-08-03" {
        description "update the QoS policy templates.";
        reference "draft-hu-bess-l2vpn-service-yang-0".txt";
    }
    revision "2016-03-17" {
        description "Initial revision of mpls-l2vpn-service model.";
        reference "draft-hu-bess-l2vpn-service-yang-00.txt";
    }
    /* identities */
    identity link-discovery-protocol {
        description "Base identity from which identities describing link discovery protocols are derived.";
        reference "draft-hu-bess-l2vpn-service-yang-0".txt";
    }
    identity lacp {
        ...
identity lldp {
    base "link-discovery-protocol";
    description "This identity represents LLDP";
}

identity bpdu {
    base "link-discovery-protocol";
    description "This identity represents BPDU";
}

identity cpd {
    base "link-discovery-protocol";
    description "This identity represents CPD";
}

identity udld {
    base "link-discovery-protocol";
    description "This identity represents UDLD";
}

/* typedefs */

typedef bitmap {
    type string {
        pattern '^(\d+\-\d+,)*\d+,\d+$';
    }
    description "This typedef provide the capability that user can use it to in" + "put multi-sections parameter and these parameters will be stor" + "ed with bit style. For example,\"1-300,400,450-500\",etc. The" + " format of one section is \"from-to\" or a fixed digit, \"from" + " \"and \"to\" are also a digit and \"to\" must be greater than" + " \"; the next section's \"from\" or fixed digit must be" + "greater than the previous's \"to\" or fixed digit. sections ar" + "e concatenated by comma (\",\")";
}

typedef qos-dscp-bitmap {
    type bitmap {
        pattern '^(\d+\-\d+,)*\d+,\d+$';
    }
    description "This typedef provide the capability that user can use it to in" + "put multi-sections parameter and these parameters will be stor" + "ed with bit style. For example,\"1-300,400,450-500\",etc. The" + " format of one section is \"from-to\" or a fixed digit, \"from" + " \"and \"to\" are also a digit and \"to\" must be greater than" + " \"; the next section's \"from\" or fixed digit must be" + "greater than the previous's \"to\" or fixed digit. sections ar" + "e concatenated by comma (\",\")";
}
typedef vni-bitmap {
  type string;
  description 'The definition of vni in range mode, support single vni'
    + 'value or range(max num:4;spacing mark:,) in <1-16777215>. For exampl'
    + 'e,"6,10-30,40,50-60",etc.;'
}

typedef qos-prec-bitmap {
  type bitmap {
    pattern '^(((\[0-7\]-\[0-7\])|\[0-7\]),){0,3}((\[0-7\]-\[0-7\])|\[0-7\]){1,1})$' {
      description 'Configure IP-Precedence in range mode, support single prec'
        + 'edence value or range(max num:4;spacing mark:,) in <0-7>.';
    }
  }
  description 'The definition of QoS IP-Precedence in range mode, support single pre' '
    + 'idence value or range(max num:4;spacing mark:,) in <0-7>. For exampl'
    + 'e,"0,4-7",etc.;'
}

typedef qos-mpls-bitmap {
  type bitmap {
    pattern '^(((\[0-7\]-\[0-7\])|\[0-7\]),){0,3}((\[0-7\]-\[0-7\])|\[0-7\]){1,1})$' {
      description 'Configure MPLS-EXP in range mode, support single MPLS-EXP '
        + 'value or range(max num:4;spacing mark:,) in <0-7>.';
    }
  }
  description 'The definition of QoS MPLS-EXP in range mode, support single MPLS-EXP '
    + 'value or range(max num:4;spacing mark:,) in <0-7>. For exampl'
    + 'e,"0,3-5,7",etc.;'
}

typedef qos-vlan-bitmap {
  type bitmap {
    pattern '^((((([1-9]\[0-9]{1,2})|([2-9]\[0-9]{0,0})|1|([1-3]\[0-9]{3,3})|(40' '
      + '0[0-8]\[0-9]\{0,1\}|(409[0-3])|(4094)-((([1-9]\[0-9]{1,2})|((2-9]\[0-9]'

typedef qos-8021p-bitmap {
    type bitmap {
        pattern '^(((([0-7]-[0-7])|[0-7]),){0,3}(([0-7]-[0-7])|([0-7])){1,1})$'
        
        description 'Configure 802.1p in range mode, support single 802.1p value or'
        + ' e or range(max num:4;spacing mark:,) in <0-7>.';
    }
}

description 'The definition of QoS 802.1p in range mode, support single 802.1p value or'
+ 'range(max num:4;spacing mark:,) in <0-7>.

