INTERNET-DRAFT
Updates: 6074 (if approved)
Intended Status: Standards Track
Expires: August 23, 2015

Avik Bhattacharya
Apratim Mukherjee
Ixia

February 19, 2015

Provisioning, Auto-Discovery, and Signaling in L2VPNs for IPv6 Remote PE draft-abhattacharya-bess-l2vpn-ipv6-remotepe-03

Abstract

L2VPN Signaling specification defines the semantic structure of the endpoint identifiers required by each model. It discusses the distribution of these identifiers by the discovery process, especially when such discovery is based on the Border Gateway Protocol (BGP). This document updates the end point encoding for BGP-Based Auto-Discovery and specifies a format for NLRI encoding for IPv6 PE Address. This document also specifies a new type of attachment identifier to carry IPv6 address as AII in LDP FEC 0x81. This document updates RFC6074.

Status of this Memo

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

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 August 23, 2015.

Copyright and License Notice

Table of Contents

1 Introduction

[RFC6074] specifies a number of L2VPN provisioning models, and further specifies the semantic structure of the endpoint identifiers required by each model. It discusses the distribution of these identifiers by the discovery process, especially when discovery is based on the Border Gateway Protocol (BGP). It then specifies how the endpoint identifiers are carried in the two signaling protocols that are used to set up PWs, the Label Distribution Protocol (LDP), and the Layer 2 Tunneling Protocol version 3 (L2TPv3) [RFC6074]. This document updates Section 3.2.2.1 of RFC 6074 (BGP-Based Auto-Discovery) and specifies a format for NLRI encoding that allows to carry also an IPv6 PE Address.This document also specifies a new type of attachment identifier to carry IPv6 address as AII in LDP FEC 0x81. This gap in the specification of L2VPN in IPv6 only MPLS Network is also recognized in section 3.3.1 of [RFC7439].

1.1 Terminology

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

2 BGP NLRI Format for the IPv6 PE Address

Section 3.2.2.1 of [RFC6074] specifies the BGP advertisement for a particular VSI at a given PE will contain:

- an NLRI of AFI = L2VPN, SAFI = VPLS, encoded as RD:PE_addr
- a BGP next hop equal to the loopback address of the PE
- an Extended Community Attribute containing the VPLS-id
- an Extended Community Attribute containing one or more RTs.

The format for the NLRI encoding defined in Section 3.2.2.1 of [RFC6074] is:

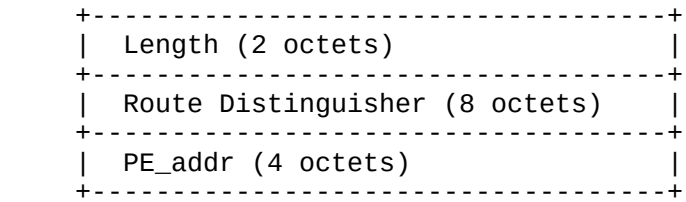


Figure 1: NLRI encoding in [RFC6074]

In this format the size of the PE_addr is defined as 4 octets which can carry only IPv4 addresses. In a situation where the route is originating from a BGP end point running on an IPv6 address, the PE_addr in the NLRI needs to carry that IPv6 address. The updated format for the NLRI encoding is depicted in Figure 2.

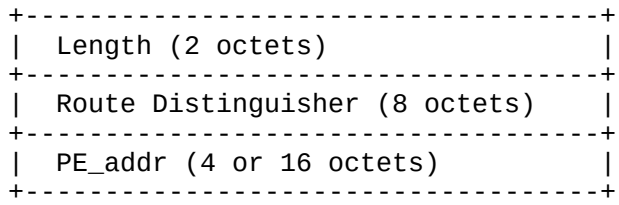


Figure 2: Updated NLRI encoding

The length field MUST contain the sum of the length of the Length field(2), the length of the Route Distinguisher (8) and the length of the 4 or 16 octet PE_addr field.

The type of the PE_addr can be derived by the receiving node by subtracting the fixed length of the Route Distinguisher and the Length field from the value of the received Length. An IPv4 PE_addr should be used to initiate adjacency of the underlying signaling protocol if it supports IPv4. An IPv6 PE_addr should be used to initiate adjacency of the underlying signaling protocol if it supports IPv6. (such as LDPv6)

3 Discussion on Route Distinguisher (RD) and Route Target (RT)

Note that RD and RT can be in format AS 2byte + 4 byte Assigned Number or IP 4 byte + 2 byte Assigned Number [RFC4364]. Just like RD or RT cannot carry 4 byte AS numbers , they also cannot utilize 16 byte IPv6 Address. Updates to RD and RT to operate in a pure IPv6 environment is outside the scope of this document.

4 Using IPv6 Remote PE address for signaling using LDP

Section 5.3.2 of [RFC4447] specifies the format of encoding for Generalized ID FEC Element (FEC 0x81)which is used for signaling in LDP. This document specifies a new type for AII carrying IPv6 address as TAII or SAII. (See Section 7)

An FEC 0x81 TLV MUST contain SAII and TAII of the same type i.e. either type 1 or type 2.

5 Interoperability in a mixed IPv4/IPv6 Network

If a VPLS instance is reachable though both IPv4 and IPv6 loopback in a PE node then the BGP instance(s) of that PE node MUST advertise the VPLS route using both NLRIs - one with IPv4 PE_addr and another with IPv6 PE_addr.

While signaling a TAII in type 2 format, the LDP implementation MUST use SAII also in type 2 format. The value of the SAII MAY be set from the IPv6 loopback address on which the BGP session is established.

While signaling a TAII type over an LDP session, on which it has already signaled with the other TAII type but with the same AGI, it SHOULD use the same label value in the Label Mapping for both TAII types.

On receiving an FEC 0x81 TLV in a Label Advertisement with a TAII type, the LDP implementation MAY lookup if on the same LDP session it has received a Label Mapping with the other TAII type but for the same AGI. If yes then it MUST store the Label Mapping but MAY choose not to install the label. If it chooses not to do the lookup stated above then it MUST install the received label.

If the LDP implementation chooses to do the lookup stated above during receipt of the Label Mapping, on receiving an FEC 0x81 TLV in a Label Withdraw with a TAII type, the LDP implementation MUST lookup if on the same LDP session it has received another Label Mapping with other TAII type but same AGI. If yes then it MUST install the stored Label Mapping and keep using that thereafter. (Along with taking necessary actions for processing the Label Withdraw as specified in [RFC5036])

6 Security Considerations

There is no additional security impact in addition to what is mentioned in [RFC6074].

7 IANA Considerations

This document requires a new AII type to be used in Generalized ID FEC (0x81). IANA already maintains a registry of name "Attachment Individual Identifier(AII) Type" specified by [RFC4446].

The following value is suggested for assignment:

AII Type	Length	Description
0x02	16	A 128 bit unsigned number local identifier.

8. Acknowledgments

Thanks to Mohamed Boucadair for his valuable suggestions.

9 References

9.1 Normative References

[RFC2119] S. Bradner,"Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6074] E. Rosen, B. Davie, "Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)", BCP 74, RFC 6074, January 2011, <http://www.rfc-editor.org/info/rfc6074>.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, April 2006, <http://www.rfc-editor.org/info/rfc4446>.

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

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007,<http://www.rfc-editor.org/info/rfc5036>.

[RFC7439] W. George, C. Pignataro,, "Gap Analysis for Operating IPv6-Only MPLS Networks", RFC 7439, January 2015,<http://www.rfc-editor.org/info/rfc7439>.

9.2 Informative References

[RFC4364] E. Rosen, "BGP/MPLS IP Virtual Private Networks (VPNs)", BCP 78, RFC 4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

Authors' Addresses

Avik Bhattacharya
Ixia
Infinity Building, Tower 2, Floor -4
Sector 5, Saltlake,
Kolkata, West Bengal, India - 700091.

EMail: abhattacharya@ixiacom.com

Apratim Mukherjee
Ixia
Infinity Building, Tower 2, Floor -4
Sector 5, Saltlake,
Kolkata, West Bengal, India - 700091.

EMail: amukherjee@ixiacom.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 03, 2016

S. Sivabalan
S. Boutros
Cisco Systems, Inc.
H. Shah
Ciena Corp.
S. Aldrin
Google Inc.
M. Venkatesan
Comcast.
November 03, 2015

MAC Address Withdrawal over Static Pseudowire
draft-ietf-pals-mpls-tp-mac-wd-03.txt

Abstract

This document specifies a mechanism to signal MAC address withdrawal notification using PW Associated Channel (ACH). Such notification is useful when statically provisioned PWs are deployed in VPLS/H-VPLS environment.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2016.

Table of Contents

1. Introduction

An LDP-based MAC Address Withdrawal Mechanism is specified in [RFC4762] to remove dynamically learned MAC addresses when the source of those addresses can no longer forward traffic. This is accomplished by sending an LDP Address Withdraw Message with a MAC List TLV containing the MAC addresses to be removed, to all other PEs over the LDP sessions. [RFC7361] describes an optimized MAC withdrawal mechanism which can be used to remove only the set of MAC addresses that need to be re-learned in H-VPLS networks. [RFC7361] also describes optimized MAC Withdrawal operations in PBB-VPLS networks.

A PW can be signaled via the LDP or can be statically provisioned. In the case of static PW, LDP based MAC withdrawal mechanism cannot be used. This is analogous to the problem and solution described in [RFC6478] where PW OAM message has been introduced to carry PW status TLV using in-band PW Associated Channel. In this document, we propose to use PW OAM message to withdraw MAC address(es) learned via static PW.

Thus, MAC withdraw signaling for static PW re-uses concepts of

- in-band signaling mechanisms used by static PW status signaling and

- MAC withdrawal mechanisms described by [RFC4762] and [RFC7361]

The MAC withdraw signaling is a best effort scheme. It is an attempt to optimize the network convergence by reducing blackholes caused by PW failover for protected PWs. The protocol defined in this document addresses possible loss of MAC withdraw signal due to network congestion, but do not assure the guarenteed delivery, as is the case for the LDP based MAC withdraw signaling. In the event that MAC withdraw signaling does not reach the intended target, the fallback to MAC re-learning due to bi-directional traffic or as a last resort to user configured MAC entries age out, will resume the traffic via new PW path. Such fallbacks would cause temporary blackout but does not render network permanently unusable.

2. Terminology

The following terminologies are used in this document:

ACK: Acknowledgement for MAC withdraw message.

LDP: Label Distribution Protocol.

MAC: Media Access Control.

PE: Provide Edge Node.

MPLS: Multi Protocol Label Switching.

PW: PseudoWire.

PW OAM: PW Operations, Administration and Maintenance.

TLV: Type, Length, and Value.

VPLS: Virtual Private LAN Services.

3. MAC Withdraw OAM Message

LDP provides a reliable packet transport for control plackets for dynamic PWs. This can be contrasted with static PWs which rely on re-transmission and acknowledgments (ACK) for reliable OAM packet delivery as described in [RFC6478]. The proposed solution for MAC withdrawal over static PW also relies on re-transmissions and ACKs. However, ACK is mandatory. A given MAC withdrawal notification is sent as a PW OAM message, and the sender re-transmits the message for a configured number of times in the absence of an ACK response for the sequence numbered message. The receiver removes the MAC address(es) for a given sequence number MAC withdraw signaling and sends the ACK response. The receipt of same or lower sequence number message is responded with ACK but does not cause removal of MAC addresses. A new TLV to carry the sequence number has been defined.

The format of the MAC address withdraw OAM message is shown in Figure 1. The MAC withdraw PW OAM message follows same guidelines used in [RFC6478], whereby first 4-bytes of OAM message header is followed by message specific field and a set of TLVs relevant for the message. Since the MAC withdrawal PW OAM message is not refreshed forever, a MAC address withdraw OAM message MUST contain a "Sequence Number TLV" otherwise the entire message is dropped. It MAY contain MAC Flush Parameter TLVs defined in [RFC7361] when static PWs are deployed in H-VPLS and PBB-VPLS scenarios. The first 2 bits of the sequence number TLV are reserved and MUST be set to 0 on transmit and ignored on receipt.

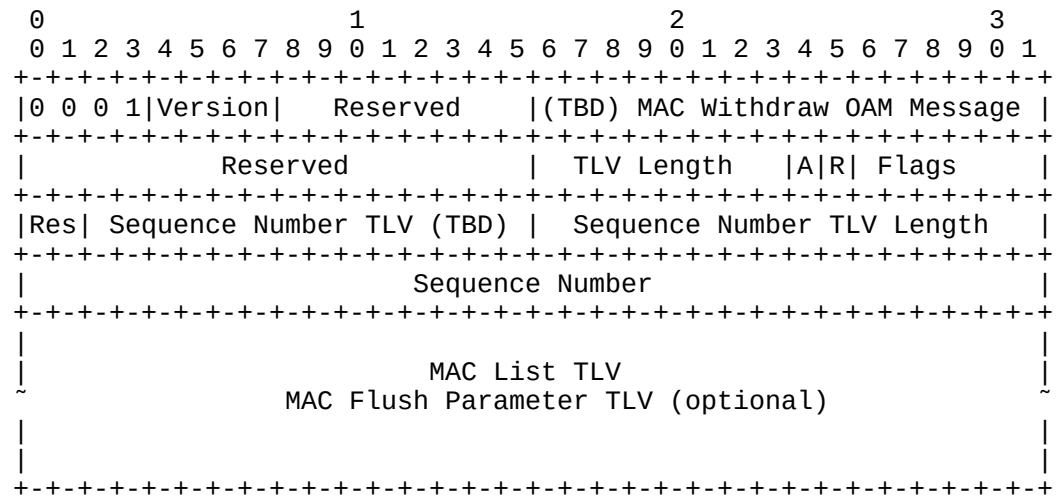


Figure 1: MAC Address Withdraw PW OAM Packet Format

In this section, MAC List TLV and MAC Flush Parameter TLV are collectively referred to as "MAC TLV(s)". The definition and processing rules of MAC List TLV are described by [RFC4762], and the corresponding rules of MAC Flush Parameter TLV are governed by [RFC7361].

"TLV Length" is the total length of all TLVs in the message, and "Sequence Number TLV Length" is the length of the sequence number field.

A single bit (called A-bit) is set by a receiver to acknowledge receipt and processing of a MAC Address Withdraw OAM message. In the acknowledge message, with A-bit set, MAC TLV(s) is/are excluded.

A single bit (called R-bit) is set to indicate if the sender is requesting reset of the sequence numbers. The sender sets this bit when the Pseudowire is restarted and has no local record of send and expected receive sequence number.

The Sequence number TLV MUST be the first TLV in the message.

The lack of reliable transport protocol for the in-band OAM necessitates a presence of sequencing and acknowledgement scheme so that the receiver can recognize newer message from retransmitted older messages. The [RFC4385] describes the details of sequence number handling which includes overflow detection for sequence number field of size 16-bits. This document leverages the same scheme with the two exemptions

- sequence number field is of size 32-bits
- overflow detection is simplified such that sequence number exceed 2,147,483,647 (0x7FFFFFFF) is considered overflow and reset to 1.

4. Operation

This section describes how the initial MAC withdraw OAM messages are sent and retransmitted, as well as how the messages are processed and retransmitted messages are identified.

4.1. Operation of Sender

Each PW is associated with a counter to keep track of the sequence number of the transmitted MAC withdrawal messages. Whenever a node sends a new set of MAC TLVs, it increments the transmitted sequence number counter, and include the new sequence number in the message. The transmit sequence number is initialized to 1 at the onset, after the wrap and after the sequence number reset request receipt. Hence the transmit sequence number is set to 2 in the first MAC withdraw message sent after the sequence number is initialized to 1.

The sender expects an ACK from the receiver within a time interval which we call "Retransmit Time" which can be either a default (1 second) or configured value. If the ACK does not arrive within the Retransmit Time, the sender retransmits the message with the same sequence number as the original message. The retransmission MUST be ceased when an ACK is received. In order to avaoid continuous retransmissions in the absence of acknowledgements, a method of suppressing retransmission MUST be implemented. A simple and well used approach is to cease retransmission after a small number of transmissions. A one second retransmission with two retries in the absence of an ACK response is RECOMMENDED. However, both the interval and the number of retries are a local matter which present no interworking issues and thus the operator MAY configure different values. Alternatively, an increasing backoff delay with a larger number of retries MAY be implemented to improve scaling issues. Whilst there are no interworking issues with any of these methods, the implementer must be mindful of not introducing network congestion and must take into account of decaying value of delayed MAC withdraw signaling against possible relearning due to bidirectional traffic or MAC age timeout.

During the period of retransmission, if a need to send a new MAC withdraw message with updated sequence number arises then retransmission of the older unacknowledged withdraw message MUST be suspended and retransmit time for the new sequence number MUST be initiated. In essence, sender engages in retransmission logic only for the latest send withdraw message for a given PW.

In the event that a Pseudowire was deleted and re-added or the router is restarted with configuration, the local node may lose information about the send sequence number of previous incarnation. This becomes problematic for the remote peer as it will continue to ignore the

received MAC withdraw messages with lower sequence numbers. In such cases, it is desirable to reset the sequence numbers at both ends of the Pseudowire. The 'R' reset bit is set in the first MAC withdraw to notify the remote peer to reset the send and receive sequence numbers. The 'R' bit must be cleared in subsequent MAC withdraw messages after the acknowledgement is received

4.2. Operation of Receiver

Each PW is associated with a register to keep track of the expected sequence number of the MAC withdrawal message and is initialized to 1. Whenever a MAC withdrawal message is received, and if the sequence number on the message is greater than the value in the register, the MAC address(es) contained in the MAC TLV(s) is/are removed, and the register is updated with the received sequence number. The receiver sends an ACK whose sequence number is the same as that in the received message.

If the sequence number in the received message is smaller than or equal to the value in the register, the MAC TLV(s) is/are not processed. However, an ACK with the received sequence number MUST be sent as a response. The receiver processes the ACK message as an acknowledgement for all the MAC withdraw messages sent up to the sequence number present in the ACK message and terminates retransmission.

The handling of the sequence number is described in section 3.

A MAC withdraw message with 'R' bit set MUST be processed by resetting the send and receive sequence number first. The rest of MAC withdraw message processing is performed as described above. The acknowledgement is sent with 'R' bit cleared.

5. Security Consideration

The security measures described in [RFC4447], [RFC5085], and [RFC6073] are adequate for the proposed mechanism.

6. IANA Considerations

6.1. MPLS G-Ach type

This document requests IANA to assign new channel type (requested value 0x0028) from the registry named "MPLS Generalized Associated Channel (G-ACh) Types (including Pseudwire Associated Channel Types)". The description of the new channel type is "MAC Withdraw OAM Message". [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/g-ach-parameters/g-ach-parameters.xhtml]. The channel type value of 0x0028

is requested as it is used in implementations that are deployed in the field.

6.2. Sequence Number TLV

This document requests IANA to assign a new TLV Type (requested value 0x0001) from the existing LDP "TLV Type Name Space" registry. The description for the new TLV Type is "Sequence Number TLV".

