PALS WG
Internet-Draft
Intended status: Standards Track
Expires: June 14, 2018

Ran. Chen
Fangwei. Hu
ZTE Corporation

December 11, 2017

YANG Data Model for PW Protocol
draft-chen-pals-pw-yang-03.txt

Abstract

This document defines a YANG data model for PW configuration and operation.

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 https://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 June 14, 2018.

Table of Contents

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 PW configuration and operation. It includes point-to-point SS-PWs and MS-PWs that are signaled via LDP, and also for static provisioned (MPLS-TP) SS-PWs and MS-PWs.

2. SS-PW

This container defines all the configuration parameters related to ss-pw.

It includes pw name, peer ip, tunnel parameters, pw-type, parameters associated with interface, parameters associated with LDP pw and parameters associated with static pw.

3. MS-PW

This container defines all the configuration parameters related to ms-pw.

It includes ms-pw name, list of pw-segment-a and list of pw-segment-z.

4. Design of the Data Model

```
module: ietf-pw
  +--rw pwe3
     +--rw ss-pw
     |  +--rw ss-pw* [name]
     |     +--rw name                    string
     |     +--rw peer-ip?                inet:ip-address
     |     +--rw cw-capable?             cw-capable-type
     |     +--rw type?                   pw-type
     |     +--rw tunnel* [tunnel-id]
     |     |  +--rw tunnel-id    string
     |     +--rw leaf-type?              pw-type
     |     +--rw autodiscovery-enable?   boolean
     |     +--rw interfaces
     |     |  +--rw interface* [name]
     |     |     +--rw name                       if:interface-ref
     |     |     +--rw mtu?                       uint32
     |     |     +--rw fcs-retention-indicator?   uint16
     |     |     +--rw vccv-parameter
     |     |     |  +--rw cc?   cc-type
     |     |     |  +--rw cv?   cv-type
     |     |     +--rw requested-vlan-id?         uint32
     |     |     +--rw frag-indicator?            uint32
     |     |     +--rw interface-description?     string
     |     |     +--rw (pw-emu-type)?
     |     |        +--:(tdm)
     |     |        |  +--rw bit-rate?                  uint32
     |     |        |  +--rw payload-bytes?             uint16
     |     |        |  +--rw cells-per-packet?          uint16
     |     |        |  +--rw tdm-options
     |     |        |  |  +--rw rtp?              pw-rtp-flag
     |     |        |  |  +--rw timestamp-mode?   pw-timestamp-mode
     |     |        |  |  +--rw frequency?        uint16
     |     |        |  |  +--rw ssrc?             uint32
     |     |        |  |  +--rw payload-type?     uint8
     |     |        |  |  +--rw cas?              uint8
     |     |        |  |  +--rw sp?               uint8
     |     |        |  +--rw cep-option
     |     |        |     +--rw ais?        uint8
     |     |        |     +--rw une?        uint8
     |     |        |     +--rw rtp?        uint8
     |     |        |     +--rw ebm?        uint8
     |     |        |     +--rw async-t3?   uint8
     |     |        |     +--rw async-e3?   uint8
     |     |        |     +--rw cep-type?   uint16
     |     |        +--:(fr)
     |     |        |  +--rw fr-dlci-len?              uint16
```

```
          |     |        +--:(atm)
          |     |           +--rw max-atm-cells?            uint16
          |     +--rw (pw-type)?
          |        +--:(ldp-pw)
          |        |  +--rw (fec-type)?
          |        |     +--:(generalized-pwid-fec-type)
          |        |     |  +--rw agi?                     string
          |        |     |  +--rw source-AII?              string
          |        |     |  +--rw target-AII?              string
          |        |     +--:(pwid-fec)
          |        |        +--rw pw-id?                  uint32
          |        +--:(static-pw)
          |           +--rw static-pw-id?           uint32
          |           +--rw transmit-label?         uint32
          |           +--rw receive-label?          uint32
          +--rw ms-pw
             +--rw ms-pw* [name]
                +--rw name            string
                +--rw pw-segment-a* [name]
                |  +--rw name    string
                +--rw pw-segment-z* [name]
                   +--rw name    string
```

5. PW YANG Data Model

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

  import ietf-inet-types {
    prefix "inet";
  }

 import ietf-interfaces {
       prefix "if";
     }

  organization  "ietf";
  contact       "ietf";
  description   "pw";
   revision "2017-04-28" {
        description "02 revision.";
        reference "draft-chen-pals-pw-yang-02.txt";
                }
  revision "2016-10-05" {
```