typedef traffic-direction-type {
    type enumeration {
        enum input {
            value 1 ;
            description "input.";
        }
        enum output {
            value 2 ;
            description "output.";
        }
    }

    description "traffic direction type";
}
typedef template-type {
      type enumeration {
        enum car-template {
          value 1;
          description "car template.";
        }
        enum mapping-template {
          value 2;
          description "mapping template.";
        }
        enum queue-template {
          value 3;
          description "queue template.";
        }
      }
      description "template type";
    }

typedef l2vpn-service-type {
      type enumeration {
        enum ethernet {
          value 0;
          description "Ethernet service";
        }
        enum ATM {
          value 1;
          description "Asynchronous Transfer Mode";
        }
        enum FR {
          value 2;
          description "Frame-Relay";
        }
        enum TDM {
          value 3;
          description "Time Division Multiplexing";
        }
      }
      description "L2VPN service type";
    }

typedef network-type {
      type enumeration {
        enum simple {
          value 0;
          description "simple p2p";
        }
        enum simple-protection {
          value 1;
description "p2p with protection";
}
enum add-2drop {
    value 2;
    description "3AC";
}
enum double-inter-connect {
    value 3;
    description "DNI-PW 3AC";
}
enum l2l3bridge {
    value 4;
    description "l2l3bridge";
}
}
description "L2VPN service type";
}
typedef l2vpn-signaling-type {
    type enumeration {
        enum static {
            value 0;
            description "Static configuration of labels (no signaling)";
        }
        enum ldp {
            value 1;
            description "Label Distribution Protocol (LDP) signaling";
        }
        enum bgp {
            value 2;
            description "Border Gateway Protocol (BGP) signaling";
        }
        enum mixed {
            value 3;
            description "Mixed";
        }
    }
    description "L2VPN signaling type";
}
typedef tunnel-signaling-type {
    type enumeration {
        enum static {
            value 0;
            description "static";
        }
        enum RSVP-TE {
            value 1;
            description "RSVP-TE signaling";
        }
    }
    description "L2VPN tunnel signaling type";
}
typedef bgp-rt-type {
    type enumeration {
        enum import {
            value 0;
            description "For import";
        } enum export {
            value 1;
            description "For export";
        } enum both {
            value 2;
            description "For both import and export";
        }
    } description "BGP route-target type. Import from BGP YANG";
}

typedef cw-negotiation-type {
    type enumeration {
        enum "non-preferred" {
            value 0;
            description "No preference for control-word";
        } enum "preferred" {
            value 1;
            description "Prefer to have control-word negotiation";
        }
    } description "control-word negotiation preference type";
}

typedef link-discovery-protocol-type {
    type identityref {
        base "link-discovery-protocol";
    } description "This type is used to identify link discovery protocol";
}
typedef cc-type {
    type enumeration {
        enum pw-ach {
            value 0;
            description "PWE3 Control Word with 0001b as first nibble (PW-ACH, see [RFC4385])";
        }
        enum alert-label {
            value 1;
            description "MPLS Router Alert Label";
        }
        enum ttl {
            value 2;
            description "MPLS PW Label with TTL == 1";
        }
        enum ach-alert-ttl {
            value 3;
            description "all";
        }
    }
    description "The defined values for CC(Control Channel) Types for MPLS PWs.";
}

typedef cv-type {
    type enumeration {
        enum BFD-basic-ip {
            value 0;
            description "BFD basic ip";
        }
        enum BFD-basic-raw {
            value 1;
            description "BFD basic raw";
        }
        enum BFD-signalling-ip {
            value 2;
            description "BFD signalling ip";
        }
        enum BFD-signalling-raw {
            value 3;
            description "BFD signalling raw";
        }
    }
    description "The defined values for CV(Connectivity Verification) Types for MPLS PWs";
}

typedef pipe-mode{
    type enumeration {
        enum pipe {
            value 0;
        }
}
description "regular pipe mode";
enum "short-pipe" {
    value 1;
    description "short pipe mode";
}
enum "uniform" {
    value 2;
    description "uniform pipe mode";
}