[Note to IANA TO BE REMOVED BY THE RFC EDITOR: This registration should take place at the following location: http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml]. In an earlier revision of this draft, we created a new sub-TLV registry with one entry, a new "Sequence Number TLV" with the value 0x0001. This has been implemented by several vendors. The IESG proposed that rathen than create a new sub-TLV registry, we just allocate a new code point from the existing LDP "TLV Type Name Space" registry. In this registry, the value 0x0001 is available for allocation by standards action, so we request this code point for the new "Sequence Number" TLV type to avoid needing to change the existing implementations.

7. References

7.1. Normative References

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC4447] Martini, L., Rosen, E., El-Aawar, N., Smith, T., and G. Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", RFC 4447, April 2006.

[RFC4762] Lasserre, M. and V. Kompella, "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, January 2007.

[RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6073] Martini, L., Metz, C., Nadeau, T., Bocci, M., and M. Aissaoui, "Segmented Pseudowire", RFC 6073, January 2011.

[RFC6478] Martini, L., Swallow, G., Heron, G., and M. Bocci, "Pseudowire Status for Static Pseudowires", RFC 6478, May 2012.

[RFC7361] Dutta, P., Balus, F., Stokes, O., Calvignac, G., and D. Fedyk, "LDP Extensions for Optimized MAC Address Withdrawal in a Hierarchical Virtual Private LAN Service (H-VPLS)", RFC 7361, September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

Authors' Addresses

Siva Sivabalan
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: msiva@cisco.com

Sami Boutros
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
US

Email: sboutros@cisco.com

Himanshu Shah
Ciena Corp.
3939 North First Street
San Jose, CA 95134
US

Email: hshah@ciena.com

Sam Aldrin
Google Inc.

Email: aldrin.ietf@gmail.com

Mannan Venkatesan
Comcast.
1800 Bishop Gate Blvd
Mount Laurel, NJ 08075
US

Email: mannan_venkatesan@cable.comcast.com

Internet Engineering Task Force
Internet Draft
Intended status: Internet Standard
Expires: January 5, 2017
Obsoletes: 6723, 4447

Luca Martini Ed.
Giles Heron Ed.
Cisco

July 5, 2016

Pseudowire Setup and Maintenance using the Label Distribution Protocol

draft-ietf-pals-rfc4447bis-05.txt

Status of this Memo

Abstract

Layer 2 services (such as Frame Relay, Asynchronous Transfer Mode, and Ethernet) can be "emulated" over an MPLS backbone by encapsulating the Layer 2 Protocol Data Units (PDU) and then transmitting them over "pseudowires". It is also possible to use pseudowires to provide low-rate Time Division Multiplexed and

Synchronous Optical NETworking circuit emulation over an MPLS-enabled network. This document specifies a protocol for establishing and maintaining the pseudowires, using extensions to the Label Distribution Protocol (LDP). Procedures for encapsulating Layer 2 PDUs are specified in a set of companion documents.

This document has been written to address errata in a previous version of this standard.

Table of Contents

1. Introduction

[RFC4619], [RFC4717], [RFC4618], and [RFC4448] explain how to encapsulate a Layer 2 Protocol Data Unit (PDU) for transmission over an MPLS-enabled network. Those documents specify that a "pseudowire header", consisting of a demultiplexor field, will be prepended to the encapsulated PDU. The pseudowire demultiplexor field is prepended before transmitting a packet on a pseudowire. When the packet arrives at the remote endpoint of the pseudowire, the demultiplexor is what enables the receiver to identify the particular pseudowire on which the packet has arrived. To transmit the packet from one pseudowire endpoint to another, the packet may need to travel through a "Packet Switched Network (PSN) tunnel"; this will require that an additional header be prepended to the packet.

Accompanying documents [RFC4842], [RFC4553] specify methods for transporting time-division multiplexing (TDM) digital signals (TDM circuit emulation) over a packet-oriented MPLS-enabled network. The transmission system for circuit-oriented TDM signals is the Synchronous Optical Network [ANSI] (SONET)/Synchronous Digital Hierarchy (SDH) [ITUG]. To support TDM traffic, which includes voice, data, and private leased-line service, the pseudowires must emulate the circuit characteristics of SONET/SDH payloads. The TDM signals and payloads are encapsulated for transmission over pseudowires. A pseudowire demultiplexor and a PSN tunnel header is prepended to this encapsulation.

[RFC4553] describes methods for transporting low-rate time-division multiplexing (TDM) digital signals (TDM circuit emulation) over PSNs, while [RFC4842] similarly describes transport of high-rate TDM (SONET/SDH). To support TDM traffic, the pseudowires must emulate the circuit characteristics of the original T1, E1, T3, E3, SONET, or SDH signals. [RFC4553] does this by encapsulating an arbitrary but constant amount of the TDM data in each packet, and the other methods encapsulate TDM structures.

In this document, we specify the use of the MPLS Label Distribution Protocol, LDP [RFC5036], as a protocol for setting up and maintaining the pseudowires. In particular, we define new TLVs, FEC elements, parameters, and codes for LDP, which enable LDP to identify pseudowires and to signal attributes of pseudowires. We specify how a pseudowire endpoint uses these TLVs in LDP to bind a demultiplexor field value to a pseudowire, and how it informs the remote endpoint of the binding. We also specify procedures for reporting pseudowire status changes, for passing additional information about the pseudowire as needed, and for releasing the bindings. These procedures are intended to be independent of the underlying version of IP used for LDP signaling.

In the protocol specified herein, the pseudowire demultiplexor field is an MPLS label. Thus, the packets that are transmitted from one end of the pseudowire to the other are MPLS packets, which must be transmitted through an MPLS tunnel. However, if the pseudowire endpoints are immediately adjacent and penultimate hop popping behavior is in use, the MPLS tunnel may not be necessary. Any sort of PSN tunnel can be used, as long as it is possible to transmit MPLS packets through it. The PSN tunnel can itself be an MPLS LSP, or any other sort of tunnel that can carry MPLS packets. Procedures for setting up and maintaining the MPLS tunnels are outside the scope of this document.

This document deals only with the setup and maintenance of point-to-point pseudowires. Neither point-to-multipoint nor multipoint-to-point pseudowires are discussed.

QoS-related issues are not discussed in this document.

The following two figures describe the reference models that are derived from [RFC3985] to support the PW emulated services.

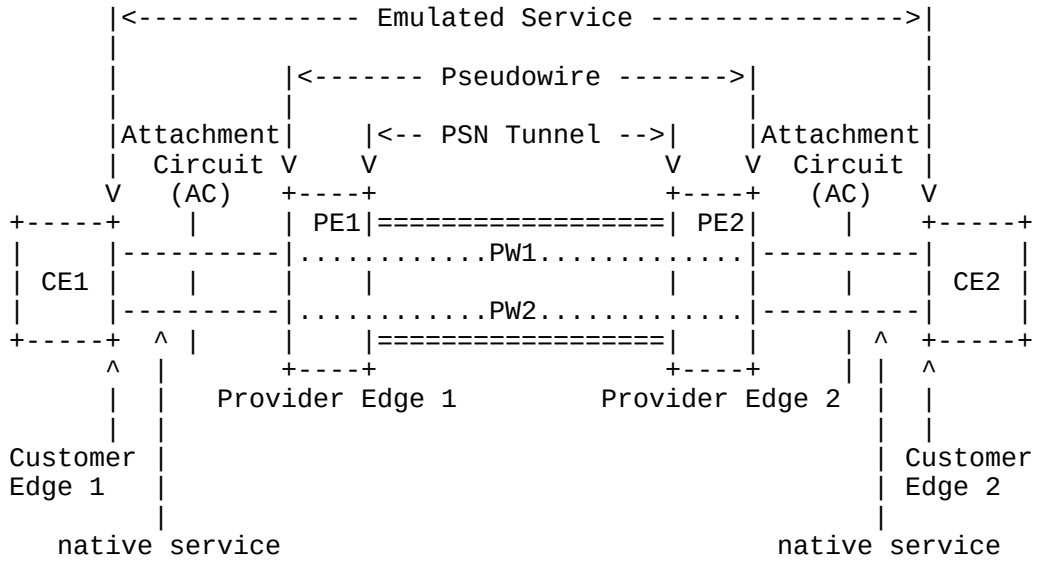


Figure 1: PWE3 Reference Model

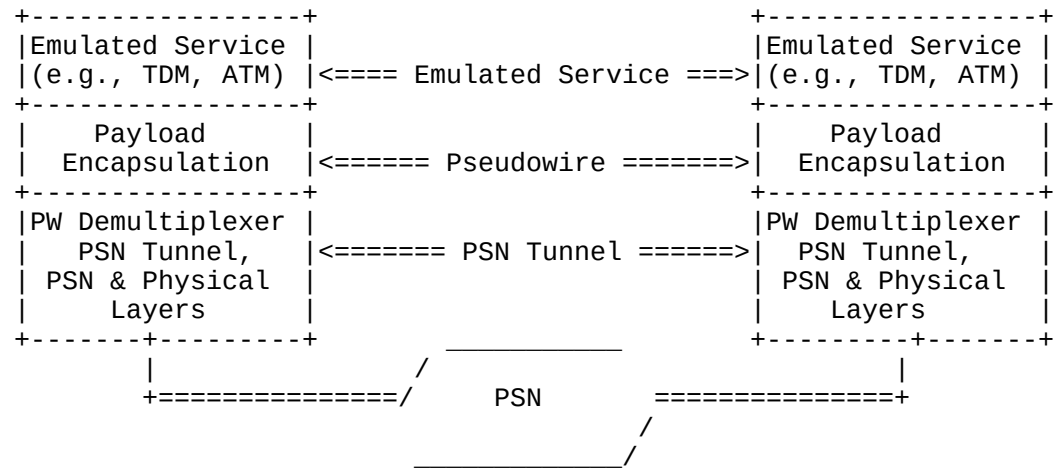


Figure 2: PWE3 Protocol Stack Reference Model

For the purpose of this document, PE1 will be defined as the ingress router, and PE2 as the egress router. A layer 2 PDU will be received at PE1, encapsulated at PE1, transported and decapsulated at PE2, and transmitted out of PE2.

2. Changes from RFC4447

The changes in this document are mostly minor fixes to spelling and grammar, or clarifications to the text, which were either noted as errata to [RFC4447] or found by the editors.

Additionally a new section (7.3) on control-word renegotiation by label request message has been added, obsoleting [RFC6723]. The diagram of C-bit handling procedures has also been removed. A note has been added in section 6.3.2 to clarify that the C-bit is part of the FEC.

A reference has also been added to [RFC7358] indicating the use of downstream unsolicited mode to distribute PW FEC label bindings, independent of the negotiated label advertisement mode of the LDP session.

3. Specification of Requirements

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].

4. The Pseudowire Label

Suppose that it is desired to transport Layer 2 PDUs from ingress LSR PE1 to egress LSR PE2, across an intervening MPLS-enabled network. We assume that there is an MPLS tunnel from PE1 to PE2. That is, we assume that PE1 can cause a packet to be delivered to PE2 by encapsulating the packet in an "MPLS tunnel header" and sending the result to one of its adjacencies. The MPLS tunnel is an MPLS Label Switched Path (LSP); thus, putting on an MPLS tunnel encapsulation is a matter of pushing on an MPLS label.

We presuppose that a large number of pseudowires can be carried through a single MPLS tunnel. Thus it is never necessary to maintain state in the network core for individual pseudowires. We do not presuppose that the MPLS tunnels are point to point; although the pseudowires are point to point, the MPLS tunnels may be multipoint to point. We do not presuppose that PE2 will even be able to determine the MPLS tunnel through which a received packet was transmitted. (For example, if the MPLS tunnel is an LSP and penultimate hop popping is used, when the packet arrives at PE2, it will contain no information identifying the tunnel.)

When PE2 receives a packet over a pseudowire, it must be able to determine that the packet was in fact received over a pseudowire, and it must be able to associate that packet with a particular pseudowire. PE2 is able to do this by examining the MPLS label that serves as the pseudowire demultiplexor field shown in Figure 2. Call this label the "PW label".

When PE1 sends a Layer 2 PDU to PE2, it creates an MPLS packet by adding the PW label to the packet, thus creating the first entry of the label stack. If the PSN tunnel is an MPLS LSP, the PE1 pushes another label (the tunnel label) onto the packet as the second entry of the label stack. The PW label is not visible again until the MPLS packet reaches PE2. PE2’s disposition of the packet is based on the PW label.

If the payload of the MPLS packet is, for example, an ATM AAL5 PDU, the PW label will generally correspond to a particular ATM VC at PE2. That is, PE2 needs to be able to infer from the PW label the outgoing interface and the VPI/VCI value for the AAL5 PDU. If the payload is a Frame Relay PDU, then PE2 needs to be able to infer from the PW

label the outgoing interface and the DLCI value. If the payload is an Ethernet frame, then PE2 needs to be able to infer from the PW label the outgoing interface, and perhaps the VLAN identifier. This process is uni-directional and will be repeated independently for bi-directional operation. When using the PWid FEC Element, it is REQUIRED that the same PW ID and PW type be assigned for a given circuit in both directions. The group ID (see below) MUST NOT be required to match in both directions. The transported frame MAY be modified when it reaches the egress router. If the header of the transported Layer 2 frame is modified, this MUST be done at the egress LSR only. Note that the PW label must always be at the bottom of the packet's label stack, and labels MUST be allocated from the per-platform label space.

This document does not specify a method for distributing the MPLS tunnel label or any other labels that may appear above the PW label on the stack. Any acceptable method of MPLS label distribution will do. This document specifies a protocol for assigning and distributing the PW label. This protocol is LDP, extended as specified in the remainder of this document. An LDP session must be set up between the pseudowire endpoints. LDP MUST exchange PW FEC label bindings in downstream unsolicited mode, independent of the negotiated label advertisement mode of the LDP session according to the specifications in specified in [RFC7358]. LDP's "liberal label retention" mode SHOULD be used. However all the LDP procedures that are specified in [RFC5036], and that are also applicable to this protocol specification MUST be implemented.

This document requires that a receiving LSR MUST respond to a Label Request message with either a Label Mapping for the requested label or with a Notification message that indicates why it cannot satisfy the request. These procedures are specified in [RFC5036] section 3.5.7 "Label Mapping Message", and 3.5.8 "Label Request Message". Note that sending these responses is a stricter requirement than is specified in [RFC5036] but these response messages are REQUIRED to ensure correct operation of this protocol.

In addition to the protocol specified herein, static assignment of PW labels may be used, and implementations of this protocol SHOULD provide support for static assignment. PW encapsulation is always symmetrical in both directions of traffic along a specific PW, whether the PW uses an LDP control plane or not.

This document specifies all the procedures necessary to set up and maintain the pseudowires needed to support "unswitched" point to point services, where each endpoint of the pseudowire is provisioned with the identity of the other endpoint. There are also protocol mechanisms specified herein that can be used to support switched

services and other provisioning models. However, the use of the protocol mechanisms to support those other models and services is not described in this document.

5. Details Specific to Particular Emulated Services

5.1. IP Layer 2 Transport

This mode carries IP packets over a pseudowire. The encapsulation used is according to [RFC3032]. The PW control word MAY be inserted between the MPLS label stack and the IP payload. The encapsulation of the IP packets for forwarding on the attachment circuit is implementation specific, is part of the native service processing (NSP) function [RFC3985], and is outside the scope of this document.

6. LDP

The PW label bindings are distributed using the LDP downstream unsolicited mode described in [RFC5036]. The PEs will establish an LDP session using the Extended Discovery mechanism described in [LDP, sectionn 2.4.2 and 2.5].

An LDP Label Mapping message contains an FEC TLV, a Label TLV, and zero or more optional parameter TLVs.

The FEC TLV is used to indicate the meaning of the label. In the current context, the FEC TLV would be used to identify the particular pseudowire that a particular label is bound to. In this specification, we define two new FEC TLVs to be used for identifying pseudowires. When setting up a particular pseudowire, only one of these FEC TLVs is used. The one to be used will depend on the particular service being emulated and on the particular provisioning model being supported.

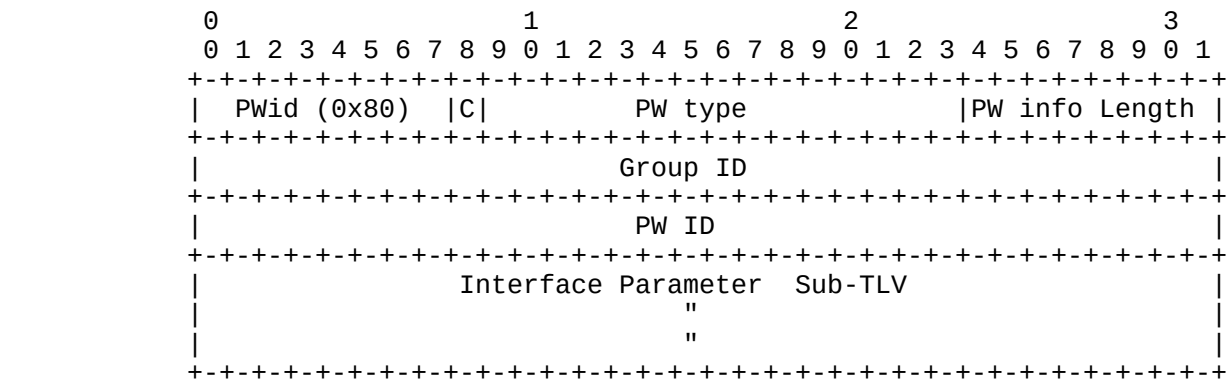
LDP allows each FEC TLV to consist of a set of FEC elements. For setting up and maintaining pseudowires, however, each FEC TLV MUST contain exactly one FEC element.

The LDP base specification has several kinds of label TLVs, including the Generic Label TLV, as specified in [RFC5036], section 3.4.2.1. For setting up and maintaining pseudowires, the Generic Label TLV MUST be used.

6.1. The PWid FEC Element

The PWid FEC element may be used whenever both pseudowire endpoints have been provisioned with the same 32-bit identifier for the pseudowire.

For this purpose, a new type of FEC element is defined. The FEC element type is 0x80 and is defined as follows:



- PW type

  A 15 bit quantity containing a value that represents the type of PW. Assigned Values are specified in "IANA Allocations for pseudo Wire Edge to Edge Emulation (PWE3)" [RFC4446].

- Control word bit (C)

  The bit (C) is used to flag the presence of a control word as follows:

  C = 1 control word present on this PW.
  C = 0 no control word present on this PW.

  Please see the section "Control Word" for further explanation.

- PW information length

  Length of the PW ID field and the interface parameters sub-TLV in octets. If this value is 0, then it references all PWs using the specified group ID, and there is no PW ID present, nor are there any interface parameter sub-TLVs.

- Group ID

  An arbitrary 32 bit value which represents a group of PWs that is used to create groups in the PW space. The group ID is intended to be used as a port index, or a virtual tunnel index. To simplify configuration a particular PW ID at ingress could be part of a Group ID assigned to the virtual tunnel for transport to the egress router. The Group ID is very useful for sending wild card label withdrawals, or PW wild card status notification messages to remote PEs upon physical port failure.

- PW ID

  A non-zero 32-bit connection ID that together with the PW type identifies a particular PW. Note that the PW ID and the PW type MUST be the same at both endpoints.