```
        description "01 revision.";
        reference "draft-chen-pals-pw-yang-01.txt";
                }
revision "2016-03-30" {
        description "Initial revision of pw model.";
        reference "draft-chen-pals-pw-yang-00.txt";
                }
 /*  typedefs  */

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

        typedef cc-type {
        type enumeration {
            enum pw-ach {
                description "PWE3 Control Word with 0001b as first nibble (PW-
ACH, see [RFC4385])";
            }
            enum alert-label {
                description "MPLS Router Alert Label";
            }
            enum ttl {
                description "MPLS PW Label with TTL == 1";
            }
        }
        description "The defined values for CC(Control Channel) Types for MPLS
PWs.";
    }

    typedef cv-type {
      type enumeration {
            enum ICMP-ping {
                description "ICMP-ping.";
            }
            enum LSP-ping {
                description "LSP-ping";
            }
            enum BFD-basic-ip {
                description "BFD basic ip";
```

```
            }
            enum BFD-basic-raw {
```

```
                description "BFD basic raw ";
            }
            enum BFD-signalling-ip {
                description "BFD signalling ip";
            }
            enum BFD-signalling-raw {
                description "BFD signalling raw";

   }
        }
        description "The defined values for CV(Connectivity Verification) Types
for MPLS PWs";
    }

     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";
            }
        }
        description "The PW type of the PW.";
    }

 typedef pw-rtp-flag {
    type enumeration {
      enum UNUSE {
        value 0 ;
        description 'Not use the rtp header.' ;
      }
      enum USE {
        value 1 ;
        description 'Use the rtp header.' ;
      }
      enum UNKNOWN {
        value 3 ;
        description 'The usage of the rtp header is unknown.' ;
      }
    }
    description 'The use flag of rtp header.' ;
  }

  typedef pw-timestamp-mode {
    type enumeration {
```

```
      enum Absolute {
        value 0 ;
        description 'The timestamp mode is absolute.' ;
      }
      enum Differential {
        value 1 ;
        description 'The timestamp mode is differential .' ;
      }
      enum UNKNOWN {
 value 3 ;
        description 'The timestamp mode is unknown.' ;
      }
    }
    description 'The timestamp mode of TDM service.' ;
  }

  container pwe3 {
    description "configure pw";
        container ss-pw {
         description "configure ss-pw";
     list ss-pw {

        key "name";
        leaf name {
          type string;
          description "ss-pseudowire name";
            }
         leaf peer-ip {
            type inet:ip-address;
            description "peer IP address";
        }
        leaf cw-capable {
            type cw-capable-type;
            default "preferred";
            description "control-word negotiation preference";
        }
        leaf type {
            type pw-type;
            description "pseudo-wire type";
        }

        list tunnel {
                  key "tunnel-id";
          leaf tunnel-id {
             type string;
             description "tunnel identifier";
            }
         description "tunnel list";
```

```
                }

                leaf leaf-type {
            type pw-type;
            description "pseudo-wire type";
        }
        leaf autodiscovery-enable{
  type boolean;
            description "enable the auto-discovery";
        }

        container interfaces {
         description "Interfaces";
         list interface{
                   key "name";
           leaf name {
             type if:interface-ref;
             description "Interfaces used for pw";
             }
                   leaf mtu {
             type uint32;
             description "pseudowire mtu";
            }
                   leaf fcs-retention-indicator {
                    type uint16;
             description "The negotiated fcs retention indicator of the PW";
           }
           container vccv-parameter {
            description "vccv-parameter";
                        leaf cc {
              type cc-type;
              description "Control Channel Types";
            }
            leaf cv {
              type cv-type;
              description "Connectivity Verification Types";
            }
           }

                   leaf requested-vlan-id {
                    type uint32;
            description "The local requested VLAN ID of the PW";
           }

           leaf frag-indicator {
                         type uint32;
              description "The local fragmentation indicator of the PW";
            }
```

```
            leaf interface-description {
              type string {
              length 0..81;
              }
              description "The local interface description of the PW";
             }

  choice pw-emu-type {
              description "The emulation type of the PW. It could be tdm, fr
and atm. There are different interface parameters for different emulation
types";

              case tdm {
                leaf bit-rate {
                                  type uint32;
                  description "The local bit rate of the PW";
                }

                                leaf payload-bytes {
                                  type uint16;
                  description "The local payload bytes of the PW";
                }

                leaf cells-per-packet {
                                  type uint16;
                  description "The local TDMoIP AAL1 cells per packet of the
PW";
                }