description " ";

typedef pw-type {
    type enumeration {
        enum unknown {
            value 0 ;
            description "The PW type is unknown";
        }
        enum dlciOld {
            value 1 ;
            description "The PW type is dlciOld";
        }
        enum atmSdu {
            value 2 ;
            description "The PW type is atmSdu";
        }
        enum atmCell {
            value 3 ;
            description "The PW type is atmCell";
        }
        enum vlan {
            value 4 ;
            description "The PW type is vlan";
        }
        enum ethernet {
            value 5 ;
            description "The PW type is ethernet";
        }
        enum hdlc {
            value 6 ;
            description "The PW type is hdlc";
        }
        enum ppp {
            value 7 ;
            description "The PW type is ppp";
        }
    }
}
enum sdhCESoM {
  value 8;
  description "The PW type is sdhCESoM";
}
enum atmVCCn {
  value 9;
  description "The PW type is atmVCCn";
}
enum atmVPCn {
  value 10;
  description "The PW type is atmVPCn";
}
enum ipL2 {
  value 11;
  description "The PW type is ipL2";
}
enum atmVCC1 {
  value 12;
  description "The PW type is atmVCC1";
}
enum atmVPC1 {
  value 13;
  description "The PW type is atmVPC1";
}
enum atmPDU {
  value 14;
  description "The PW type is atmPDU";
}
enum frPort {
  value 15;
  description "The PW type is frPort";
}
enum sdhCEoP {
  value 16;
  description "The PW type is sdhCEoP";
}
enum saTopE1 {
  value 17;
  description "The PW type is saTopE1";
}
enum saTopT1 {
  value 18;
  description "The PW type is saTopT1";
}
enum saTopE3 {
  value 19;
  description "The PW type is saTopE3";
}
enum saTopT3 {
  value 20;
  description "The PW type is saTopT3";
}
enum ceSoPSNB {
  value 21;
  description "The PW type is ceSoPSNB";
}
enum tdmAAL1 {
  value 22;
  description "The PW type is tdmAAL1";
}
enum ceSoPSNC {
  value 23;
  description "The PW type is ceSoPSNC";
}
enum tdmAAL2 {
  value 24;
  description "The PW type is tdmAAL2";
}
enum dlciNew {
  value 25;
  description "The PW type is dlciNew";
}

typedef node-role {
  type enumeration {
    enum "none" {
      value 0;
      description "";
    }
    enum "master" {
      value 1;
      description "";
    }
    enum "slave" {
      value 2;
      description "";
    }
    default none;
    description "node-role";
  }
}
typedef protect-type {
    type enumeration {
        enum unprotected {
            value 0;
            description "unprotected.";
        }
        enum 1to1 {
            value 1;
            description "protect type is 1:1";
        }
        enum 1plus1 {
            value 2;
            description "protect type is 1+1";
        }
        enum dni-pw {
            value 3;
            description "protect type is dni-pw";
        }
        enum dni-ac {
            value 4;
            description "protect type is dni-ac";
        }
    }
    description "define the protect type";
}

typedef receive-mode {
    type enumeration {
        enum selective {
            value 0;
            description "receive mode is selective";
        }
        enum both {
            value 1;
            description "receive mode is both";
        }
    }
    description "define the receive mode";
}

typedef tunnel-mode {
    type enumeration {
        enum static {
            value 2;
            description "static tunnel";
        }
        enum RSVP-TE {
            description "define the tunnel mode";
        }
    }
    description "define the tunnel mode";
}
typedef hub-spoken {
  type enumeration {
    enum hub {
      value 0;
      description "the hub role in the network";
    }
    enum spoken {
      value 1;
      description "the spokend role in the network";
    }
  }
  description "define the hub spoken type";
}

grouping match-criteria {
  leaf match-relation {
    type enumeration {
      enum match-all {
        value 1;
        description 'Packets meet all of the match criteria';
      }
      enum match-any {
        value 2;
        description 'Packets meet any of the match criteria';
      }
    }
    mandatory true;
    description 'The type of packets matching, see bellow:
    + ' match-all Packets meet all of the match criteria'
    + ' match-any Packets meet any of the match criteria';
  }
}
description "match-criteria";
}