- Interface Parameter Sub-TLV

  This variable length TLV is used to provide interface specific parameters, such as attachment circuit MTU.

  Note that as the "interface parameter sub-TLV" is part of the FEC, the rules of LDP make it impossible to change the interface parameters once the pseudowire has been set up. Thus the interface parameters field must not be used to pass information, such as status information, that may change during the life of the pseudowire. Optional parameter TLVs should be used for that purpose.

Using the PWid FEC, each of the two pseudowire endpoints independently initiates the setup of a unidirectional LSP. An outgoing LSP and an incoming LSP are bound together into a single pseudowire if they have the same PW ID and PW type.

6.2. The Generalized PWid FEC Element

The PWid FEC element can be used if a unique 32-bit value has been assigned to the PW, and if each endpoint has been provisioned with that value. The Generalized PWid FEC element requires that the PW endpoints be uniquely identified; the PW itself is identified as a pair of endpoints. In addition, the endpoint identifiers are structured to support applications where the identity of the remote endpoints needs to be auto-discovered rather than statically configured.

The "Generalized PWid FEC Element" is FEC type 0x81.

The Generalized PWid FEC Element does not contain anything corresponding to the "Group ID" of the PWid FEC element. The functionality of the "Group ID" is provided by a separate optional LDP TLV, the "PW Group ID TLV", described below. The Interface Parameters field of the PWid FEC element is also absent; its functionality is replaced by the optional Interface Parameters TLV, described below.

6.2.1. Attachment Identifiers

As discussed in [RFC3985], a pseudowire can be thought of as connecting two "forwarders". The protocol used to set up a pseudowire must allow the forwarder at one end of a pseudowire to identify the forwarder at the other end. We use the term "attachment identifier", or "AI", to refer to the field that the protocol uses to identify the forwarders. In the PWid FEC, the PWid field serves as the AI. In this section, we specify a more general form of AI that is structured and of variable length.

Every Forwarder in a PE must be associated with an Attachment Identifier (AI), either through configuration or through some algorithm. The Attachment Identifier must be unique in the context of the PE router in which the Forwarder resides. The combination <PE router IP address, AI> must be globally unique.

It is frequently convenient to regard a set of Forwarders as being members of a particular "group", where PWs may only be set up among members of a group. In such cases, it is convenient to identify the Forwarders relative to the group, so that an Attachment Identifier would consist of an Attachment Group Identifier (AGI) plus an Attachment Individual Identifier (AII).

An Attachment Group Identifier may be thought of as a VPN-id, or a VLAN identifier, some attribute that is shared by all the Attachment PWs (or pools thereof) that are allowed to be connected.

The details of how to construct the AGI and AII fields identifying the pseudowire endpoints are outside the scope of this specification. Different pseudowire applications, and different provisioning models, will require different sorts of AGI and AII fields. The specification of each such application and/or model must include the rules for constructing the AGI and AII fields.

As previously discussed, a (bidirectional) pseudowire consists of a pair of unidirectional LSPs, one in each direction. If a particular pseudowire connects PE1 with PE2, the PW direction from PE1 to PE2 can be identified as:

<PE1, <AGI, AII1>, PE2, <AGI, AII2>>,

and the PW direction from PE2 to PE1 can be identified by:

<PE2, <AGI, AII2>, PE1, <AGI, AII1>>.

Note that the AGI must be the same at both endpoints, but the AII will in general be different at each endpoint. Thus, from the perspective of a particular PE, each pseudowire has a local or "Source AII", and a remote or "Target AII". The pseudowire setup protocol can carry all three of these quantities:

- Attachment Group Identifier (AGI).

- Source Attachment Individual Identifier (SAII)

- Target Attachment Individual Identifier (TAII)

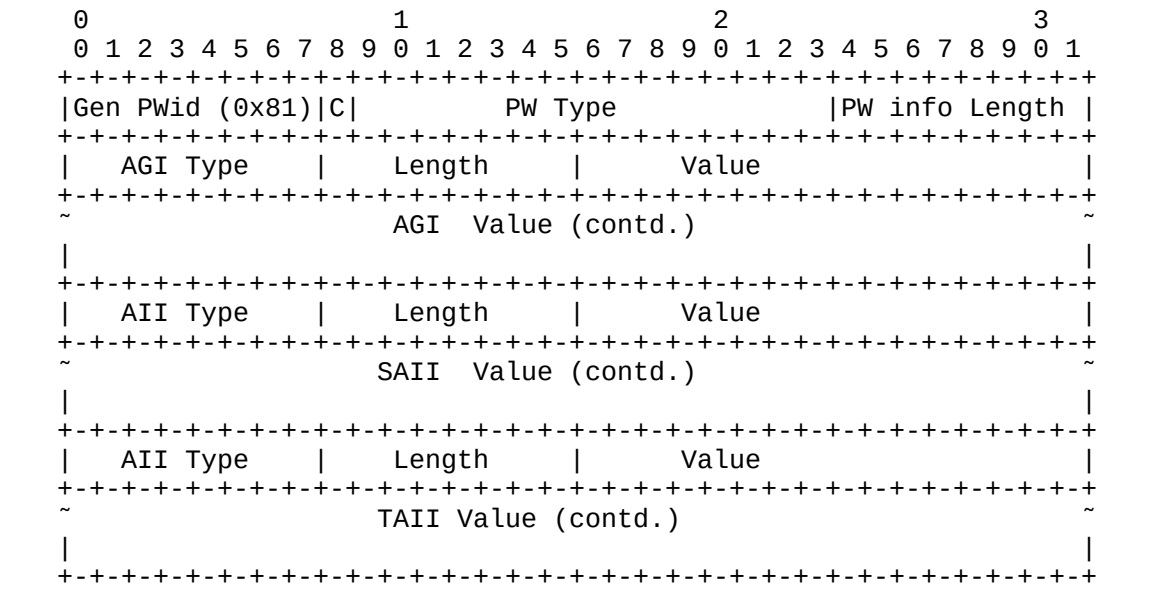
If the AGI is non-null, then the Source AI (SAI) consists of the AGI together with the SAII, and the Target AI (TAI) consists of the TAII together with the AGI. If the AGI is null, then the SAII and TAII are the SAI and TAI, respectively.

The interpretation of the SAI and TAI is a local matter at the respective endpoint.

The association of two unidirectional LSPs into a single bidirectional pseudowire depends on the SAI and the TAI. Each application and/or provisioning model that uses the Generalized PWid FEC element must specify the rules for performing this association.

6.2.2. Encoding the Generalized PWid FEC Element

FEC element type 0x81 is used. The FEC element is encoded as follows:



This document does not specify the AII and AGI type field values; specification of the type field values to be used for a particular application is part of the specification of that application. IANA has assigned these values using the method defined in the [RFC4446] document.

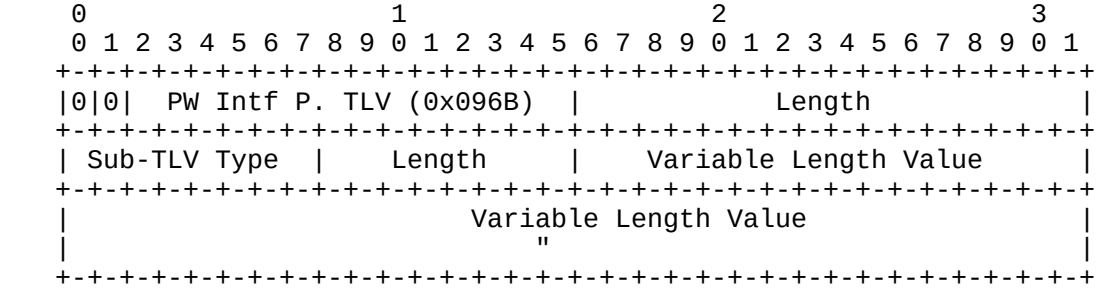
The SAII, TAII, and AGI are simply carried as octet strings. The length byte specifies the size of the Value field. The null string can be sent by setting the length byte to 0. If a particular application does not need all three of these sub-elements, it MUST send all the sub-elements but set the length to 0 for the unused sub-elements.

The PW information length field contains the length of the SAII, TAII, and AGI, combined in octets. If this value is 0, then it references all PWs using the specific grouping ID (specified in the PW Group ID TLV). In this case, there are no other FEC element fields (AGI, SAII, etc.) present, nor any interface parameters TLVs.

Note that the interpretation of a particular field as AGI, SAII, or TAII depends on the order of its occurrence. The type field identifies the type of the AGI, SAII, or TAII. When comparing two occurrences of an AGI (or SAII or TAII), the two occurrences are considered identical if the type, length, and value fields of one are identical, respectively, to those of the other.

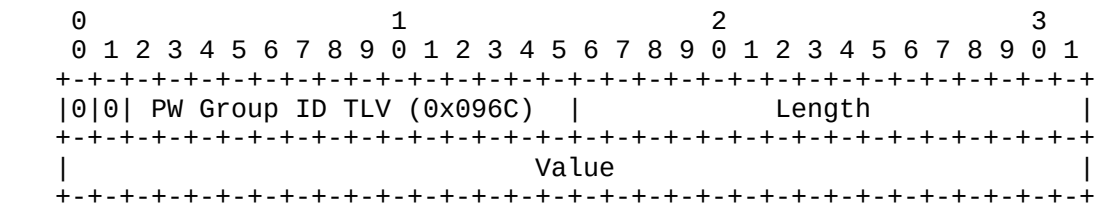
6.2.2.1. Interface Parameters TLV

This TLV MUST only be used when sending the Generalized PW FEC. It specifies interface-specific parameters. Specific parameters, when applicable, MUST be used to validate that the PEs and the ingress and egress ports at the edges of the circuit have the necessary capabilities to interoperate with each other.



A more detailed description of this field can be found in the section "Interface Parameters Sub-TLV", below.

6.2.2.2. PW Group ID TLV



The PW Group ID is an arbitrary 32-bit value that represents an arbitrary group of PWs. It is used to create group PWs; for example, a PW Grouping ID can be used as a port index and assigned to all PWs that lead to that port. Use of the PW Group ID enables a PE to send "wild card" label withdrawals, or "wild card" status notification messages, to remote PEs upon physical port failure.

Note Well: The PW Group ID is different from and has no relation to, the Attachment Group Identifier.

The PW Group ID TLV is not part of the FEC and will not be advertised except in the PW FEC advertisement. The advertising PE MAY use the wild card withdraw semantics, but the remote PEs MUST implement support for wild card messages. This TLV MUST only be used when sending the Generalized PW ID FEC.

To issue a wild card command (status or withdraw):

- Set the PW Info Length to 0 in the Generalized PWid FEC Element.
- Send only the PW Group ID TLV with the FEC (no AGI/SAII/TAII is sent).

6.2.3. Signaling Procedures

In order for PE1 to begin signaling PE2, PE1 must know the address of the remote PE2, and a TAI. This information may have been configured at PE1, or it may have been learned dynamically via some autodiscovery procedure.

The egress PE (PE1), which has knowledge of the ingress PE, initiates the setup by sending a Label Mapping Message to the ingress PE (PE2). The Label Mapping message contains the FEC TLV, carrying the Generalized PWid FEC Element (type 0x81). The Generalized PWid FEC element contains the AGI, SAII, and TAII information.

Next, when PE2 receives such a Label Mapping message, PE2 interprets the message as a request to set up a PW whose endpoint (at PE2) is the Forwarder identified by the TAI. From the perspective of the signaling protocol, exactly how PE2 maps AIs to Forwarders is a local matter. In some Virtual Private Wire Services (VPWS) provisioning models, the TAI might, for example, be a string that identifies a particular Attachment Circuit, such as "ATM3VPI4VCI5", or it might, for example, be a string, such as "Fred", that is associated by configuration with a particular Attachment Circuit. In VPLS, the AGI could be a VPN-id, identifying a particular VPLS instance.

If PE2 cannot map the TAI to one of its Forwarders, then PE2 sends a Label Release message to PE1, with a Status Code of "Unassigned/Unrecognized TAI", and the processing of the Label Mapping message is complete.

The FEC TLV sent in a Label Release message is the same as the FEC TLV received in the Label Mapping being released (but without the interface parameter TLV). More generally, the FEC TLV is the same in all LDP messages relating to the same PW. In a Label Release this means that the SAII is the remote peer’s AII and the TAII is the sender’s local AII.

If the Label Mapping Message has a valid TAI, PE2 must decide whether to accept it. The procedures for so deciding will depend on the particular type of Forwarder identified by the TAI. Of course, the Label Mapping message may be rejected due to standard LDP error conditions as detailed in [RFC5036].

If PE2 decides to accept the Label Mapping message, then it has to make sure that a PW LSP is set up in the opposite (PE1-->PE2) direction. If it has already signaled for the corresponding PW LSP in that direction, nothing more needs to be done. Otherwise, it must initiate such signaling by sending a Label Mapping message to PE1. This is very similar to the Label Mapping message PE2 received, but the SAI and TAI are reversed.

Thus, a bidirectional PW consists of two LSPs, where the FEC of one has the SAII and TAII reversed with respect to the FEC of the other.

6.3. Signaling of Pseudowire Status

6.3.1. Use of Label Mapping Messages

The PEs MUST send Label Mapping Messages to their peers as soon as the PW is configured and administratively enabled, regardless of the attachment circuit state. The PW label should not be withdrawn unless the operator administratively configures the pseudowire down (or the PW configuration is deleted entirely). Using the procedures outlined in this section, a simple label withdraw method MAY also be supported as a legacy means of signaling PW status and AC status. In any case, if the label-to-PW binding is not available the PW MUST be considered in the down state.

Once the PW status negotiation procedures are completed and if they result in the use of the label withdraw method for PW status communication, and this method is not supported by one of the PEs, then that PE must send a Label Release Message to its peer with the following error:

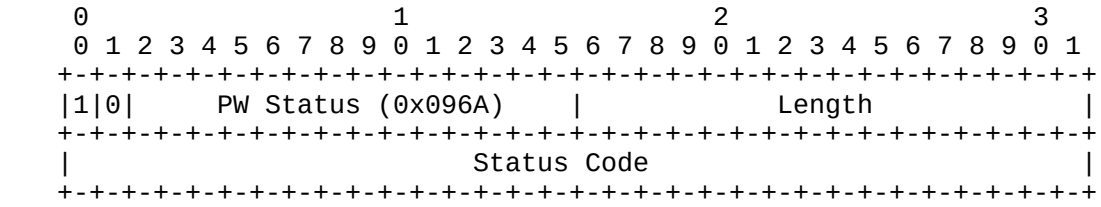
"Label Withdraw PW Status Method Not Supported"

If the label withdraw method for PW status communication is selected for the PW, it will result in the Label Mapping Message being advertised only if the attachment circuit is active. The PW status signaling procedures described in this section MUST be fully implemented.

6.3.2. Signaling PW Status

The PE devices use an LDP TLV to indicate status to their remote peers. This PW Status TLV contains more information than the alternative simple Label Withdraw message.

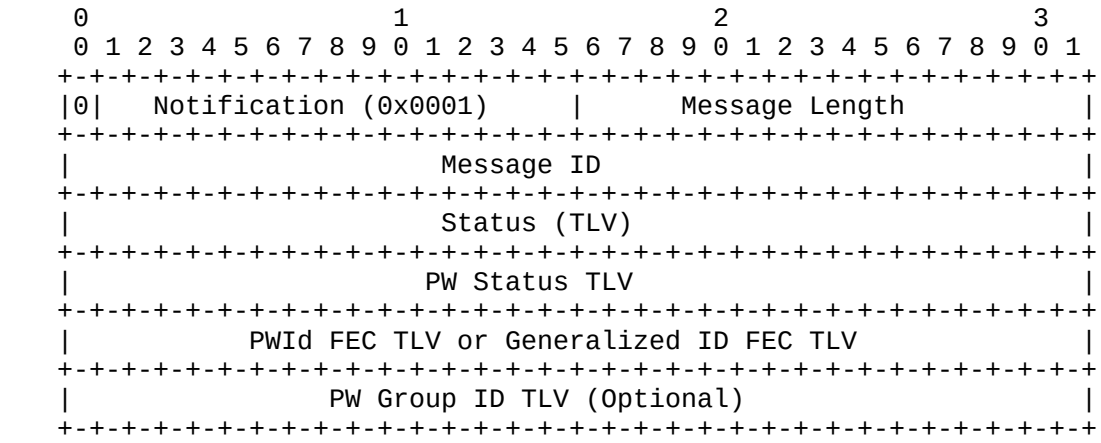
The format of the PW Status TLV is:



The status code is a 4 octet bit field as specified in the PW IANA Allocations document [RFC4446]. The length specifies the length of the Status Code field in octets (equal to 4).

Each bit in the status code field can be set individually to indicate more than a single failure at once. Each fault can be cleared by sending an appropriate Notification message in which the respective bit is cleared. The presence of the lowest bit (PW Not Forwarding) acts only as a generic failure indication when there is a link-down event for which none of the other bits apply.

The Status TLV is transported to the remote PW peer via the LDP Notification message as described in [RFC5036]. The format of the Notification Message for carrying the PW Status is as follows:



The Status TLV status code is set to 0x00000028, "PW status", to indicate that PW status follows. Since this notification does not refer to any particular message, the Message Id field is set to 0.

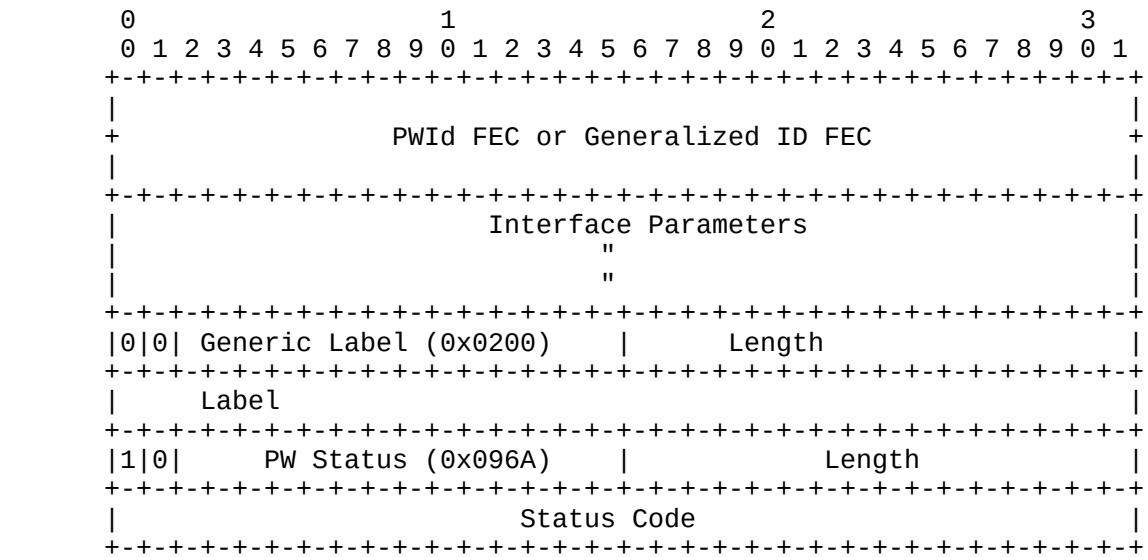
The PW FEC TLV SHOULD NOT include the interface parameter sub-TLVs, as they are ignored in the context of this message. However the PW

FEC TLV MUST include the C-bit, where aplicable, as it is part of the FEC. When a PE's attachment circuit encounters an error, use of the PW Notification Message allows the PE to send a single "wild card" status message, using a PW FEC TLV with only the group ID set, to denote this change in status for all affected PW connections. This status message contains either the PW FEC TLV with only the group ID set, or else it contains the Generalized FEC TLV with only the PW Group ID TLV.