                                container tdm-options {
                  description "The TDM Options parameter of the PW";
                  leaf rtp {
                                    type pw-rtp-flag;
                    description "The local rtp header usage";
                  }
                  leaf timestamp-mode {
                                    type pw-timestamp-mode;
                    description "The local timestamp mode";
                  }
                  leaf frequency {
                                    type uint16;
                    description "The local frequency of timestamping clock";
                  }
                  leaf ssrc {
                                    type uint32;
                    description "The local value of the Synchronization source
ID";
                  }
```

```
                leaf payload-type {
                                  type uint8;
                  description "The local payload type in the RTP header
expected by the PW endpoint distributing this FEC";
                }
```

```
                  leaf cas {
                                  type uint8 ;
                    description "The local cas of the PW";
                  }
                  leaf sp {
                                  type uint8 ;
                    description "The local sp of the PW";
  }
               }

                              container cep-option {
                  description "The CEP Options parameter of the PW";
                  leaf ais {
                                  type uint8;
                    description "The local ais of CEP Options parameter of the
PW";
                  }
                  leaf une {
                                  type uint8;
                    description "The local une of CEP Options parameter of the
PW";
                  }
                  leaf rtp {
                                  type uint8;
                    description "The local rtp of CEP Options parameter of the
PW";
                  }
                  leaf ebm {
                                  type uint8;
                    description "The local ebm of CEP Options parameter of the
PW";
                  }
                  leaf async-t3 {
                                  type uint8;
                    description "The local async-t3 of CEP Options parameter of
the PW";
                  }
                  leaf async-e3 {
                                  type uint8;
                    description "The local async-e3 of CEP Options parameter of
the PW";
                  }
                  leaf cep-type {
                                  type uint16;
                    description "The local cep type of CEP Options parameter of
the PW";
               }
              }
```

```
            }
                        case fr {
              description "The emulation type of the PW is fr";
              leaf fr-dlci-len {
                                type uint16;
                description "The local fr dlci length of the PW";
              }
```

```
              }

              case atm {
                description "The emulation type of the PW is atm";
                leaf max-atm-cells {
                                  type uint16;
                  description "The local max atm cells of the PW";
  }
                           }
              }
              description "interface list";
            }
                }

                    choice pw-type {
               description "A choice of pseudowire type";
                           case ldp-pw {
                choice fec-type{
                description "fec type";
                 case generalized-pwid-fec-type {
                   leaf agi {
                     type string;
                     description "Attachment Group Identifier";
                    }
                                   leaf source-AII {
                     type string;
                     description "Source Attachment individual identifier";
                    }
                                  leaf target-AII {
                    type string;
                    description "Target Attachment individual identifier";
                    }
                                }
                                case pwid-fec{
                   leaf pw-id {
                     type uint32;
                     description "pseudowire id";
                    }
                                   }
                 }
                             }
                        case static-pw {
                leaf static-pw-id {
                     type uint32;
                     description "pseudowire id";
                }
                leaf transmit-label {
                  type uint32;
```

```
                  description "transmit lable";
                }
                leaf receive-label {
                  type uint32;
                  description "receive label";
                }
              }

    }

        description "ss-pw list";
        }
      }

          container ms-pw {
            description "configure ms-pw";
                list ms-pw {
        key "name";
        leaf name {
          type string;
          description "ms-pseudowire name";
           }
                list pw-segment-a{
                  key "name";
                  leaf name {
                  type string;
          description "pseudowire segment a name";
           }
           description "pw segment-a list";
                  }
                list pw-segment-z{
                  key "name";
                  leaf name {
                    type string;
           description "pseudowire segment z name";
           }
           description "pw segment-z list";
         }
        description "ms-pw list";
                }
         }
  }
}
```

<CODE ENDS>

6. Security Considerations

TBD.

7. Acknowledgements

TBD.

8. IANA Considerations

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

9. References

9.1. Normative references

[RFC4447] Martini, L., Ed., Rosen, E., El-Aawar, N., Smith, T., and G. Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", RFC 4447, DOI 10.17487/RFC4447, April 2006, <https://www.rfc-editor.org/info/rfc4447>.

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

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

[RFC7267] Martini, L., Ed., Bocci, M., Ed., and F. Balus, Ed., "Dynamic Placement of Multi-Segment Pseudowires", RFC 7267, DOI 10.17487/RFC7267, June 2014, <https://www.rfc-editor.org/info/rfc7267>.

9.2. Informative references

[I-D.ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-07 (work in progress), October 2017.

Authors' Addresses

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

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

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