grouping rule {

container match-elements {
    leaf dscp-value {
        type qos-dscp-bitmap;
        description 'Configure QoS dscp in range mode, support single DSCP value or range(max num:4; spacing mark:,) in <0-63>. For example,"6,10-30,40,50', 
                            + '-60", etc.';
    }

    leaf precedence-value {
        type qos-prec-bitmap;
        description 'Configure QoS IP Precedence in range mode, support single precedence value or range(max num:4; spacing mark:,) in <0-7>. For example,"0,2-4,7", etc.';
    }

    leaf mpls-exp-value {
        type qos-mpls-bitmap;
        description 'Configure QoS MPLS-EXP value in range mode, support single MPLS-EXP value or range(max num:4; spacing mark:,) in <0-7>. For example,"0,2-4,7", etc.';
    }

    leaf vni {
        type vni-bitmap;
        description "";
    }

    leaf vlan-value {
        type qos-vlan-bitmap;
        description 'Configure outer-VLAN value in range mode, support single outer-VLAN value or range(max num:4; spacing mark:,) in <1-4094>. For example,"1,3', 
                            + '00-400,1000,2000-4000", etc.';
    }

    leaf out-8021p-value {

type qos-8021p-bitmap;
description 'Configure out 802.1p value in range mode, su'
+ 'pport single 802.1p value or range(max num:4;s'
+ 'pacing mark:,) in <0-7>. For example,"0,2-4,7","'
+ ',etc.';
}
description 'Set value as match criteria, no rules means match all traffic';
}
description "rule ";
}
grouping set{

carrier dscp {
  presence 'Select DSCP as mark typ' ;
  choice method {
    mandatory true ;
    case set-value {
      leaf dscp-value {
        type uint16 {
          range 0..63 {
            description 'DSCP value, support <0-63>' ;
          }
        }
        mandatory true ;
        description 'DSCP value, support <0-63>' ;
      }
      description 'Set the DSCP value directly' ;
    }
    description 'Specify the method to set DSCP' ;
  }
  description 'Select DSCP as mark type' ;
}

carrier ip-precedence {
  presence 'Select precedence as mark type' ;
  choice method {
    mandatory true ;
    case set-value {
      leaf prec-value {
        type uint16 {
          range 0..7 {
            description 'IP Precedence value, support <0-7>' ;
          }
        }
        mandatory true ;
      }
    }
  }
}
description 'IP Precedence value, support <0-7>';
}
description 'Set the IP Precedence value directly';
}
description 'Specify the method to set IP Precedence';
}
description 'Select precedence as mark type';
}

container mpls-exp {
    presence 'Select MPLS-EXP as mark type';
    choice method {
        mandatory true;
        case set-value {

            leaf mpls-exp-value {

                type uint16 {
                    range 0..7 {
                        description 'MPLS-EXP value, support <0-7>';
                    }
                }
                mandatory true;
                description 'MPLS-EXP value, support <0-7>';
            }
            description 'Set the MPLS-EXP value directly';
        }
        description 'Specify the method to set MPLS-EXP';
    }
    description 'Select MPLS-EXP as mark type';
}

carrier-association {
    presence 'Select carrier-association as mark type';
    leaf carrier-association-value {

        type uint32 {
            range 0..7 {
                description 'carrier-association value, support <0-7>';
            }
        }
        mandatory true;
        description 'carrier-association value, support <0-7>';
    }
    description 'Select carrier-association as mark type';
}

carrier-breadth {
    presence 'Select carrier-breadth as mark type';
    leaf carrier-breadth-value {

        type uint32 {
            range 0..7 {
                description 'carrier-breadth value, support <0-7>';
            }
        }
        mandatory true;
        description 'carrier-breadth value, support <0-7>';
    }
    description 'Select carrier-breadth as mark type';
}

carrier-monotonicity {
    presence 'Select carrier-monotonicity as mark type';
    leaf carrier-monotonicity-value {

        type uint32 {
            range 0..7 {
                description 'carrier-monotonicity value, support <0-7>';
            }
        }
        mandatory true;
        description 'carrier-monotonicity value, support <0-7>';
    }
    description 'Select carrier-monotonicity as mark type';
}

cp-explicit-path {
    presence 'Select cp-explicit-path as mark type';
    leaf cp-explicit-path-value {