As mentioned above, the Group ID field of the PWid FEC element, or the PW Grouping ID TLV used with the Generalized PWid FEC element, can be used to send a status notification for all arbitrary sets of PWs. This procedure is OPTIONAL, and if it is implemented, the LDP Notification message should be as follows: If the PWid FEC element is used, the PW information length field is set to 0, the PW ID field is not present, and the interface parameter sub-TLVs are not present. If the Generalized FEC element is used, the AGI, SAII, and TAII are not present, the PW information length field is set to 0, the PW Group ID TLV is included, and the Interface Parameters TLV is omitted. For the purpose of this document, this is called the "wild card PW status notification procedure", and all PEs implementing this design are REQUIRED to accept such a notification message but are not required to send it.

6.3.3. Pseudowire Status Negotiation Procedures

When a PW is first set up, the PEs MUST attempt to negotiate the usage of the PW status TLV. This is accomplished as follows: A PE that supports the PW Status TLV MUST include it in the initial Label Mapping message following the PW FEC and the interface parameter sub-TLVs. The PW Status TLV will then be used for the lifetime of the pseudowire. This is shown in the following diagram:



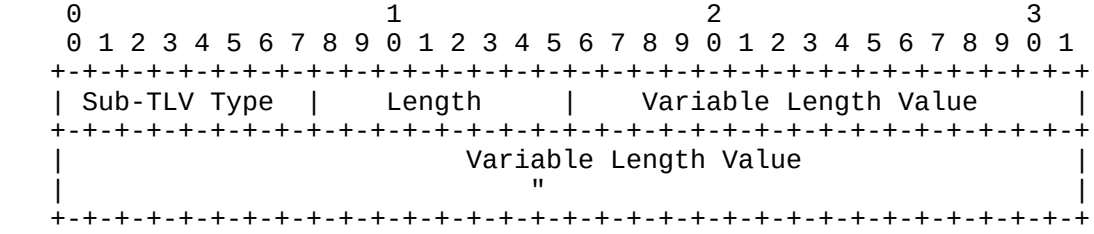
If a PW Status TLV is included in the initial Label Mapping message for a PW, then if the Label Mapping message from the remote PE for that PW does not include a PW status TLV, or if the remote PE does not support the PW Status TLV, the PW will revert to the label withdraw method of signaling PW status. Note that if the PW Status TLV is not supported by the remote peer, the peer will automatically ignore it, since the I (ignore) bit is set in the TLV. The PW Status TLV, therefore, will not be present in the corresponding FEC advertisement from the remote LDP peer, which results in exactly the above behavior.

If the PW Status TLV is not present following the FEC TLV in the initial PW Label Mapping message received by a PE, then the PW Status TLV will not be used, and both PEs supporting the pseudowire will revert to the label withdraw procedure for signaling status changes.

If the negotiation process results in the usage of the PW status TLV, then the actual PW status is determined by the PW status TLV that was sent within the initial PW Label Mapping message. Subsequent updates of PW status are conveyed through the notification message.

6.4. Interface Parameters Sub-TLV

This field specifies interface-specific parameters. When applicable, it MUST be used to validate that the PEs and the ingress and egress ports at the edges of the circuit have the necessary capabilities to interoperate with each other. The field structure is defined as follows:



The interface parameter sub-TLV type values are specified in "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)" [RFC4446].

The Length field is defined as the length of the interface parameter including the parameter id and length field itself. Processing of the interface parameters should continue when unknown interface parameters are encountered, and they MUST be silently ignored.

- Interface MTU sub-TLV type

  A 2 octet value indicating the MTU in octets. This is the Maximum Transmission Unit, excluding encapsulation overhead, of the egress packet interface that will be transmitting the decapsulated PDU that is received from the MPLS-enabled network. This parameter is applicable only to PWs transporting packets and is REQUIRED for these PW types. If this parameter does not match in both directions of a specific PW, that PW MUST NOT be enabled.

- Optional Interface Description string sub-TLV type

  This arbitrary, and OPTIONAL, interface description string is used to send a human-readable administrative string describing the interface to the remote. This parameter is OPTIONAL, and is applicable to all PW types. The interface description parameter string length is variable, and can be from 0 to 80 octets. Human-readable text MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

6.5. LDP label Withdrawal procedures

As mentioned above, the Group ID field of the PWid FEC element, or the PW Grouping ID TLV used with the Generalized PWid FEC element, can be used to withdraw all PW labels associated with a particular PW group. This procedure is OPTIONAL, and if it is implemented, the LDP Label Withdraw message should be as follows: If the PWid FEC element is used, the PW information length field is set to 0, the PW ID field is not present, the interface parameter sub-TLVs are not present, and the Label TLV is not present. If the Generalized FEC element is used, the AGI, SAII, and TAII are not present, the PW information length field is set to 0, the PW Group ID TLV is included, the Interface Parameters TLV is not present, and the Label TLV is not present. For the purpose of this document, this is called the "wild card withdraw procedure", and all PEs implementing this design are REQUIRED to accept such withdrawn message but are not required to send it. Note that the PW Group ID TLV only applies to PWs using the Generalized ID FEC element, while the Group ID only applies to PWid FEC element.

The interface parameter sub-TLVs, or TLV, MUST NOT be present in any LDP PW Label Withdraw or Label Release message. A wild card Label Release message MUST include only the group ID, or Grouping ID TLV. A Label Release message initiated by a PE router must always include the PW ID.

7. Control Word

7.1. PW Types for which the Control Word is REQUIRED

The Label Mapping messages that are sent in order to set up these PWs MUST have C=1. When a Label Mapping message for a PW of one of these types is received and C=0, a Label Release message MUST be sent, with an "Illegal C-bit" status code. In this case, the PW will not be enabled

7.2. PW Types for which the Control Word is NOT mandatory

If a system is capable of sending and receiving the control word on PW types for which the control word is not mandatory, then each such PW endpoint MUST be configurable with a parameter that specifies whether the use of the control word is PREFERRED or NOT PREFERRED. For each PW, there MUST be a default value of this parameter. This specification does NOT state what the default value should be.

If a system is NOT capable of sending and receiving the control word on PW types for which the control word is not mandatory, then it behaves exactly as if it were configured for the use of the control

word to be NOT PREFERRED.

If a Label Mapping message for the PW has already been received but no Label Mapping message for the PW has yet been sent, then the procedure is as follows:

-i. If the received Label Mapping message has C=0, send a Label Mapping message with C=0; the control word is not used.

-ii. If the received Label Mapping message has C=1, and the PW is locally configured such that the use of the control word is preferred, then send a Label Mapping message with C=1; the control word is used.

-iii. If the received Label Mapping message has C=1, and the PW is locally configured such that the use of the control word is not preferred or the control word is not supported, then act as if no Label Mapping message for the PW had been received (That is: proceed to the next paragraph).

If a Label Mapping message for the PW has not already been received (or if the received Label Mapping message had C=1 and either local configuration says that the use of the control word is not preferred or the control word is not supported), then send a Label Mapping message in which the C-bit is set to correspond to the locally configured preference for use of the control word. (That is, set C=1 if locally configured to prefer the control word, and set C=0 if locally configured to prefer not to use the control word or if the control word is not supported).

The next action depends on what control message is next received for that PW. The possibilities are as follows:

-i. A Label Mapping message with the same C-bit value as specified in the Label Mapping message that was sent. PW setup is now complete, and the control word is used if C=1 but is not used if C=0.

-ii. A Label Mapping message with C=1, but the Label Mapping message that was sent has C=0. In this case, ignore the received Label Mapping message and continue to wait for the next control message for the PW.

-iii. A Label Mapping message with C=0, but the Label Mapping message that was sent has C=1. In this case, send a Label Withdraw message with a "Wrong C-bit" status code, followed by a Label Mapping message that has C=0. PW setup is now complete, and the control word is not used.

-iv. A Label Withdraw message with the "Wrong C-bit" status code. Treat as a normal Label Withdraw, but do not respond. Continue to wait for the next control message for the PW.

If at any time after a Label Mapping message has been received a corresponding Label Withdraw or Release is received, the action taken is the same as for any Label Withdraw or Release that might be received at any time.

If both endpoints prefer the use of the control word, this procedure will cause it to be used. If either endpoint prefers not to use the control word or does not support the control word, this procedure will cause it not to be used. If one endpoint prefers to use the control word but the other does not, the one that prefers not to use it has no extra protocol to execute; it just waits for a Label Mapping message that has C=0.

7.3. Control-Word Renegotiation by Label Request Message

It is possible that after the PW C-bit negotation procedure described above is completed, the local PE is re-provisioned with a different control word preference. Therefore once the Control-Word negotation procedures are completed, the procedure can be restarted as follows:

-i. If local PE has previously sent a Label Mapping message, it MUST send a Label Withdraw message to remote PE and wait until it has received a Label Release message from the remote PE.

-ii. the local PE MUST send a label release message to the remote PE for the specific label associated with the FEC that was advertized for this specific PW. Note: the above-mentioned steps of the Label Release message and Label Withdraw message are not required to be excuted in any specific sequence.

-iii. The local PE MUST send a Label Request message to the peer PE, and then MUST wait until it receives a Label Mapping message containing the remote PE’s currently configured preference for use of the control word.

Once the remote PE has successfully processed the Label Withdraw message and Label Release messages, it will reset the C-bit negotation state machine and its use of the control word with the locally configured preference.

From this point on the local and remote PEs will follow the C-bit negotaiation procedures defined in the previous section.

The above C-bit renegotation process SHOULD NOT be interupted until

it is completed, or unpredictable results might occur.

7.4. Sequencing Considerations

In the case where the router considers the sequence number field in the control word, it is important to note the following details when advertising labels.

7.4.1. Label Advertisements

After a label has been withdrawn by the output router and/or released by the input router, care must be taken not to advertise (re-use) the same released label until the output router can be reasonably certain that old packets containing the released label no longer persist in the MPLS-enabled network.

This precaution is required to prevent the imposition router from restarting packet forwarding with a sequence number of 1 when it receives a Label Mapping message that binds the same FEC to the same label if there are still older packets in the network with a sequence number between 1 and 32768. For example, if there is a packet with a sequence number=n, where n is in the interval [1,32768] traveling through the network, it would be possible for the disposition router to receive that packet after it re-advertises the label. Since the label has been released by the imposition router, the disposition router SHOULD be expecting the next packet to arrive with a sequence number of 1. Receipt of a packet with a sequence number equal to n will result in n packets potentially being rejected by the disposition router until the imposition router imposes a sequence number of n+1 into a packet. Possible methods to avoid this are for the disposition router always to advertise a different PW label, or for the disposition router to wait for a sufficient time before attempting to re-advertise a recently released label. This is only an issue when sequence number processing is enabled at the disposition router.

7.4.2. Label Release

In situations where the imposition router wants to restart forwarding of packets with sequence number 1, the router shall 1) send to the disposition router a Label Release Message, and 2) send to the disposition router a Label Request message. When sequencing is supported, advertisement of a PW label in response to a Label Request message MUST also consider the issues discussed in the section on Label Advertisements.

8. IANA Considerations

The authors request that IANA remove this section before publication and that IANA update any references to [RFC4447] in their registries to refer to this document.

9. Security Considerations

This document specifies the LDP extensions that are needed for setting up and maintaining pseudowires. The purpose of setting up pseudowires is to enable Layer 2 frames to be encapsulated in MPLS and transmitted from one end of a pseudowire to the other. Therefore we treat the security considerations for both the data plane and the control plane.

9.1. Data-Plane Security

With regard to the security of the data plane, the following areas must be considered:

- MPLS PDU inspection.
- MPLS PDU spoofing.
- MPLS PDU alteration.
- MPLS PSN protocol security.
- Access Circuit security.
- Denial of service prevention on the PE routers.

When an MPLS PSN is used to provide pseudowire service, there is a perception that security MUST be at least equal to the currently deployed Layer 2 native protocol networks that the MPLS/PW network combination is emulating. This means that the MPLS-enabled network SHOULD be isolated from outside packet insertion in such a way that it SHOULD NOT be possible to insert an MPLS packet into the network directly. To prevent unwanted packet insertion, it is also important to prevent unauthorized physical access to the PSN, as well as unauthorized administrative access to individual network elements.

As mentioned above, an MPLS-enabled network should not accept MPLS packets from its external interfaces (i.e., interfaces to CE devices or to other providers' networks) unless the top label of the packet was legitimately distributed to the system from which the packet is being received. If the packet's incoming interface leads to a different SP (rather than to a customer), an appropriate trust relationship must also be present, including the trust that the other SP also provides appropriate security measures.

The three main security problems faced when using an MPLS-enabled network to transport PWs are spoofing, alteration, and inspection. First, there is a possibility that the PE receiving PW PDUs will get a PDU that appears to be from the PE transmitting the PW into the PSN, but that was not actually transmitted by the PE originating the PW. (That is, the specified encapsulations do not by themselves enable the decapsulator to authenticate the encapsulator.) A second problem is the possibility that the PW PDU will be altered between the time it enters the PSN and the time it leaves the PSN (i.e., the specified encapsulations do not by themselves assure the decapsulator of the packet's integrity.) A third problem is the possibility that the PDU's contents will be seen while the PDU is in transit through the PSN (i.e., the specification encapsulations do not ensure privacy.) How significant these issues are in practice depends on the security requirements of the applications whose traffic is being sent through the tunnel, and how secure the PSN itself is.

9.2. Control-Plane Security

General security considerations with regard to the use of LDP are specified in section 5 of [RFC5036]. Those considerations also apply to the case where LDP is used to set up pseudowires.

A pseudowire connects two attachment circuits. It is important to make sure that LDP connections are not arbitrarily accepted from anywhere, or else a local attachment circuit might get connected to an arbitrary remote attachment circuit. Therefore, an incoming LDP session request MUST NOT be accepted unless its IP source address is known to be the source of an "eligible" LDP peer. The set of eligible peers could be pre-configured (either as a list of IP addresses, or as a list of address/mask combinations), or it could be discovered dynamically via an auto-discovery protocol that is itself trusted. (Obviously, if the auto-discovery protocol were not trusted, the set of "eligible peers" it produces could not be trusted.)

Even if an LDP connection request appears to come from an eligible peer, its source address may have been spoofed. Therefore, some means of preventing source address spoofing must be in place. For example, if all the eligible peers are in the same network, source address filtering at the border routers of that network could eliminate the possibility of source address spoofing.

The LDP MD5 authentication key option, as described in section 2.9 of [RFC5036], MUST be implemented, and for a greater degree of security, it must be used. This provides integrity and authentication for the LDP messages and eliminates the possibility of source address

spoofing. Use of the MD5 option does not provide privacy, but privacy of the LDP control messages is not usually considered important. As the MD5 option relies on the configuration of pre-shared keys, it does not provide much protection against replay attacks. In addition, its reliance on pre-shared keys may make it very difficult to deploy when the set of eligible neighbors is determined by an auto-configuration protocol.

When the Generalized PWid FEC Element is used, it is possible that a particular LDP peer may be one of the eligible LDP peers but may not be the right one to connect to the particular attachment circuit identified by the particular instance of the Generalized PWid FEC element. However, given that the peer is known to be one of the eligible peers (as discussed above), this would be the result of a configuration error, rather than a security problem. Nevertheless, it may be advisable for a PE to associate each of its local attachment circuits with a set of eligible peers rather than have just a single set of eligible peers associated with the PE as a whole.

10. Interoperability and Deployment

Section 2.2. of [RFC6410] specifies four requirements that an Internet Standard must meet. This section documents how this document meets those requirements.

The pseudowire technology was first deployed in 2001 and has been widely deployed by many carriers. [RFC7079] documents the results of a survey of PW implementations, with specific emphasis on Control Word usage. [EANTC] documents a public multi-vendor interoperability test of MPLS and Carrier Ethernet equipment, which included testing of Ethernet, ATM and TDM pseudowires.

The errata against [RFC4447] are generally editorial in nature and have been addressed in this document.

All features in this specification have been implemented by multiple vendors.

No IPR disloures have been made to the IETF related to this document, to RFC4447 or RFC6723, or to the Internet-Drafts that resulted in RFC4447 and RFC6723.

11. Acknowledgments

The authors wish to acknowledge the contributions of Vach Kompella, Vanson Lim, Wei Luo, Himanshu Shah, and Nick Weeds. The authors wish to also acknowledge the contribution of the authors of rfc6723, Lizhong Jin, Raymond Key, Simon Delord, Tom Nadeau, Sami Boutros, whose work has been incorporated in this revised document.

12. Normative References

[RFC2119] Bradner S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997

[RFC5036] "LDP Specification." Andersson, L. Ed., Minei, I. Ed., Thomas, B. Ed. January 2001. RFC5036, October 2007

[RFC3032] "MPLS Label Stack Encoding", Rosen E., Rekhter Y., Tappan D., Fedorkow G., Farinacci D., Li T., Conta A.. RFC3032

[RFC4446] "IANA Allocations for pseudo Wire Edge to Edge Emulation (PWE3)" Martini L. RFC4446, April 2006

[RFC7358] "Label Advertisement Discipline for LDP Forwarding Equivalence Classes (FECs)", Raza K., Boutros S., Martini L., RFC7358, October 2014

13. Informative References

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC3985] "PWE3 Architecture" Bryant, et al., RFC3985.

[RFC4842] "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation over Packet (CEP)", A. Malis, P. Pate, R. Cohen, Ed., D. Zelig, RFC4842, April 2007

[RFC4553] "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", Vainshtein A. Ed., Stein, YJ. Ed. RFC4553, June 2006

[RFC4619] "Encapsulation Methods for Transport of Frame Relay over Multiprotocol Label Switching (MPLS) Networks", Martini L. Ed. C. Kawa Ed., A. Malis, Ed. RFC4619, September 2006

[RFC4717] "Encapsulation Methods for Transport of Asynchronous Transfer Mode (ATM) over MPLS Networks", Martini L., Jayakumar J., Bocci M., El-Aawar N., Brayley J., Koleyni G. RFC4717, December 2006

[RFC4618] "Encapsulation Methods for Transport of PPP/High-Level Data Link Control (HDLC) Frames over MPLS Networks", Martini L. Rosen E., Heron G., Malis A. RFC4618, September 2006

[RFC4448] "Encapsulation Methods for Transport of Ethernet over MPLS Networks", Martini L. Ed., Rosen E., El-Aawar N., Heron G. RFC4448, April 2006.