        type uint32 {
            range 0..7 {
                description 'cp-explicit-path value, support <0-7>';
            }
        }
        mandatory true;
        description 'cp-explicit-path value, support <0-7>';
    }
    description 'Select cp-explicit-path as mark type';
}

cp-explicit-path-lsp {
    presence 'Select cp-explicit-path-lsp as mark type';
    leaf cp-explicit-path-lsp-value {

        type uint32 {
            range 0..7 {
                description 'cp-explicit-path-lsp value, support <0-7>';
            }
        }
        mandatory true;
        description 'cp-explicit-path-lsp value, support <0-7>';
    }
    description 'Select cp-explicit-path-lsp as mark type';
}

cp-explicit-path-lsp-sub {
    presence 'Select cp-explicit-path-lsp-sub as mark type';
    leaf cp-explicit-path-lsp-sub-value {

        type uint32 {
            range 0..7 {
                description 'cp-explicit-path-lsp-sub value, support <0-7>';
            }
        }
        mandatory true;
        description 'cp-explicit-path-lsp-sub value, support <0-7>';
    }
    description 'Select cp-explicit-path-lsp-sub as mark type';
}

container pbts {
    presence 'Select PBTS as mark type';
    leaf pbts-value {

        type uint8 {
            range 0..7 {
                description 'PBTS value, support <0-7>';
            }
        }
        mandatory true;
        description 'PBTS value, support <0-7>';
    }
    description 'Select PBTS as mark type';
}

description 'Mark a packet';

grouping interface-key {
  leaf node {
    type string;
    description "node name";
  }
  leaf interface-name {
    type string;
    description "interface name";
  }
  leaf traffic-direction {
    type traffic-direction-type;
    description "traffic direction";
  }
  description "interface-key";
}

grouping car {
  leaf cir {
    type uint32 {
      range 8..400000000 {
        description 'The CIR value (default: kilobits per sec' + 'ond),support <8-400000000>'';
      }
    }
    mandatory true;
    description 'Configure the CIR(Committed Information Rate' + ' ) value (kilobits per second)'';
  }
  leaf cbs {
    type uint32 {
      range 15000..2000000000 {
        description 'The CBS(Committed Burst Size) size (defa' + 'ult: byte), support <15000-2000000000>'';
      }
    }
    mandatory true;
    description 'Configure the CBS(Committed Burst Size) size' + ' (default: byte)'';
  }
  leaf pir {
    type uint32 {
      range 8..400000000 {
        description 'The PIR value,pir must >= cir, (default: kilobits per sec' + 'ond),support <8-400000000>'';
      }
    }
    }
mandatory true;
description 'Configure the PIR(Committed Information Rate')
+ ' value (kilobits per second)';
}
leaf pbs {
    type uint32 {
        range 15000..2000000000 {
            description 'The PBS(Peak Burst Size) size (default: byte), support <15000-2000000000>,'
        }
    }
    mandatory true;
description 'Configure the PBS(Peak Burst Size) size (d'
+ 'efault: byte)';
}
description 'set car parameter';
}
grouping schedule-policy {
    choice method {
        case wfq {
            leaf percentage {
                type uint16 {
                    range 1..100 {
                        description 'The exact bandwidth percentage, support '
+ '<1-100>';
                    }
                }
                mandatory true;
description 'Set the exact bandwidth percentage, support '
+ '<1-100>>';
            }
        }
        case pq {
            leaf priority-value {
                type uint16 {
                    range 1..4 {
                        description 'The priority level of PQ, support <1-4>,'
                    }
                }
                mandatory true;
description 'Set the priority level of PQ, support <1-4>',';
            }
        }
    }
    description 'Specify the method to set schedule-policy';
}
description 'set schedule-policy, support PQ or WFQ or PQ+WFQ' ;

grouping car-template {
  leaf car-template-name {
    type string;
    description "car template name.";
  }
  list policy-map {
    key "policy-map-name";
    leaf policy-map-name {
      type string;
      description "policy map name";
    }
    container class-policy {
      uses match-criteria;
      uses rule;
      description "class policy.";
    }
    container action {
      uses car;
      description "action.";
    }
    description "policy map list";
  }
  description "car template";
}