[RFC4447] "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", Martini L. Ed., Rosen E., El-Aawar N., Smith T., Heron G. RFC4447, April 2006

[RFC6410] "Reducing the Standards Track to Two Maturity Levels", Housley R., Crocker D., Burger E. RFC6410, October 2011

[RFC6723] "Update of the Pseudowire Control-Word Negotiation Mechanism", Jin L. Ed., Key R. Ed., Delord S., Nadeau T., Boutros S. RFC5723, September 2012

[RFC6410] "Reducing the Standads Track to Two Maturity Levels", Housley R., Crocker D., Burger E. RFC6410, October 2011

[RFC7079] "The Pseudowire (PW) and Virtual Circuit Connectivity Verification (VCCV) Implementation Survey Results", Del Regno N., Malis A. RFC7079, November 2013

[ANSI] American National Standards Institute, "Synchronous Optical Network Formats", ANSI T1.105-1995.

[ITUG] ITU Recommendation G.707, "Network Node Interface For The Synchronous Digital Hierarchy", 1996.

[EANTC] The European Advanced Networking Test Center "MPLS and Carrier Ethernet: Service - Connect - Transport. Public Multi-Vendor Interoperability Test", February 2009.

14. Author Information

Luca Martini
Cisco Systems, Inc.
1899 Wynkoop Street
Suite 600
Denver, CO, 80202
e-mail: lmartini@cisco.com

Giles Heron
Cisco Systems
10 New Square
Bedfont Lakes
Feltham
Middlesex
TW14 8HA
UK
e-mail: giheron@cisco.com

15. Additional Historical Contributing Authors

This historical list is from the original RFC, and is not updated. It is intended for recognition of their work on RFC4447.

Nasser El-Aawar
Level 3 Communications, LLC.
1025 Eldorado Blvd.
Broomfield, CO, 80021
e-mail: nna@level3.net

Eric C. Rosen
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
e-mail: erosen@cisco.com

Dan Tappan
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
e-mail: tappan@cisco.com

Toby Smith
Google
6425 Penn Ave. #700
Pittsburgh, PA 15206
e-mail: tob@google.com

Dimitri Vlachos
Riverbed Technology
e-mail: dimitri@riverbed.com

Jayakumar Jayakumar,
Cisco Systems Inc.
3800 Zanker Road, MS-SJ02/2,
San Jose, CA, 95134
e-mail: jjayakum@cisco.com

Alex Hamilton,
Cisco Systems Inc.
485 East Tasman Drive, MS-SJC07/3,
San Jose, CA, 95134
e-mail: tahamilt@cisco.com

Steve Vogelsang
ECI Telecom
Omega Corporate Center
1300 Omega Drive
Pittsburgh, PA 15205
e-mail: stephen.vogelsang@ecitele.com

John Shirron
ECI Telecom
Omega Corporate Center
1300 Omega Drive
Pittsburgh, PA 15205
e-mail: john.shirron@ecitele.com

Andrew G. Malis
Verizon
60 Sylvan Rd.
Waltham, MA 02451
e-mail: andrew.g.malis@verizon.com

Vinai Sirkay
Reliance Infocomm
Dhirubai Ambani Knowledge City
Navi Mumbai 400 709
e-mail: vinai@sirkay.com

Vasile Radoaca
Nortel Networks
600 Technology Park
Billerica MA 01821
e-mail: vasile@nortelnetworks.com

Chris Liljenstolpe
149 Santa Monica Way
San Francisco, CA 94127
e-mail: ietf@cdl.asgaard.org

Dave Cooper
Global Crossing
960 Hamlin Court
Sunnyvale, CA 94089
e-mail: dcooper@gblx.net

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
e-mail: kireeti@juniper.net

Expiration Date: January 2017

PWE3
Internet-Draft
Updates: 5085 (if approved)
Intended status: Standards Track
Expires: March 6, 2015

T. Nadeau
lucidvision
L. Martini
S. Bryant
Cisco Systems

September 2, 2014

A Unified Control Channel for Pseudowires
draft-ietf-pwe3-vccv-for-gal-02

Abstract

This document describes a unified mode of operation for Virtual Circuit Connectivity Verification (VCCV), which provides a control channel that is associated with a pseudowire (PW). VCCV applies to all supported access circuit and transport types currently defined for PWs, as well as those being transported by the MPLS Transport Profile. This new mode is intended to augment those described in RFC5085. It describes new rules requiring this mode to be used as the default/mandatory mode of operation for VCCV. The older VCCV types will remain optional.

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 March 6, 2015.

Table of Contents

1. Requirements Language and Terminology

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

AC	Attachment Circuit [RFC3985].
AVP	Attribute Value Pair [RFC3931].
CC	Control Channel (used as CC Type).
CE	Customer Edge.
CV	Connectivity Verification (used as CV Type).
CW	Control Word [RFC3985].
L2SS	L2-Specific Sublayer [RFC3931].
LCCE	L2TP Control Connection Endpoint [RFC3931].

OAM	Operation and Maintenance.
PE	Provider Edge.
PSN	Packet Switched Network [RFC3985].
PW	Pseudowire [RFC3985].
PW-ACH	PW Associated Channel Header [RFC4385].
VCCV	Virtual Circuit Connectivity Verification [RFC5085].

2. Introduction

There is a need for fault detection and diagnostic mechanisms that can be used for end-to-end fault detection and diagnostics for a Pseudowire, as a means of determining the PW's true operational state. Operators have indicated in [RFC4377], and [RFC3916] that such a tool is required for PW operation and maintenance. To this end, the IETF's PWE3 Working Group defined the Virtual Circuit Connectivity Verification Protocol (VCCV) in [RFC5085] . Since then a number of interoperability issues have arisen with the protocol as it is defined.

Over time, a variety of VCCV options or "modes" have been created to support legacy hardware, these modes use of the CW in some cases, while in others the CW is not used. The difficulty of operating these different combinations of "modes" have been detailed in an implementation survey conducted by the PWE3 Working Group and documented in [RFC7079]. The implementation survey and the PWE3 Working Group have concluded that operators have difficulty deploying the VCCV OAM protocol due to the number of combinations and options for its use.

In addition to the implementation issues just described, the ITU-T and IETF have set out to enhance MPLS to make it suitable as an optical transport protocol. The requirements for this protocol are defined as the MPLS Transport Profile (MPLS-TP). The requirements for MPLS-TP can be found in [RFC5654]. In order to support VCCV when an MPLS-TP PSN is in use, the GAL-ACH had to be created [RFC5586]. This resulted in yet another mode of VCCV operation.

This document defines two modes of operation of VCCV: 1) with a control word or 2) without a control word, both with a ACH encapsulation making it possible to handle all of the other cases handled by the other modes of VCCV. The modes of operation defined in this document MUST be implemented.

Figure 1 depicts the architecture of a pseudowire as defined in [RFC3985]. It further depicts where the VCCV control channel resides within this architecture, which will be discussed in detail later in this document.

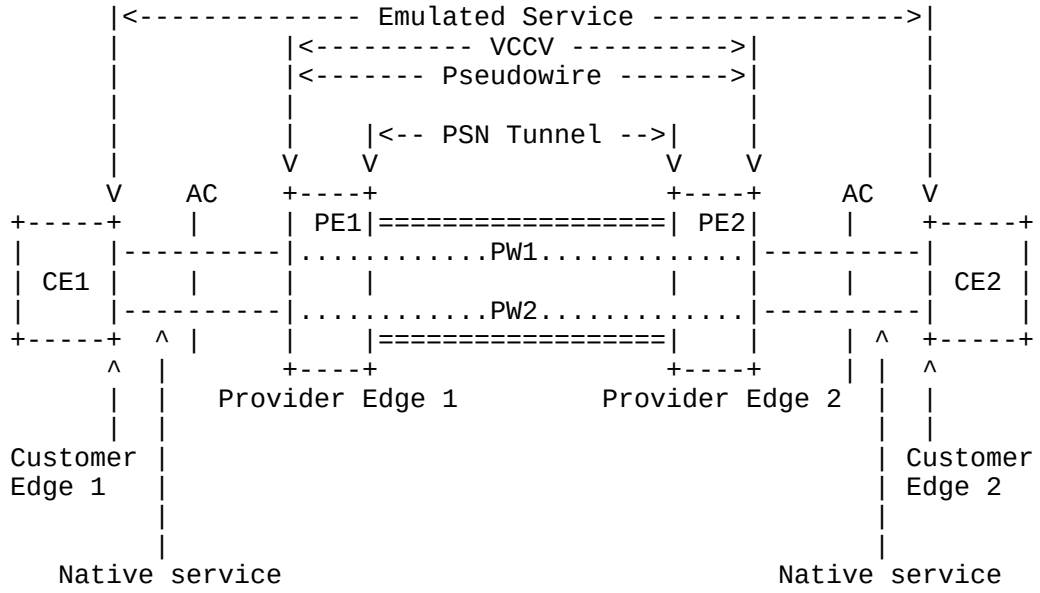


Figure 1: PWE3 VCCV Operation Reference Model

From Figure 1, Customer Edge (CE) routers CE1 and CE2 are attached to the emulated service via Attachment Circuits (AC), and to each of the Provider Edge (PE) routers (PE1 and PE2, respectively). An AC can be a Frame Relay Data Link Connection Identifier (DLCI), an ATM Virtual Path Identifier / Virtual Channel Identifier (VPI/VCI), an Ethernet port, or any other attachment type for which a PW is defined. The PE devices provide pseudowire emulation, enabling the CEs to communicate over the PSN. A pseudowire exists between these PEs traversing the provider network. VCCV provides several means of creating a control channel over the PW, between the PE routers that attach the PW.

Figure 2 depicts how the VCCV control channel is associated with the pseudowire protocol stack.

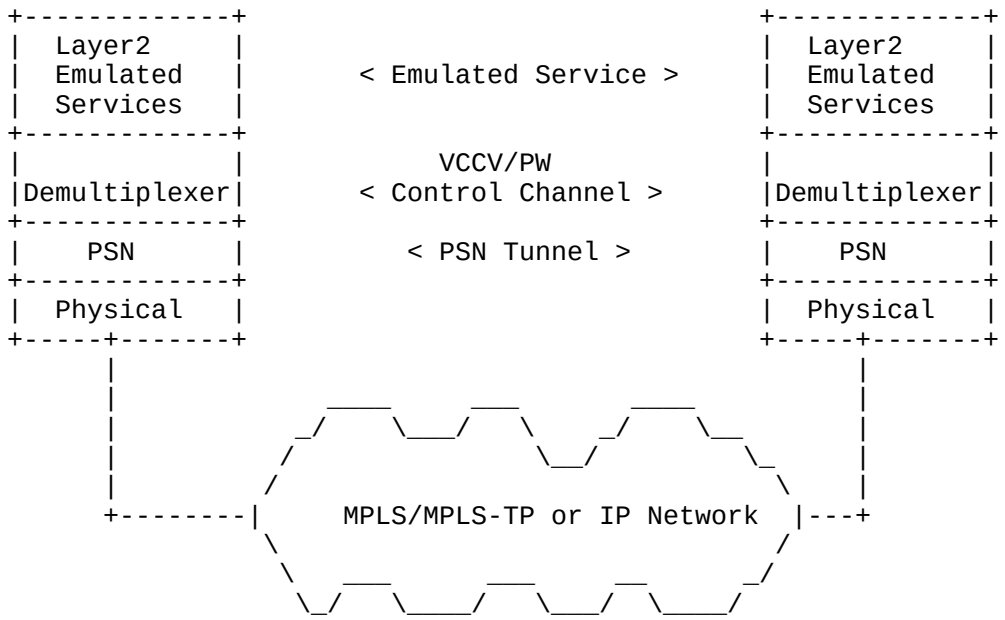


Figure 2: PWE3 Protocol Stack Reference Model including the VCCV Control Channel

VCCV messages are encapsulated using the PWE3 encapsulation as described in Section 3 and Section 4, so that they are handled and processed in the same manner (or in some cases, a similar manner) the PW PDUs for which they provide a control channel. These VCCV messages are exchanged only after the capability (the VCCV Control Channel and Connectivity Verification types) and the desire to exchange VCCV traffic has been advertised between the PEs (see Sections 5.3 and 6.3 of [RFC5085]), and VCCV type to use have been chosen.

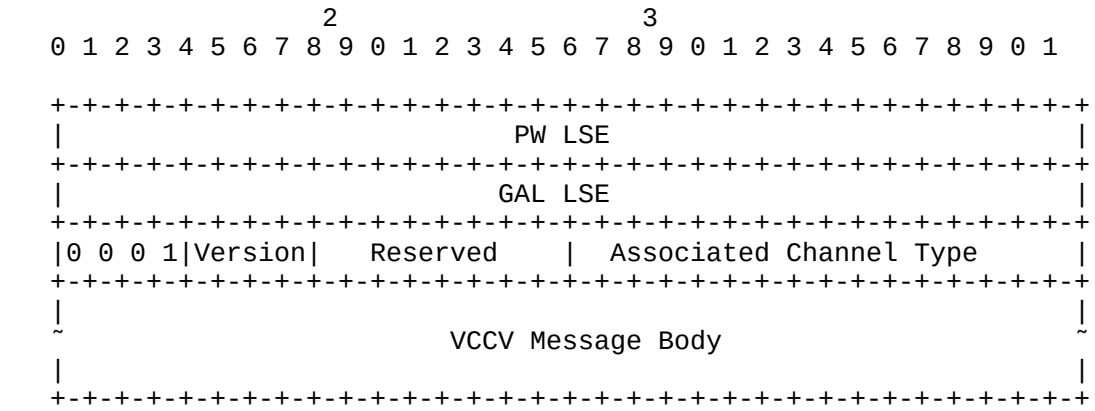
[EDITOR'S NOTE - Why are we talking about 6.3 which is L2TPv3 related in a text on GAL?]

3. VCCV Control Channel When The Control Word is Used

When the PWE3 Control Word is used to encapsulate pseudowire traffic, the rules described for encapsulating VCCV CC Type 1 as specified in section 9.5.1 of [RFC6073] and section 5.1.1 of [RFC5085] MUST be used. In this case the advertised CC Type is 1, and Associated Channel Types of 21, 07, or 57 are allowed.

4. VCCV Control Channel When The Control Word is Not Used

When the PWE3 Control Word is not used a new CC Type 4 is defined as follows:



EDITOR’s note = when we wrote RFC3985 I seem to remember that TTL=1 was problematic do we want to specify TTL=1 in the text below?

EDITOR’s note = not sure if it should be MUST or SHOULD in the text below.

When the PW is a single segment PW, the TTL field of the PW Label Stack Entry (LSE) SHOULD be set to 1. In the case of multi-segment pseudo-wires, the PW LSE TTL SHOULD be set to the value needed to reach the intended destination PE as described in [RFC6073].

The GAL LSE MUST contain the GAL reserved label as defined in [RFC5586].

As defined in [RFC4385] and [RFC4446] the first nibble of the next field is set to 0001b to indicate an ACH associated with a pseudowire instead of PW data. The Version and the Reserved fields MUST be set to 0, and the Channel Type is set to 0x0021 for IPv4, 0x0057 for IPv6 payloads [RFC5085] or 0x0007 for BFD payloads [RFC5885].

The Associated Channel Type defines how the "VCCV Message Body" field is to be interpreted by the receiver.

5. VCCV Capability Advertisement

The capability advertisement MUST match the c-bit setting that is advertised in the PW FEC element. If the c-bit is set, indicating the use of the control word, type 1 MUST be advertised and type 4 MUST NOT be advertised. If the c-bit is not set, indicating that the control word is not in use, type 4 MUST be advertised, and type 1 MUST NOT be advertised.

A PE supporting Type 4 MAY advertise other CC types as defined in [RFC5085] . If the remote PE also supports Type 4, then Type 4 MUST be used superseding the Capability Advertisement Selection rules of section 7 from [RFC5085] . If a remote PE does not support Type 4, then the rules from section 7 of [RFC5085] apply. If a CW is in use, then Type 4 is not applicable, and therefore the normal capability advertisement selection rules of section 7 from [RFC5085] apply.

6. Manageability Considerations

Editor’s note - this is a placeholder - I am not sure if it sis needed

7. Security Considerations

This document does not by itself raise any new security considerations beyond those described in [RFC5085].

8. IANA Considerations

8.1. VCCV Interface Parameters Sub-TLV

EDITOR’S NOTE ASFAICS this section can be deleted.

The VCCV Interface Parameters Sub-TLV code point is defined in [RFC4446]. IANA has created and will maintain registries for the CC Types and CV Types (bit masks in the VCCV Parameter ID). The CC Type and CV Type new registries (see Sections 8.1.1 and 8.1.2, respectively of[RFC5085]) have been created in the Pseudo Wires Name Spaces, . The allocations must be done using the "IETF Review" policy defined in [RFC5226].

8.2. MPLS VCCV Control Channel (CC) Type 4

IANA is requested to assign a new bit from the MPLS VCCV Control Channel (CC) Types registry in the PWE3-parameters name space in order to identify VCCV type 4. It is recommended that Bit 3 be assigned to this purpose which would have a value of 0x08.

MPLS VCCV Control Channel (CC) Types

Bit (Value)	Description	Reference
Bit X (0x0Y)	Type 4	[This Specification]

9. Acknowledgements

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3931] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, April 2006.

[RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC5654] Niven-Jenkins, B., Brungard, D., Betts, M., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, September 2009.

[RFC5885] Nadeau, T. and C. Pignataro, "Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)", RFC 5885, June 2010.

[RFC6073] Martini, L., Metz, C., Nadeau, T., Bocci, M., and M. Aissaoui, "Segmented Pseudowire", RFC 6073, January 2011.

10.2. Informative References

[RFC3916] Xiao, X., McPherson, D., and P. Pate, "Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)", RFC 3916, September 2004.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4377] Nadeau, T., Morrow, M., Swallow, G., Allan, D., and S. Matsushima, "Operations and Management (OAM) Requirements for Multi-Protocol Label Switched (MPLS) Networks", RFC 4377, February 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC7079] Del Regno, N. and A. Malis, "The Pseudowire (PW) and Virtual Circuit Connectivity Verification (VCCV) Implementation Survey Results", RFC 7079, November 2013.

Authors' Addresses

Thomas D. Nadeau
lucidvision

Email: tnadeau@lucidvision.com

Luca Martini
Cisco Systems

Email: lmartini@cisco.com

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

PALS Workgroup
Internet-Draft
Intended status: Standards Track
Expires: May 25, 2017

P. Jain
Cisco Systems, Inc.
S. Boutros
VMWare, Inc.
S. Aldrin
Google Inc.

November 21, 2016

Definition of P2MP PW TLV for LSP-Ping Mechanisms
draft-jain-pals-p2mp-pw-lsp-ping-02

Abstract

LSP-Ping is a widely deployed Operation, Administration, and Maintenance (OAM) mechanism in MPLS networks. This document describes a mechanism to verify connectivity of Point-to-Multipoint (P2MP) Pseudowires (PW) using LSP Ping.

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 May 25, 2017.

Table of Contents

1. Introduction

A Point-to-Multipoint (P2MP) Pseudowire (PW) emulates the essential attributes of a unidirectional P2MP Telecommunications service such as P2MP ATM over PSN. Requirements for P2MP PW are described in [RFC7338]. P2MP PWs are carried over P2MP MPLS LSP. The Procedures for P2MP PW signaling using BGP are described in [RFC7117] and LDP for single segment P2MP PWs are described in [I-D.ietf-pwe3-p2mp-pw]. Many P2MP PWs can share the same P2MP MPLS LSP and this arrangement is called Aggregate P-tree. The aggregate P2MP trees require an upstream assigned label so that on the tail of the P2MP LSP, the traffic can be associated with a VPN or a VPLS instance. When a P2MP MPLS LSP carries only one VPN or VPLS service instance, the arrangement is called Inclusive P-Tree. For Inclusive P-Trees, P2MP MPLS LSP label itself can uniquely identify the VPN or VPLS service being carried over P2MP MPLS LSP. The P2MP MPLS LSP can also be used in Selective P-Tree arrangement for carrying multicast traffic. In a Selective P-Tree arrangement, traffic to each multicast group in a VPN or VPLS instance is carried by a separate unique P-tree. In Aggregate Selective P-tree arrangement, traffic to a set of multicast groups from different VPN or VPLS instances is carried over a same shared P-tree.

The P2MP MPLS LSP are setup either using P2MP RSVP-TE [RFC4875] or Multipoint LDP (mDLP) [RFC6388]. Mechanisms for fault detection and isolation for data plane failures for P2MP MPLS LSPs are specified in

[RFC6425]. This document describes a mechanism to detect data plane failures for P2MP PW carried over P2MP MPLS LSPs.

This document defines a new P2MP Pseudowire sub-TLV for Target FEC Stack for P2MP PW. The P2MP Pseudowire sub-TLV is added in Target FEC Stack TLV by the originator of the Echo Request to inform the receiver at P2MP MPLS LSP tail, of the P2MP PW being tested.

Multi-segment Pseudowires support is out of scope of this document at present and may be included in future.

2. Specification of Requirements

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].

3. Terminology

ATM: Asynchronous Transfer Mode

LSR: Label Switching Router

MPLS-OAM: MPLS Operations, Administration and Maintenance

P2MP-PW: Point-to-Multipoint PseudoWire

PW: PseudoWire

TLV: Type Length Value

4. Identifying a P2MP PW

This document introduces a new LSP Ping Target FEC Stack sub-TLV, P2MP Pseudowire sub-TLV, to identify the P2MP PW under test at the P2MP LSP Tail/Bud node.

4.1. P2MP Pseudowire Sub-TLV

The P2MP Pseudowire sub-TLV has the format shown in Figure 1. This TLV is included in the echo request sent over P2MP PW by the originator of request.

The Attachment Group Identifier (AGI) in P2MP Pseudowire Sub-TLV as described in Section 3.4.2 in [RFC4446], identifies the VPLS instance. The Originating Router’s IP address is the IPv4 or IPv6 address of the P2MP PW root.

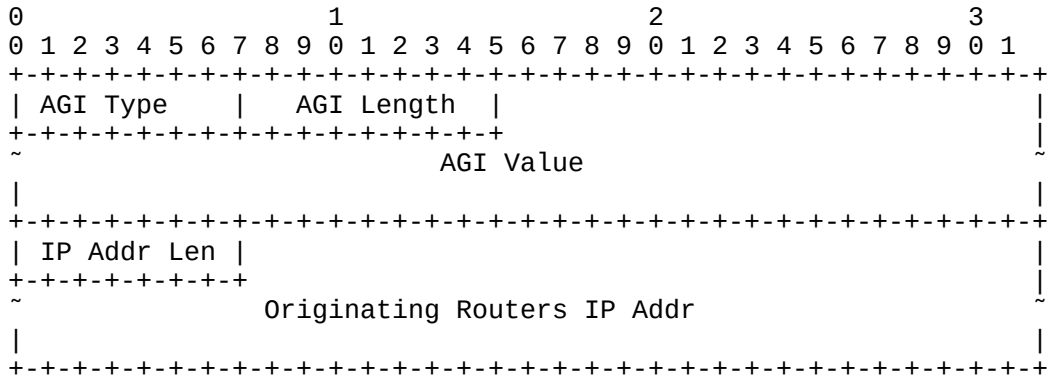


Figure 1: P2MP Pseudowire sub-TLV format

For Inclusive and Selective P2MP MPLS P-trees, the echo request is sent using the P2MP MPLS LSP label.

For Aggregate Inclusive and Aggregate Selective P-trees, the echo request is sent using a label stack of [P2MP MPLS P-tree label, upstream assigned P2MP PW label]. The P2MP MPLS P-tree label is the outer label and upstream assigned P2MP PW label is inner label.

5. Encapsulation of OAM Ping Packets

The LSP Ping Echo request IPv4/UDP packets will be encapsulated with the MPLS label stack as described in previous sections, followed by the GAL Label [RFC6426]. The GAL label will be followed by the ACH with the Pseudowire Associated Channel Type 16 bit value in the ACH set to IPv4 indicating that the carried packet is an IPv4 packet.

6. Operations

In this section, we explain the operation of the LSP Ping over P2MP PW. Figure 2 shows a P2MP PW PW1 setup from T-PE1 to remote PEs (T-PE2, T-PE3 and T-PE4). The transport LSP associated with the P2MP PW1 can be MLDP P2MP MPLS LSP or P2MP TE tunnel.

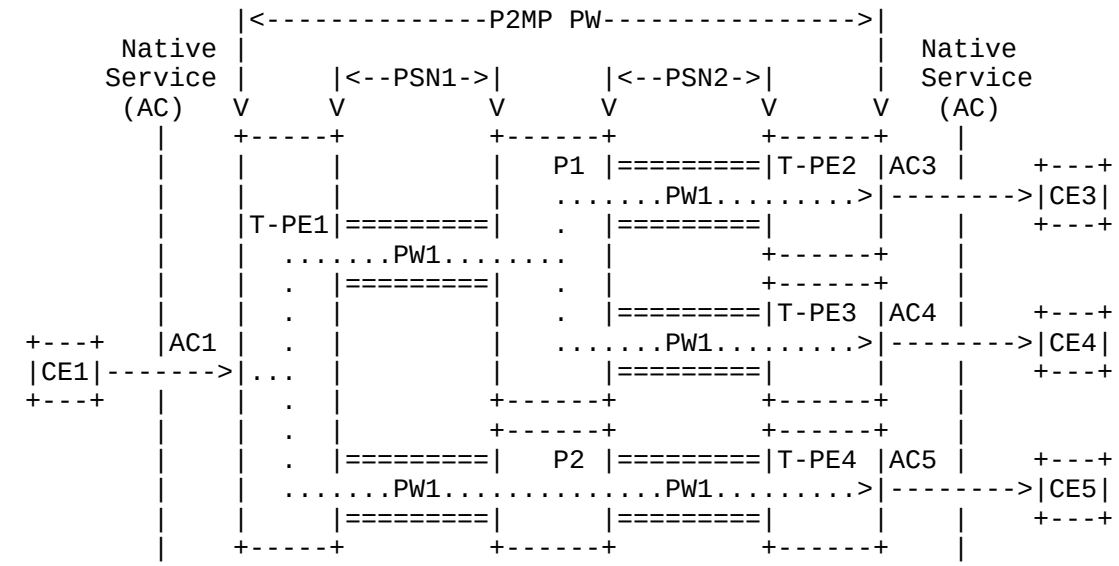


Figure 2: P2MP PW

When an operator wants to perform a connectivity check for the P2MP PW1, the operator initiate a LSP-Ping request with the Target FEC Stack TLV containing P2MP Pseudowire sub-TLV in the echo request packet. For an Inclusive P2MP P-tree arrangement, the echo request packet is sent over the P2MP MPLS LSP with {P2MP P-tree label, GAL} MPLS label stack and IP ACH Channel header. For an Aggregate Inclusive P-tree arrangement, the echo request packet is sent over the P2MP MPLS LSP with {P2MP P-tree label, P2MP PW upstream assigned label, GAL} MPLS label stack and IP ACH Channel header. The intermediate P router will do swap and replication based on the MPLS LSP label. Once the echo request packet reaches remote terminating PEs, T-PE1s will use the GAL label and the IP ACH Channel header to determine that the packet is IPv4 OAM Packet. The T-PEs will process the packet and perform checks for the P2MP Pseudowire sub-TLV present in the Target FEC Stack TLV as described in Section 4.4 in [RFC4379] and respond according to [RFC4379] processing rules.

7. Controlling Echo Responses

The procedures described in [RFC6425] for preventing congestion of Echo Responses (Echo Jitter TLV) and limiting the echo reply to a single egress node (Node Address P2MP Responder Identifier TLV) can be applied to P2MP PW LSP Ping.

8. Security Considerations

The proposal introduced in this document does not introduce any new security considerations beyond that already apply to [RFC6425].

9. IANA Considerations

This document defines a new sub-TLV type to be included in Target FEC Stack TLV (TLV Type 1) [RFC4379] in LSP Ping.

IANA is requested to assign a sub-TLV type value to the following sub-TLV from the "Multiprotocol Label Switching (MPLS) Label Switched Paths (LSPs) Parameters - TLVs" registry, "TLVs and sub- TLVs" sub-registry:

- P2MP Pseudowire sub-TLV

10. Acknowledgments

The authors would like to thank Shaleen Saxena, Michael Wildt, Tomofumi Hayashi, Danny Prairie for their valuable input and comments.

11. References

11.1. Normative References

[I-D.ietf-pwe3-p2mp-pw]
Sivabalan, S., Boutros, S., and L. Martini, "Signaling Root-Initiated Point-to-Multipoint Pseudowire using LDP", draft-ietf-pwe3-p2mp-pw-04 (work in progress), March 2012.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <http://www.rfc-editor.org/info/rfc4446>.

[RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <http://www.rfc-editor.org/info/rfc6425>.

[RFC6426] Gray, E., Bahadur, N., Boutros, S., and R. Aggarwal, "MPLS On-Demand Connectivity Verification and Route Tracing", RFC 6426, DOI 10.17487/RFC6426, November 2011, <http://www.rfc-editor.org/info/rfc6426>.

[RFC7117] Aggarwal, R., Ed., Kamite, Y., Fang, L., Rekhter, Y., and C. Kodeboniya, "Multicast in Virtual Private LAN Service (VPLS)", RFC 7117, DOI 10.17487/RFC7117, February 2014, <http://www.rfc-editor.org/info/rfc7117>.

11.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <http://www.rfc-editor.org/info/rfc4875>.

[RFC5085] Nadeau, T., Ed. and C. Pignataro, Ed., "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, DOI 10.17487/RFC5085, December 2007, <http://www.rfc-editor.org/info/rfc5085>.

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <http://www.rfc-editor.org/info/rfc6388>.

[RFC7338] Jounay, F., Ed., Kamite, Y., Ed., Heron, G., and M. Bocci, "Requirements and Framework for Point-to-Multipoint Pseudowires over MPLS Packet Switched Networks", RFC 7338, DOI 10.17487/RFC7338, September 2014, <http://www.rfc-editor.org/info/rfc7338>.

Authors' Addresses

Parag Jain
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8
Canada

Email: paragj@cisco.com

Sami Boutros
VMWare, Inc.
USA

Email: sboutros@vmware.com

Sam Aldrin
Google Inc.
USA

Email: aldrin.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

S. Zhuang
H. Wang
Z. Li
Huawei Technologies
March 8, 2015

Yang Model for L2VPN
draft-zhuang-pals-l2vpn-yang-00

Abstract

This document defines a YANG data model that can be used to configure and manage L2VPN. Both VPWS and VPLS are supported.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 9, 2015.

Copyright Notice

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 is proving relevant beyond its initial confines, as bindings to other interfaces(e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage L2VPN. Both VPWS and VPLS are supported.

2. Terminology

L2VPN: Layer 2 Virtual Private Network

VPLS: Virtual Private LAN Service

VPWS: Virtual Private Wire Service

3. Design of Data Model

3.1. Overview

The L2VPN Yang module is divided in following containers :

o l2vpncommon : that contains common writable configuration and readable objects for VPWS and VPLS.

o l2vpnvpws : that contains writable configuration and readable objects for VPWS.

o l2vpnvpls: that contains writable configuration and readable objects for VPLS.

The figure below describe the overall structure of the L2VPN Yang module :

```
module: l2vpn
   +--rw l2vpncommon
   ...
   +--rw l2vpnvpws
   ...
   +--rw l2vpnvpls
   ...
      ...
```

3.2. L2VPN Common Configuration

L2VPN common configuration container includes the global parameters for L2VPN, PW template configuration, etc. These parameters can be used by both VPWS and VPLS.

PW template configuration includes peer address, control word, MTU, sequence number, tunnel policy, parameters of AC, etc.

```
+--rw l2vpncommon
|  +--rw l2vpnGlobal
|  |  +--rw l2vpnEnable             boolean
|  |  +--rw vplsLoopDetectEnable?   boolean
|  +--rw pwTemplates
|  |  +--rw pwTemplate* [pwTemplateName]
|  |     +--rw pwTemplateName          string
|  |     +--rw peerAddr?               inet:ip-address
|  |     +--rw mtu?                    uint16
|  |     +--rw ctrlWord?               enumeration
|  |     +--rw tunnelPolicy?           string
|  |     +--rw tdmEncapsulateNumber?   uint8
|  |     +--rw jitterBuffer?           uint16
|  |     +--rw rtpHeader?              boolean
|  |     +--rw idleCode?               string
|  |     +--rw tdmSequenceNumber?      boolean
|  |     +--rw payloadCompression?     boolean
|  |     +--rw timeSlot?               uint8
|  |     +--rw maxAtmCells?            uint8
|  |     +--rw atmPackOvertime?        uint16
|  |     +--rw atmTransmitCell?        uint8
|  |     +--rw sequenceNumber?         boolean
...
```

3.3. VPWS Configuration

The VPWS configuration container includes VPWS instance configuration, VPWS switch instance configuration and VPWS statistics information.

```
+--rw l2vpnvpws
|  +--rw vpwsStatisticInfo
|  ...
|  +--rw vpwsInstances
|  ...
|  +--rw vpwsSwitchInstances
|  ...
```

3.3.1. VPWS Instances Configuration

The VPWS instance configuration includes per-instance parameters, AC configuration, PW configuration, TDM parameters, ATM parameters, reliability (including PW redundancy) configuration etc.

```
   +--rw vpwsInstances
   |  +--rw vpwsInstance* [instanceName instanceType]
   |     +--rw instanceName       leafref
   |     +--rw instanceType       instanceType
   |     +--rw encapsulateType?   pw-encapsulation
   |     +--rw description?       string
   |     +--ro instanceState?     enumeration
   |     +--ro lastUpTime?        yang:date-and-time
   |     +--ro totalUpTime?       string
   |     +--rw tdmParameters
   |     |  +--rw tdmEncapsulateNumber?   uint8
   |     |  ...
   |     +--rw atmParameters
   |     |  ...
   |     +--rw l2vpnAcs
   |     |  ...
   |     +--rw vpwsPws
   |     |  ...
   |     +--rw reliabilitys
   |        +--rw reliability* [pwRedundancyMode]
   ...
```

3.3.2. VPWS Switch Instances Configuration

VPWS switch instance configuration includes the configuration for multi-segment PW such as per-instance parameters, PW configuration, ATM parameters, TDM parameters etc.

```
   +--rw vpwsSwitchInstances
      +--rw vpwsSwitchInstance* [instanceName instanceType]
         +--rw instanceName       string
         +--rw instanceType       instanceType
         +--rw encapsulateType?   pw-encapsulation
         +--rw switchType?        enumeration
         +--rw ctrlWordTrans?     boolean
         +--rw controlWord?       enumeration
         +--ro instanceState?     enumeration
         +--ro createTime?        string
         +--ro upTime?            string
         +--ro lastChgTime?       string
         +--ro lastUpTime?        yang:date-and-time
         +--ro totalUpTime?       string
         +--rw vpwsPws
            +--rw vpwsPw* [pwRole pwId]
               +--rw pwRole            pw-role
               +--rw pwId              uint32
               +--rw peerIp?           inet:ip-address
               +--rw transmitLabel?    uint32
               +--rw receiveLabel?     uint32
               +--rw ctrlWord?         enumeration
               +--rw vccvAbility?      boolean
               +--rw tnlPolicyName?    string
               +--rw pwTemplateName?   string
               +--rw requestVlanId?    uint16
               +--rw vlanTpId?         string
               +--rw pwTtl?            uint8
               +--rw tdmParameters
               ...
               +--rw atmParameters
               ...
               +--rw vpwsLdpPwInfo
   ...
```

3.3.3. VPWS Statistics Information

The VPWS statistics information container includes statistics information of VPWS.

```
   +--rw vpwsStatisticInfo
   |  +--rw vpwsLdpAcStatInfo
   |  |  +--ro totalLdpAcNum?   uint32
   |  |  +--ro upLdpAcNum?      uint32
   |  |  +--ro downLdpAcNum?    uint32
   |  +--rw vpwsLdpPwStatInfo
   |  |  +--ro totalLdpPwNum?   uint32
   |  |  +--ro upLdpPwNum?      uint32
   |  |  +--ro downLdpPwNum?    uint32
   |  +--rw vpwsLdpPwRemoteStatInfo
   |  |  +--ro remoteVcNum?   uint32
   |  +--rw vpwsSwitchInstanceStatInfo
   |     +--ro totalSwitchInstanceNum?   uint32
   |     +--ro upSwitchInstanceNum?      uint32
   |     +--ro downSwitchInstanceNum?    uint32
```

3.4. VPLS Configuration

The L2VPN VPLS configuration includes VPLS instance configuration, VPLS statistics information.

```
   +--rw l2vpnvpls
      +--rw vplsStatisticInfo
      |  +--rw vplsInstStatisticsInfo
      ...
      |  +--rw vplsPwStatisticsInfo
      ...
      |  +--rw vplsAcStatisticsInfo
      ...
      |  +--ro vplsTnlRefInfos
      ...
      |  +--rw vplsLoopDetectStaticInfo
      |     +--ro totalVplsLoopDetectNum?   uint32
      +--rw vplsInstances
         +--rw vplsInstance* [instanceName]
            +--rw instanceName            string
            +--rw description?            string
            +--rw memberDiscoveryMode?    enumeration
            +--rw encapsulateType?        pw-encapsulation
            +--rw mtuValue?               uint16
            ...
```

3.4.1. VPLS Instance Configuration

The VPLS instance configuration includes member discovery mode, encapsulate type, VPLS LDP instance configuration, VPLS BGP AD instance configuration, VPLS BGP instance configuration and VPLS ACs configuration etc.

-- VPLS LDP instance configuration: This configuration describes how to configure LDP-based VPLS, with the signaling type being LDP.

-- VPLS BGP AD instance configuration: This configuration describes how to configure BGP AD VPLS to exchange extended BGP packets to automatically discover member VSIs in a VPLS domain and then use LDP FEC 129 to negotiate PW establishment to achieve automatic VPLS PW deployment.

-- VPLS BGP instance configuration: This configuration describes how to configure BGP VPLS. Detailed operations include configuring BGP as the signaling protocol, and configuring VPN targets to implement automatic discovery of VPLS PEs.