grouping queue-schedule-template {
  leaf queue-schedule-template-name {
    type string;
    description "name";
  }
  list policy-map {
    key "policy-map-name";
    leaf policy-map-name {
      type string;
      description "policy map name";
    }
    container class-policy {
      uses match-criteria;
      uses rule;
      description "class policy";
    }
    container action {
      uses schedule-policy;
      description "action";
    }
  }
  description "queue-schedule-template";
}
grouping mapping-template {
    leaf mapping-template-name {
        type string;
        description "mapping template name.";
    }
    list policy-map {
        key "policy-map-name";
        leaf policy-map-name {
            type string;
            description "policy map name";
        }
        container class-policy {
            uses match-criteria;
            uses rule;
            description "class policy.";
        }
        container action {
            uses set;
            description "action.";
        }
        description "policy map";
    }
    description "mapping template";
}

grouping template{
    list templates {
        key "template-name";
        leaf template-name {
            type string;
            description "template name";
        }
        leaf template-type {
            type template-type;
            description "template type";
        }
        description "template list";
    }
    description "template";
}

case undetect {
    description "unprotected.";
}

case BFD {
    description "bfd.";
    leaf local-discriminator {
        type int32;
        description "local-discriminator";
    }
    leaf remote-discriminator {
        type int32;
        description "remote-discriminator";
    }
    leaf tx-interval {
        type int32;
        description "tx-interval";
    }
    leaf rx-interval {
        type int32;
        description "rx-interval";
    }
    leaf multiplier {
        type int32;
        description "detect-interval";
    }
    description "OAM type";
}

description "oam type";
}

grouping pw-template{
    description "pw-template";
    leaf name {
        type string;
        description "name";
    }
    leaf cw-negotiation {
        type cw-negotiation-type;
        default "preferred";
        description "control-word negotiation preference";
    }
    leaf cc {
        type cc-type;
        description "Control Channel Types";
    }
}
leaf cv {
    type cv-type;
    description "Connectivity Verification Types";
}

grouping pseudowire{
    description "pseudowire";
    leaf name {
        type string;
        description "pseudowire name";
    }
    leaf asbr-id {
        type string;
        description "asbr name for the cross domain lsp";
    }
    leaf node-role{
        type node-role;
        description "node role.";
    }
    leaf peer {
        type inet:ip-address;
        description "pw peer address, use IPv4";
    }
    leaf vcid {
        type uint32;
        description "pseudo-wire vcid";
    }
    leaf type {
        type pw-type;
        description "pseudo-wire type";
    }
    leaf tunnel-policy {
        type string;
        description "Used to override the tunnel policy name specified in the pseudowire template";
    }
    leaf request-vlanid {
        type uint16;
        description "request vlanid";
    }
}
leaf vlan-tpid {
    type string;
    description "vlan tpid";
}

leaf pw-template {
    type string;
    description "pw templet name";
}

leaf cw-negotiation {
    type cw-negotiation-type;
    default "preferred";
    config false;
    description "control-word negotiation preference";
}

leaf cc {
    type cc-type;
    config false;
    description "Control Channel Types";
}

leaf cv {
    type cv-type;
    config false;
    description "Connectivity Verification Types";
}

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

container qos-input-templates{
    uses template;
    description "qos templates.";
}
container qos-output-templates{
    uses template;
    description "qos templates. ";
}
leaf node-role{
    type node-role;
    description "node role.";
}
leaf pipe-type{
    type pipe-mode;
    description "pipe mode. Can update.";
}
leaf link-discovery-protocol-type{
    type link-discovery-protocol-type;
    description "link discovery protocol";
}
choice access-type{
    description "access-type";
    case port{
        description "port.";
    }
    case dot1q {
        description "Dot1Q";
        leaf dot1q-vlan-bitmap {
            type int32 {
                range "1..4094";
            }
            description "Dot1Q Vlan Bitmap.";
        }
    }
    case qinq {
        description "QinQ";
        leaf qinq-svlan-bitmap {
            type int32 {
                range "1..4094";
            }
            description "QinQ svlan Bitmap.";
        }
        leaf qinq-cvlan-bitmap {
            type int32 {
                range "1..4094";
            }
        }
    }
}
description "QinQ cvlan Bitmap." ;
}
)
}

choice access-action {
  description "access type." ;
  case keep {
    description "keep." ;
  }
  case push {
    description "push." ;
    leaf push-vlan-id {
      type int32 {
        range "1..4094";
      }
      description "action vlan id." ;
    }
  }
  case pop {
    description "pop." ;
  }
  case swap {
    description "swap." ;
    leaf swap-vlan-id {
      type int32 {
        range "1..4094";
      }
      description "action vlan id." ;
    }
  }
}