-- VPLS ACs configuration: This configuration describes configuration parameters of ACs.

```
+--rw vplsInstances
   +--rw vplsInstance* [instanceName]
      +--rw instanceName            string
      +--rw description?            string
      +--rw memberDiscoveryMode?    enumeration
      +--rw encapsulateType?        pw-encapsulation
      +--rw mtuValue?               uint16
      ...
      +--rw vsiPipe
      ...
      +--rw vplsLdpInst
      |  +--rw vsiId?               uint32
      ...
      +--rw vplsBgpAdInst
      |  +--rw vplsId?           string
      |  +--ro bgpAdRd?          string
      |  +--ro vsiId?            inet:ip-address
      |  +--rw vpnTargets
      ...
      +--rw vplsBgpInst
      |  +--rw bgpRd?          string
      |  +--rw ignoreMtu?      boolean
      ...
      +--rw vplsAcs
      |  +--rw vplsAc* [interfaceName]
      ...
      +--rw vplsLoopDetectInfo
         ...
```

3.4.2. VPLS Statistics Information

The VPLS statistics information container includes VPLS instance statistics information, VPLS PW statistics information, VPLS AC statistics information etc.

```
+--rw vplsStatisticInfo
|  +--rw vplsInstStatisticsInfo
...
|  +--rw vplsPwStatisticsInfo
...
|  +--rw vplsAcStatisticsInfo
...
|  +--ro vplsTnlRefInfos
...
|  +--rw vplsLoopDetectStaticInfo
|     +--ro totalVplsLoopDetectNum?   uint32
...
```