container service-redundancy {
  uses redundancy-policy;
  description "service redundancy";
}

grouping protect-policy-grp {
  leaf protect-type {
    type protect-type;
    description "protection-type";
  }
  leaf receive-mode {
    type receive-mode;
    description "receive-mode";
  }
}
choice revertive-type {
  description "revertive-type";
  case never {
    description "never type";
  }
  case wtr {
    leaf revert-delay {
      type uint16;
      description "the revertive type is wtr";
    }
  }
}

description "define the group of protect-policy.";

grouping redundancy-policy{
  container primary-pw{
    description "primary options";
    leaf nodeid{
      type string;
      description "The nodeid of the policy.";
    }
    leaf pw-name {
      type string;
      description "Reference a pseudowire";
    }
  }
  container backup-pw {
    description "backup options";
    leaf nodeid{
      type string;
      description "The nodeid of the policy.";
    }
    leaf pw-name {
      type string;
      description "Reference a pseudowire";
    }
  }
  leaf communicate-unit{
    type int32;
    description "communicate id for dni-pw";
  }
}
container protect-policy{
    description "protect-policy";
    leaf protect-object{
        type enumeration {
            enum ac-policy {
                value 0;
                description "ac-policy for dni-pw";
            }
            enum pw-policy {
                value 1;
                description "pw-policy for master-slave pw";
            }
        }
        description "protect object";
    }
    uses protect-policy;
}
description "redundancy-policy";

grouping protect-policy {
    leaf protect-type {
        type enumeration {
            enum unprotected {
                value 0;
                description "unprotected.";
            }
            enum 1to1 {
                value 1;
                description "1to1 protection.";
            }
            enum 1plus1 {
                value 2;
                description "1plus1 protection.";
            }
            enum loadbalance {
                value 3;
                description "loadbalance for ac protection.";
            }
        }
        description "protection type";
    }
    leaf receive-mode {
        type enumeration {
            enum selective {
                value 0;
            }


```yang

description "selective.";

enum both {
  value 1;
  description "both.";
}

description "receive mode";

leaf revertive-type {
  type enumeration {
    enum no-revertive {
      value 0;
      description "unprotected.";
    }
    enum revertive {
      value 1;
      description "protection.";
    }
  }
  description "revertive mode. Can update.";
}

leaf wtr {
  type uint8 {
    range "0..12";
    default 5;
    description "WTR.unit:m. Can update.";
  }
  description "protect policy";
}

grouping vpws-service-policy-grp {
  leaf service-type {
    type l2vpn-service-type;
    default ethernet;
    description "eline service type";
  }

  leaf signaling-type {
    type l2vpn-signaling-type;
    mandatory true;
    description "eline signaling type";
  }

  leaf type {
```
type pw-type;
   description "pseudo-wire type";
}

leaf pw-template {
   type string;
   description "pw templet name";
}

leaf network {
   type network-type;
   description "network-type";
}

container oam-policy {
   uses oam-policy;
   description "bfd param";
}
   description "vpws service policy group";
}

grouping vpls-service-policy-grp {
   leaf split-horizon-group {
      type string;
      description "split horizon group for the vpls instance";
   }

   leaf pw-type {
      type pw-type;
      description "the pseudowires type";
   }

container oam-policy {
   uses oam-policy;
   description "bfd param";
}
   description "vpls service policy group";
}

grouping tunnel-policy-grp {
   container tunnel-policy {
      leaf tunnel-signaling-type {
         type tunnel-signaling-type;
         description "signaling-type";
      }
      leaf tunnel-mode {
         type tunnel-mode;
         description "tunnel-mode";
      }
}
uses protect-policy-grp;

description "tunnel policy";
}
description "tunnel policy group";
}
grouping mspw-path{
  list pw-path{
    key "serialNumber source destination";

    leaf serialNumber {
      type int16;
      description "serial number of link.";
    }

    leaf source{
      type string;
      description "the source node id ";
    }

    leaf destination{
      type string;
      description "the destination node id ";
    }

    leaf signaling-type {
      type l2vpn-signaling-type;
      description "eline signaling type";
    }

    description "pw path.";
  }
  description "multi-segment pw path";
}

container l2vpn {
  description "l2vpn";
  container common {
    container pw-templates {
      description "pw-templates";
      list pw-template {
        key "name";
        uses pw-template;
        description "pw-templates";
      }
    }
  }
}
container qos-policy-templates {
    list mapping-policy {
        uses mapping-template;
        key 'mapping-template-name';
        list binding-interface {
            key "node interface-name traffic-direction";
            uses interface-key;
            description "binding-interface list.";
        }
        description "mapping policy list.";
    }
}

list car-policy {
    uses car-template;
    key 'car-template-name';
    list binding-interface {
        key "node interface-name traffic-direction";
        uses interface-key;
        description "binding-interface list.";
    }
    description "car policy list.";
}

list queue-schedule-policy {
    uses queue-schedule-template;
    key 'queue-schedule-template-name';
    list binding-interface {
        key "node interface-name traffic-direction";
        uses interface-key;
        description "binding-interface list.";
    }
    description "queue schedule policy list.";
}

description "qos policy template";
}

description "common l2pn attributes";
}

container eline-service {
    description "configure elan-instances";
    list elan-instance {
        key "name";
        leaf name {
            type string;
            description "the name of a eline service instance";
        }
    }
}
leaf description {
  type string;
  description "Description of the eline service instance";
}

list pw{
  key "name";
  uses pseudowire;
  description "pseudowires";
}

list ac{
  key "name";
  uses attachment-circuit;
  description "attachment-circuit";
}

container service-policy{
  uses vpws-service-policy-grp;
  description "The service-policy of the eline.";
}

container tunnel-config{
  uses tunnel-policy-grp;
  description "The tunnel-policy of the eline.";
}

container master-mspw-path{
  description "master mspw path.";
  uses mspw-path;
}

container slave-mspw-path{
  description "slave mspw path.";
  uses mspw-path;
}

description "vpws service instance list";
}

container elan-service {
  description "configure elan-instances";
  list elan-instance {
    key "name";
    leaf name {

type string;  
  description "the name of a elan service instance";
 }

leaf description {  
  type string;  
  description "Description of the elan service instance";
 }

leaf mac-withdraw {  
  type boolean;  
  description "Withdraw MAC";
 }

container bgp-parameters {  
  leaf route-distinguisher {  
    type string;  
    description "BGP RD";
  }

  list vpn-targets {  
    key rt-value;  
    description "Route Targets";

    leaf rt-value {  
      type string;  
      description "Route-Target value";
    }

    leaf rt-type {  
      type bgp-rt-type;  
      mandatory true;  
      description "Type of RT";
    }
  }

  container discovery {  
    description "BGP parameters for discovery";
    leaf vpn-id {  
      type string;  
      description "VPN ID";
    }

    description "Parameters for BGP";
  }

  leaf service-type {  
    type l2vpn-service-type;
  }
default ethernet;

description "vpls service type";
}

leaf signaling-type {
    type l2vpn-signaling-type;
    mandatory true;
    description "vpls signaling type";
}

list pw{
    key "name";
    uses pseudowire;
    leaf hub-spoken {
        type hub-spoken;
        description "hub-spoken role";
    }
    description "pseudowires";
}

list ac{
    key "name";
    uses attachment-circuit;
    leaf split-horizon-group {
        type string;
        description "split horizon group for the vpls instance";
    }
    description "attachment-circuit";
}

container service-policy{
    uses vpls-service-policy-grp;
    description "The service-policy of the elan.";
}

container tunnel-config{
    uses tunnel-policy-grp;
    description "The tunnel-policy of the elan.";
}

container master-mspw-path{
    description "master mspw path.";
    uses mspw-path;
}

container slave-mspw-path{
    description "slave mspw path.";
uses mspw-path;

description "vpls service instance list";

7. Security Considerations

8. Acknowledgements

9. IANA Considerations

This document requires no IANA Actions. Please remove this section before RFC publication.

10. Normative References


Authors’ Addresses
Fangwei Hu
ZTE Corporation
No.689 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Ran Chen
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Jie Yao
ZTE Corporation
Zijinghua Rd. Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Email: yao.jie@zte.com.cn

Cao Chang
Chinaunicom
Beijing
China

Email: caochang@chinaunicom.cn