4. L2VPN Yang Module

<CODE BEGINS> file "l2vpn-yang@2014-08-21.yang"

```
module l2vpn {
    namespace "urn:huawei:params:xml:ns:yang:l2vpn";
    prefix "l2vpn";

    /* import */
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-yang-types {
        prefix yang;
    }
    description
        "This YANG module defines the generic configuration data for
         L2VPN services.

         Terms and Acronyms
         L2VPN: Layer 2 Virtual Private Network
         VPLS: Virtual Private LAN Service
         VPWS: Virtual Private Wire Service
         ...
         ";
```

```
    revision 2014-08-21 {
        description
            "Initial revision.";
    }

    /* Typedef */
    typedef pw-encapsulation {
        description "PW encapsulation type.";
        type enumeration {
            enum fr {
                value "0";
                description "fr:";
            }
            enum atm-aal5-sdu {
                value "1";
                description "atm-aal5-sdu:";
            }
            enum atm-trans-cell {
                value "2";
                description "atm-trans-cell:";
            }
            enum vlan {
                value "3";
                description "vlan:";
            }
            enum ethernet {
                value "4";
                description "ethernet:";
            }
            enum hdlc {
                value "5";
                description "hdlc:";
            }
            enum ppp {
                value "6";
                description "ppp:";
            }
            enum cem {
                value "7";
                description "cem:";
            }
            enum atm-nto1-vcc {
                value "8";
                description "atm-nto1-vcc:";
            }
            enum atm-nto1-vpc {
                value "9";
                description "atm-nto1-vpc:";
```

```
            }
            enum ip-layer2 {
                value "10";
                description "ip-layer2:";
            }
            enum atm-1to1-vcc {
                value "11";
                description "atm-1to1-vcc:";
            }
            enum atm-1to1-vpc {
                value "12";
                description "atm-1to1-vpc:";
            }
            enum atm-aal5-pdu {
                value "13";
                description "atm-aal5-pdu:";
            }
            enum fr-port-mode {
                value "14";
                description "fr-port-mode:";
            }
            enum cesop {
                value "15";
                description "cesop:";
            }
            enum satop-e1 {
                value "16";
                description "satop-e1:";
            }
            enum satop-t1 {
                value "17";
                description "satop-t1:";
            }
            enum satop-e3 {
                value "18";
                description "satop-e3:";
            }
            enum satop-t3 {
                value "19";
                description "satop-t3:";
            }
            enum cesopsn-basic {
                value "20";
                description "cesopsn-basic:";
            }
            enum tdmoip_aal1 {
                value "21";
                description "tdmoip_aal1:";
```

```
            }
            enum cesopsn_tdm {
                value "22";
                description "cesopsn_tdm:";
            }
            enum tdmoip_aal2 {
                value "23";
                description "tdmoip_aal2:";
            }
            enum fr_dlci {
                value "24";
                description "fr_dlci:";
            }
            enum ip-interworking {
                value "25";
                description "ip-interworking:";
            }
            enum unsupport {
                value "26";
                description "unsupport:";
            }
        }
    }

    typedef pw-role {
        description "pw role.";
        type enumeration {
            enum primary {
                value "0";
                description "primary:";
            }
            enum backup {
                value "1";
                description "backup:";
            }
            enum bypass {
                value "2";
                description "bypass:";
            }
            enum multiHopOneSidePrimary {
                value "3";
                description "multiHopOneSidePrimary:";
            }
            enum multiHopOtherSidePrimary {
                value "4";
                description "multiHopOtherSidePrimary:";
            }
            enum multiHopOtherSideBackup {
```

```
                value "5";
                description "multiHopOtherSideBackup:";
            }
        }
    }

    typedef tunnelType {
        description "Indicates the type of tunnel used by the PW.";
        type enumeration {
            enum invalid {
                value "0";
                description "invalid tunnel type";
            }
            enum ldp {
                value "1";
                description "LDP LSP";
            }
            enum bgp {
                value "2";
                description "BGP LSP";
            }
            enum te {
                value "3";
                description "TE tunnel";
            }
            enum static_lsp {
                value "4";
                description "static lsp";
            }
            enum gre {
                value "5";
                description "GRE tunnel";
            }
            enum uni {
                value "6";
                description "uni tunnel";
            }
            enum tnl_group {
                value "7";
                description "tnl-group";
            }
            enum sub_te {
                value "8";
                description "TE sub tunnel";
            }
            enum sub_group {
                value "9";
                description "sub tunnel group";
```

```
            }
            enum 6over4 {
                value "10";
                description "manual IPv6 tunnel carry IPv4 traffic";
            }
            enum 6to4 {
                value "11";
                description "automatic IPv6 tunnel carry IPv4 traffic";
            }
            enum bgp_local_ifnet {
                value "12";
                description "BGP created mpls localifnet tunnel";
            }
            enum ldp6 {
                value "13";
                description "IPv6 LDP LSP";
            }
        }
    }

    typedef ifState {
        description "Interface state.";
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum plugOut {
                value "2";
                description "plugOut:";
            }
            enum notifyDown {
                value "3";
                description "notifyDown:";
            }
            enum downNotify {
                value "4";
                description "downNotify:";
            }
        }
    }

    typedef pwState {
        description "Indicates the status of the PW.";
```

```
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum backup {
                value "2";
                description "backup:";
            }
        }
    }

    typedef instanceType {
        description "Instance type. ";

        type enumeration {
            enum vpwsLocalccc {
                value "0";
                description "vpwsLocalccc:";
            }
            enum vpwsRemoteccc {
                value "1";
                description "vpwsRemoteccc:";
            }
            enum vpwsSvc {
                value "2";
                description "vpwsSvc:";
            }
            enum vpwsLdp {
                value "3";
                description "vpwsLdp:";
            }
            enum vpwsSwitch {
                value "4";
                description "vpwsSwitch:";
            }
            enum vpls {
                value "5";
                description "vpls:";
            }
        }
    }

    /* Grouping */
```

```
    grouping tdmParameter {
        description
            "Configure TDM parameter.";
        leaf tdmEncapsulateNumber {
            description "Number of encapsulated TDM frames.";
            config "true";
            type uint8 {
                range "1..40";
            }
        }
        leaf jitterBuffer {
            description "Depth of the TDM jitter buffer.";
            config "true";
            type uint16 {
                range "1000..64000";
            }
        }
        leaf rtpHeader {
            description
                "Whether or not the RTP header is added into the
                 transparently transported TDM frame.";
            config "true";
            type boolean;
        }
        leaf idleCode {
            description
                "Specifies the value of the idle code that is filled
                 manually when the jitter buffer underflow occurs.";
            config "true";
            type string {
                length "1..2";
                pattern "^((([1-9]|[a-f]|[A-F])([0-9]|[a-f]|[A-F])?)|
                         (0([0-9]|[a-f]|[A-F])?))$";
            }
        }
        leaf tdmSequenceNumber {
            description "Enable the seq-number option.";
            config "true";
            type boolean;
        }
        leaf payloadCompression {
            description
                "Specifies the dynamic bandwidth allocation for payload
                 compression.";
            config "true";
            type boolean;
        }
        leaf timeSlot {
```

```
            description
                "Specifies the time slot of the serial interface.";
            config "true";
            type uint8 {
                range "1..32";
            }
        }
    }

    grouping atmParameter {

        description "Configure ATM parameter.";

        leaf maxAtmCells {
            description "Maximum number of transmitted ATM cells.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf atmPackOvertime {
            description "Delay in packing ATM cells.";
            config "true";
            type uint16 {
                range "100..10000";
            }
        }
        leaf atmTransmitCell {
            description "ATM transmit cell.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf sequenceNumber {
            description
                "Enable the seq-number option.";
            config "true";
            type boolean;
        }
    }

    grouping speInfos {

        leaf speCount {
            description "Number of Spe.";
            config "false";
            type uint32;
```

```
        }

        list speInfo {

            config "false";
            key "spePwId spePeerIp";

            leaf spePwId {
                description
                    "Indicates the identifier of the PW.";
                type uint32;
            }
            leaf spePeerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                type inet:ip-address;
            }
        }
    }

    grouping vpwsPws {

        list vpwsPw {

            key "pwRole pwId";
            description "L2vpn vpws pw class.";

            leaf pwRole {
                description
                    "VPWS pw role:primary, backup,bypass.";
                config "true";
                type pw-role;

            }
            leaf pwId {
                description
                    "Indicates the identifier of the PW.";
                config "true";
                type uint32 {
                    range "1..4294967295";
                }
            }
            leaf peerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                config "true";
                type inet:ip-address;
            }
```

```
            leaf transmitLabel {
                description
                    "Indicates the label value of sent packets.";
                config "true";
                type uint32 {
                    range "0..1048575";
                }
            }
            leaf receiveLabel {
                description
                    "Indicates the label value of received packets.";
                config "true";
                type uint32 {
                    range "16..32767";
                }
            }
            leaf ctrlWord {
                description
                    "Enables the control word function. The control word
                    function is usually enabled on PWs with encapsulation
                    types being TDM, ATM or FR.

                    By default:
                    The control word function is enabled for TDM-, ATM-, or
                    Frame Relay-encapsulated PWs if PW profiles are not used.
                    If a PW profile is used, the control word function can
                    be enabled only after the control word is explicitly
                    specified. The control word function can be enabled for
                    PWs that use other types of encapsulation only after
                    the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
            }
            leaf vccvAbility {
                description
```

```
                    "Configures VC connectivity detection. VC connectivity
                    detection supports two modes: control word mode and
                    label alert mode.";

                config "true";
                default "true";
                type boolean;
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the L2VC. If no name
                    is specified for a tunnel policy, the default tunnel
                    policy is adopted. The LSP tunnel is preferred and only
                    one LSP is used for load balancing in the default tunnel
                    policy. If the name of the tunnel policy is specified but
                    no tunnel policy is configured, the default tunnel policy
                    is still adopted.";

                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf pwTemplateName {
                description
                    "Specifies the name of a PW template. You can set
                     attributes for a PW template, including the remote
                     peer, tunnel policy, and control word. When configuring
                     an LDP PW, you can directly apply the PW template rather
                     than specifying attributes for the PW.";

                config "true";
                type string {
                    length "1..19";
                }
            }
            leaf requestVlanId {
                description
                    "Indicates the requested VLAN ID.";
                config "true";
                type uint16 {
                    range "1..4094";
                }
            }
            leaf vlanTpId {
                description "Indicates the TPID of requested VLAN ID.";
                config "true";
                type string {
```

```
                length "1..4";
                pattern "^([0-9]|[a-f]|[A-F]){0,4}$";
            }
        }
        leaf pwTtl {
            description "Specify the TTL for PW.";
            config "true";
            type uint8 {
                range "1..255";
            }
        }
        container tdmParameters {
            uses tdmParameter;
        }

        container atmParameters {
            uses atmParameter;
        }

        container vpwsLdpPwInfo {

            leaf interfaceName {
                description
                    "Indicates the type and number of the AC
                     interface.";
                config "false";
                type leafref {
                    path "/if:interfaces/if:interface/if:name";
                }
            }
            leaf ifState {
                description "Indicates  status of the AC interface.";
                config "false";
                type ifState;
            }
            leaf sessionState {
                description
                    "Indicates the status of the LDP session established
                     between both ends of the VC.";
                config "false";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum down {
                        value "1";
                        description "down:";
```

```
                    }
                    enum up {
                        value "2";
                        description "up:";
                    }
                }
            }
            leaf integrativeAcState {
                description
                    "The integrative status of the AC.";
                config "false";
                type ifState;
            }
            leaf acState {
                description
                    "Indicates the status of the AC.";
                config "false";
                type ifState;
            }
            leaf pwState {
                description
                    "Indicates the status of the PW.";
                config "false";
                type pwState;
            }
            leaf pwId {
                description
                    "Indicates the identifier of the PW.";
                config "false";
                type uint32;
            }
            leaf encapType {
                description
                    "Indicates the encapsulation type of the PW.";
                config "false";
                type pw-encapsulation ;

            }
            leaf destination {
                description
                    "Indicates the LSR ID of the VC peer device.";
                config "false";
                type inet:ip-address;
            }
            leaf localGroupId {
                description "Indicates the local group ID.";
                config "false";
                type uint32;
```

```
                }
                leaf remoteGroupId {
                    description "Indicates the remote group ID.";
                    config "false";
                    type uint32;
                }
                leaf localVcLabel {
                    description "Indicates the local VC label.";
                    config "false";
                    type uint32;
                }
                leaf remoteVcLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32;
                }
                container tdmInfo {

                    description "TDM info";
                    config "false";

                    leaf localTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf remoteTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf jitterBuffer {
                        description "Depth of the TDM jitter buffer.";
                        config "false";
                        type uint8;
                    }
                    leaf idleCode {
                        description
                             "Indicates the idle code that is filled manually
                              when the jitter buffer underflow occurs.";
                        config "false";
                        type string {
                            length "0..3";
                        }
                    }
                    leaf localRtpHeaderEnable {
                        description
                             "Whether or not the RTP header is added into
```

```
                         the transparently transported TDM frame.";
                    config "false";
                    type boolean;
                }
                leaf remoteRtpHeaderEnable {
                    description
                         "Whether or not the RTP header is added into the
                         transparently transported TDM frame.";
                    config "false";
                    type boolean;
                }
                leaf localBitRate {
                    description "Indicates the bit-rate of the local VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteBitRate {
                    description "Indicates the bit-rate of the remoteVC.";
                    config "false";
                    type uint16;
                }
            }
            container atmInfo {

                description "TDM info";
                config "false";

                leaf maxAtmCells {
                    description "Maximum number of transmitted ATM cells.";
                    config "false";
                    type uint8 {
                        range "1..28";
                    }
                }
                leaf remoteMaxAtmCells {
                    description "Maximum number of transmitted ATM cells.";
                    config "false";
                    type uint8 {
                        range "0..28";
                    }
                }
                leaf atmPackOvertime {
                    description "Delay in packing ATM cells.";
                    config "false";
                    type uint16 {
                        range "100..10000";
                    }
                }
```

```
                    leaf atmTransmitCell {
                        description "ATM transmit cell.";
                        config "false";
                        type uint16 {
                            range "1..28";
                        }
                    }
                    leaf sequenceNumber {
                        description
                             "Enable the seq-number option.By default, the
                              seq-number option is disabled.";
                        config "false";
                        type boolean;
                    }
                }
                leaf localFwdState {
                    description
                        "Indicates the status of the local forwarding table.";
                    config "false";
                    type enumeration {
                        enum notForwarding {
                            value "0";
                            description "notForwarding:";
                        }
                        enum forwarding {
                            value "1";
                            description "forwarding:";
                        }
                    }
                }
                leaf localStateCode {
                    description
                        "Indicates the status code of the local PW:
                        0x0: indicates that the local PW functioning as the
                             master PW is in the Up state.
                        0x20: indicates that the local PW functioning as the
                              backup PW is in the Up state.
                        0x1: indicates that the PW functioning as the master
                             PW and is in the Down state.
                        0x21: indicates that the PW functioning as the backup
                             PW and is in the Down state.";
                    config "false";
                    type uint32;
                }
                leaf remoteFwdState {
                    description
                        "Indicates the status of the remote forwarding
                         table.";
```

```
                    config "false";
                    type enumeration {
                        enum notForwarding {
                            value "0";
                            description "notForwarding:";
                        }
                        enum forwarding {
                            value "1";
                            description "forwarding:";
                        }
                    }
                }
                leaf remoteStateCode {
                    description
                        " Indicates the status code of the remote PW:
                        0x0: indicates that the remote PW functioning as the
                             master PW is in the Up state.
                        0x20: indicates that the remote PW functioning as the
                              backup PW is in the Up state.
                        0x1: indicates that the PW functioning as the master
                             PW and is in the Down state.
                        0x21: indicates that the PW functioning as the backup
                              PW and is in the Down state.";
                    config "false";
                    type uint32;
                }
                leaf isActive {
                    description
                        "Indicates whether or not the PW is in active state
                        (if so, user packets can be forwarded).";
                    config "false";
                    type boolean;
                }
                leaf isForwardExist {
                    description
                        "Indicates whether or not forwarding entries exist.";
                    config "false";
                    type boolean;
                }
                leaf linkState {
                    description
                        "Indicates the link status of the AC interface:
                        Up: indicates that the physical layer status of
                            the interface is functional.
                        Down: indicates that the physical layer of the
                              interface fails.";
                    config "false";
                    type enumeration {
```

```
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum down {
                        value "1";
                        description "down:";
                    }
                    enum up {
                        value "2";
                        description "up:";
                    }
                }
            }
            leaf localVcMtu {
                description "Indicates the MTU of the local VC.";
                config "false";
                type uint16;
            }
            leaf remoteVcMtu {
                description "Indicates the MTU of the remote VC.";
                config "false";
                type uint16;
            }
            leaf localVCCV {
                description
                    "Indicates the type of VCCV that is supported
                     locally. By default, the control word function
                     is not enabled, and the supported VCCV type is
                     alert lsp-ping bfd, indicating that LSP ping
                     and BFD are supported for the alert channel.
                     If the control word function is enabled, the
                     VCCV type is cw alert lsp-ping bfd, indicating
                     that LSP ping and BFD are supported both for the
                     control word channel and the alert channel.";
                config "false";
                type string {
                    length "0..40";
                }
            }
            leaf remoteVCCV {
                description
                    "Indicates the type of VCCV that is supported
                     remotely. By default, the control word function
                     is not enabled, and the supported VCCV type is
                     alert lsp-ping bfd, indicating that LSP ping and
                     BFD are supported for the alert channel. If the
                     control word function is enabled, the VCCV type
```

```
                        is cw alert lsp-ping bfd, indicating that LSP ping
                        and BFD are supported both for the control word
                        channel and the alert channel.";

                    config "false";
                    type string {
                        length "0..40";
                    }
                }
                leaf localCtrlWord {
                    description
                        "Indicates whether or not the control word is enabled
                        on the local end.";
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf remoteCtrlWord {
                    description
                        "Indicates whether or not the control word is enabled
                         on the remote end.";
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
```

```
                }
                leaf tnlPolicyName {
                    description "Indicates the name of the tunnel policy.";
                    config "false";
                    type string {
                        length "1..39";
                    }
                }
                leaf pwTemplateName {
                    description "Indicates the name of the PW template.";
                    config "false";
                    type string {
                        length "1..19";
                    }
                }
                leaf priOrSec {
                    description
                        "Indicates whether the local status of the VC is
                         primary or secondary.";
                    config "false";
                    type enumeration {
                        enum primary {
                            value "0";
                            description "primary:";
                        }
                        enum secondary {
                            value "1";
                            description "secondary:";
                        }
                        enum bypass {
                            value "2";
                            description "bypass:";
                        }
                    }
                }
                leaf tunnelCount {
                    description
                        "Indicates that the PW uses one tunnel or token";
                    config "false";
                    type uint8;
                }
                container tunnelInfos {

                    config "false";
                    list tunnelInfo {

                        key "tunnelKey";
```

```
                        leaf tunnelKey {
                            description
                                "Indicates the ID of the tunnel used by the
                                 PW.";

                            type string {
                                length "0..21";
                            }
                        }
                        leaf tunnelType {
                            description
                                "Indicates the type of tunnel used by the
                                 PW.";
                            config "false";
                            type tunnelType;
                        }
                        leaf tunnelName {
                            description
                                "Indicates the name of the tunnel used by the
                                 PW.";
                            config "false";
                            type string {
                                length "0..64";
                            }
                        }
                        leaf publicNextHop {
                            description
                                "Indicates public next hop of a tunnel.";
                            config "false";
                            type inet:ip-address;
                        }
                    }

                }

                container speInfos {
                    config "false";
                    uses speInfos;
                }

                leaf createTime {
                    description
                        "Indicates how long the VC has been created for.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
```

```
                leaf upTime {
                    description
                        "Indicates how long the VC keeps the Up state.
                         If the PW is currently in the Down state, the
                         value is 0.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf lastChgTime {
                    description
                        "Indicates how long the VC status has remained
                         unchanged.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf pwLastUpTime {
                    description
                        "Indicates the last time the VC was Up.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf pwTotalUpTime {
                    description
                        "Indicates the total duration the VC is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf supportNotification {
                    description
                        "Indicates whether or not the Notification
                         message is supported.";
                    config "false";
                    type boolean;
                }
            }

        }

    }

    grouping vplsPwInfo {
```

```
        leaf peerIp {
            description
                "Indicates the LSR ID of the VC peer device.";
            config "false";
            type inet:ip-address;
        }
        leaf pwId {
            description
                "Indicates the identifier of the PW.";
            config "false";
            type uint32;
        }
        leaf pwType {
            description "Type of the PW.label, QinQ,MEHVPLS";
            config "false";
            type enumeration {
                enum label {
                    value "0";
                    description "label:";
                }
                enum QinQ {
                    value "1";
                    description "QinQ:";
                }
                enum MEHVPLS {
                    value "2";
                    description "MEHVPLS:";
                }
            }
        }
        leaf sessionState {
            description
                "Indicates the status of the LDP session established
                 between both ends of the VC.";
            config "false";
            type enumeration {
                enum default {
                    value "0";
                    description "default:";
                }
                enum down {
                    value "1";
                    description "down:";
                }
                enum up {
                    value "2";
                    description "up:";
                }
```

```
            }
        }
        leaf pwState {
            description "Indicates the status of the PW.";
            config "false";
            type enumeration {
                enum down {
                    value "0";
                    description "down:";
                }
                enum up {
                    value "1";
                    description "up:";
                }
                enum backup {
                    value "2";
                    description "backup:";
                }
            }
        }
        leaf localVcLabel {
            description "Indicates the local VC label.";
            config "false";
            type uint32;
        }
        leaf remoteVcLabel {
            description "Indicates the remote VC label.";
            config "false";
            type uint32;
        }
        leaf tnlPolicyName {
            description
                "Indicates the name of the tunnel policy.";
            config "false";
            type string {
                length "1..39";
            }
        }
        leaf pwLastUpTime {
            description
                "Indicates the last time the VC was Up.";
            config "false";
            type yang:date-and-time;
        }
        leaf pwTotalUpTime {
            description
                "Indicates the total duration the VC is Up.";
            config "false";
```

```
            type string {
                length "1..80";
            }
        }
        container tunnelInfos {

            config "false";

            list tunnelInfo {

                key "tunnelKey";

                leaf tunnelKey {
                    description
                        "Indicates the ID of the tunnel used by the PW.";
                    type string {
                        length "0..21";
                    }
                }
                leaf tunnelType {
                    description
                        "Indicates the type of tunnel used by the PW.";
                    config "false";
                    type tunnelType;
                }
                leaf outIntf {
                    description
                        "Outbound interface.";
                    config "false";
                    type string {
                        length "0..256";
                    }
                }
                leaf tunnelName {
                    description
                        "Indicates the name of the tunnel used by the PW.";
                    config "false";
                    type string {
                        length "0..64";
                    }
                }
                leaf publicNextHop {
                    description "Assign public next hop of a tunnel.";
                    config "false";
                    type inet:ip-address;
                }
            }
```

```
        }
        container speInfos {
            config "false";
            uses speInfos;
        }
        leaf remoteGroupId {
            description "ID of the remote group.";
            config "false";
            type uint32;
        }
        leaf remoteMtu {
            description "Indicates the MTU of a remote VC.";
            config "false";
            type uint16;
        }
        leaf remoteVCCVcode {
            description "Indicates the VCCV of a remote VC.";
            config "false";
            type string {
                length "0..40";
            }
        }
        leaf remoteStateCode {
            description
                "Indicates the status of a remote VC, which can be:
                FORWARD: The remote VC is in the forwarding state.
                STANDBY: The remote VC is in the standby state.
                AC FAULT: The remote AC interface is faulty.
                PSN FAULT: The remote VC is faulty.
                NO FORWRD: The remote VC interface cannot forward packets
                           owing to other reasons. ";
            config "false";
            type enumeration {
                enum forward {
                    value "0";
                    description "forward:";
                }
                enum not-forward {
                    value "1";
                    description "not forward:";
                }
                enum standby {
                    value "2";
                    description "standby:";
                }
                enum ac-fault {
                    value "3";
                    description "ac fault:";
```

```
                }
                enum psn-fault {
                    value "4";
                    description "psn fault:";
                }
            }
        }
    }

    /* container */
    container l2vpncommon {

        container l2vpnGlobal {

            description "L2VPN golbal attribute.";

            leaf l2vpnEnable {
                description
                    "L2vpn enable flag.";
                type boolean;
                config "true";
                mandatory "true";
            }
            leaf vplsLoopDetectEnable {
                description
                    "Vpls mac withdraw loop detect enable flag.";
                type boolean;
                config "true";
            }
        }

        container pwTemplates {

            list pwTemplate {

                key "pwTemplateName";
                description "L2VPN pw template class.";

                leaf pwTemplateName {
                    description
                        "Specifies the PW template name. The value is a
                         case-sensitive string of 1 to 19 characters without
                         blank space.";
                    config "true";
                    type string {
                        length "1..19";
                    }
                }
```

```
                leaf peerAddr {
                    description
                        "Assign a peer IP address to a PW template.";
                    config "true";
                    type inet:ip-address;
                }
                leaf mtu {
                    description
                        "Configure the mtu value for PW template, 46 to 9600.";
                    config "true";
                    default "1500";
                    type uint16 {
                        range "46..9600";
                    }
                }
                leaf ctrlWord {
                    description
                        "Enable the control word in a PW template.";
                    config "true";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf tunnelPolicy {
                    description
                        "Configure a tunnel policy for a PW template.";
                    config "true";
                    type string {
                        length "1..39";
                    }
                }
                uses tdmParameter;

                uses atmParameter;

            }
        }
```

```
        container notMatchRemoteLdpInfos {

            config "false";
            list notMatchRemoteLdpInfo {

                key "pwId peerIp encapsulateType";

                leaf pwId {
                    description
                        "After an ID is set for a VC, it cannot be changed.
                         Different VCs have different IDs.";
                    type uint32;
                }
                leaf peerIp {
                    description "Indicates the peer ip of the VC peer device.";
                    type inet:ip-address;
                }
                leaf encapsulateType{
                    description "Indicates the encapsualtion VC peer device.";
                    type pw-encapsulation;
                }
                leaf remoteLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32 ;
                }
                leaf remoteGroupId {
                    description "ID of the remote group.";
                    config "false";
                    type uint32;
                }
                leaf remoteMtu {
                    description "Indicates the MTU of a remote VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteStateCode {
                    description
                        "Indicates the status of a remote VC, which can be:
                            FORWARD: The remote VC is in the forwarding state.
                            STANDBY: The remote VC is in the standby state.
                            AC FAULT: The remote AC interface is faulty.
                            PSN FAULT: The remote VC is faulty.
                            NO FORWRD: The remote VC interface cannot forward
                                       packets owing to other reasons.";
                    config "false";
                    type enumeration {
                        enum forward {
```

```
                            value "0";
                            description "forward:";
                        }
                        enum not-forward {
                            value "1";
                            description "not forward:";
                        }
                        enum standby {
                            value "2";
                            description "standby:";
                        }
                        enum ac-fault {
                            value "3";
                            description "ac fault:";
                        }
                        enum psn-fault {
                            value "4";
                            description "psn fault:";
                        }
                    }
                }
            }

        }
    }

    container l2vpnvpws {

        container vpwsStatisticInfo {

            container vpwsLdpAcStatInfo {

                leaf totalLdpAcNum {
                    description "Total number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
                leaf upLdpAcNum {
                    description "Up number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
                leaf downLdpAcNum {
                    description "Down number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
            }
```

```
            container vpwsLdpPwStatInfo {

                leaf totalLdpPwNum {
                    description
                        "Indicates the total number of established LDP PWs";
                    config "false";
                    type uint32;
                }
                leaf upLdpPwNum {
                    description "Number of LDP PWs in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downLdpPwNum {
                    description "Number of LDP PWs in the down state.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsLdpPwRemoteStatInfo {

                leaf remoteVcNum {
                    description
                        "Indicates the total number of created remote LDP
                         PWs.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsSwitchInstanceStatInfo {

                leaf totalSwitchInstanceNum {
                    description
                        "Indicates the total number of established switch-vc";
                    config "false";
                    type uint32;
                }
                leaf upSwitchInstanceNum {
                    description "Number of switch-vc in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downSwitchInstanceNum {
                    description "Number of switch-vc in the down state.";
                    config "false";
                    type uint32;
```

```
                }
            }

        }

        container vpwsInstances {

            list vpwsInstance {

                key "instanceName instanceType";

                description "L2vpn vpws instance class.";

                leaf instanceName {

                    description "Specifies VPWS instance name.";
                    config "true";

                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf instanceType {
                    description "VPWS instance type:ldp,localccc.";
                    config "true";
                    type instanceType;
                }
                leaf encapsulateType {
                    type pw-encapsulation;
                }
                leaf description {
                    description "Specifies a description for the VC.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf instanceState {
                    description "VPWS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
```

```
                        }
                        enum adminDown {
                            value "2";
                            description "adminDown:";
                        }
                    }
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the instance keeps the Up state.
                         If the PW is currently in the Down state, the value
                         is 0.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf totalUpTime {
                    description
                        "Indicates the total duration the instance is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }

                container tdmParameters {
                    uses tdmParameter;
                }

                container atmParameters {
                    uses atmParameter;
                }

                container l2vpnAcs {

                    list l2vpnAc {

                        key "interfaceName";
                        description "L2VPN ac class.";

                        leaf interfaceName {
                            description "Specifies the AC interface name.";
                            config "true";
                            type leafref {
                                path "/if:interfaces/if:interface/if:name";
                            }
                        }
                        leaf state {
                            description "Indicates the status of the AC.";
```

```
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum down {
                            value "1";
                            description "down:";
                        }
                        enum up {
                            value "2";
                            description "up:";
                        }
                    }
                }

                container l2vpnPipe {

                    description "L2VPN pipe mode.";

                    leaf pipeMode {
                        description "Pipe mode.";
                        default "uniform";
                        type enumeration {
                            enum pipe {
                                value "0";
                                description "pipe:";
                            }
                            enum shortPipe {
                                value "1";
                                description "shortPipe:";
                            }
                            enum uniform {
                                value "2";
                                description "uniform:";
                            }
                        }
                    }
                }

                container ifLinkProtocolTran {

                    description "lacp status";

                    leaf protocolLacp {
                        description "lacp status";
                        config "true";
```

```
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolLldp {
                description "lldp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolBpdu {
                description "bpdu status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolCdp {
                description "cdp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
```

```
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                    leaf protocolUdld {
                        description "udld status";
                        config "true";
                        type enumeration {
                            enum enable {
                                value "0";
                                description "enable:";
                            }
                            enum disable {
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                }

            }

            container vpwsPws {
                uses vpwsPws;
            }

            container reliabilitys {

                list reliability {

                    key "pwRedundancyMode";

                    leaf pwRedundancyMode {
                        description
                            "Specifies the redundancy mode of VPWS
                             instance.";

                        config "true";
                        type enumeration {
                            enum frr {
                                value "0";
                                description "frr:";
                            }
                            enum masterSlave {
                                value "1";
```

```
                    description "masterSlave:";
                }
                enum independent {
                    value "2";
                    description "independent:";
                }
            }
        }
        leaf switchover {
            description
                "Specifies switches traffic from the primary
                 PW to the secondary PW.";
            config "true";
            type boolean;
        }
        leaf dualReceive {
            description
                "Specifies enables a UPE interface to receive
                 packets from both the primary and secondary
                 PWs.";
            config "true";
            type boolean;
        }
        container reRoute {

            description "L2vpn vpws pw reroute class.";

            leaf reRoutePolicy {
                description "Specifies the Policy of Reroute.";
                config "true";
                type enumeration {
                    enum delay {
                        value "0";
                        description "delay:";
                    }
                    enum immediately {
                        value "1";
                        description "immediately:";
                    }
                    enum never {
                        value "2";
                        description "never:";
                    }
                }
            }
            leaf delayTime {
                description
                    "Specifies the delay for switching traffic
```

```
                    back to the primary PW. ";
                config "true";
                type uint16 {
                    range "10..600";
                }
            }
            leaf resumeTime {
                description
                    "Specifies the time after which the peer PE
                     on the secondary PW is notified that the
                     local PE is recovered from the fault. ";
                config "true";
                type uint16 {
                    range "0..600";
                }
            }
            leaf lastReRouteReason {
                description
                    "Specifies the reason of Last Reroute.";
                config "false";
                type string {
                    length "0..100";
                }
            }
            leaf lastReRouteTime {
                description
                    "Specifies the time of Last Reroute.";
                config "false";
                type string {
                    length "1..60";
                }
            }
            leaf delayResidual {
                description
                    "Specifies the residual delay time for
                     switching traffic back to the primary PW.
                    ";
                config "false";
                type uint32;
            }
            leaf resumeResidual {
                description
                    "Specifies the residual time after which
                     the peer PE on the secondary PW is
                     notified that the local PE is recovered
                     from the fault. ";
                config "false";
                type uint32;
```

```
                        }
                    }

                }

            }

        }

        container vpwsSwitchInstances {

            list vpwsSwitchInstance {

                key "instanceName instanceType";

                description "L2vpn vpws instance class.";

                leaf instanceName {
                    description "Specifies VPWS instance name.";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf instanceType {
                    description "VPWS instance type:vpwsSwitch.";
                    config "true";
                    type instanceType;
                }
                leaf encapsulateType {
                    description "VPWS instance encapsulation type.";
                    config "true";
                    default "ethernet";
                    type pw-encapsulation;
                }
                leaf switchType {
                    description
                        "VPWS switch instance type:ldp2ldp,ldp2svc.";
                    config "true";
                    default "ldp2ldp";
                    type enumeration {
                        enum svc2svc {
                            value "0";
                            description "svc2svc:";
                        }
                        enum ldp2svc {
```

```
                        value "1";
                        description "ldp2svc:";
                    }
                    enum ldp2ldp {
                        value "2";
                        description "ldp2ldp:";
                    }
                    enum upe {
                        value "3";
                        description "upe:";
                    }
                }
            }
            leaf ctrlWordTrans {
                description
                    "Transparent transmission of control word If BFD
                     is enabled to monitor dynamic multi-hop PWs,
                     transparent transmission of control word needs
                     to be configured on the SPE. Otherwise, the BFD
                     negotiation fails. By default, transparent
                     transmission of control word is disabled.";
                config "true";
                type boolean;
                default "false";
            }
            leaf controlWord {
                description
                    "Enables the control word function. The control
                     word function is usually enabled on PWs with
                     encapsulation types being TDM, ATM or FR.
                     By default:
                     The control word function is enabled for TDM-,
                     ATM-, or Frame Relay-encapsulated PWs if PW
                     profiles are not used. If a PW profile
                     is used, the control word function can be
                     enabled only after the control word is explicitly
                     specified. The control word function can be enabled
                     for PWs that use other types of encapsulation only
                     after the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
```

```
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf instanceState {
                    description "VPWS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
                        }
                        enum adminDown {
                            value "2";
                            description "adminDown:";
                        }
                    }
                }
                leaf createTime {
                    description
                        "Indicates how long the VC has been created for.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf upTime {
                    description
                        "Indicates how long the VC keeps the Up state. If the
                         PW is currently in the Down state, the value is 0.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf lastChgTime {
                    description
                        "Indicates how long the VC status has remained
                         unchanged.";
                    config "false";
                    type string {
```

```
                            length "1..80";
                        }
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the instance keeps the Up state.
                         If the PW is currently in the Down state, the value
                         is 0.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf totalUpTime {
                    description
                        "Indicates the total duration the instance is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }

                container vpwsPws {
                    uses vpwsPws;
                }

            }

        }
    }

    container l2vpnvpls {

        container vplsStatisticInfo {

            container vplsInstStatisticsInfo {

                leaf totalVsiNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf vsiUpNum {
                    config "false";
                    type uint32;
                }
                leaf vsiDownNum {
                    config "false";
                    type uint32;
                }
```

```
                leaf ldpModeNum {
                    config "false";
                    type uint32;
                }
                leaf bgpVsiNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdVsiNum {
                    config "false";
                    type uint32;
                }
                leaf unspecifiedNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsPwStatisticsInfo {

                leaf totalPwNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf upPwNum {
                    config "false";
                    type uint32;
                }
                leaf downPwNum {
                    config "false";
                    type uint32;
                }
                leaf ldpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdPwNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsAcStatisticsInfo {
```

```
                leaf totalVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf upVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf downVplsAcNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsTnlRefInfos {

                config "false";
                list vplsTnlRefInfo {

                    key "tnlPolicyName";

                    leaf tnlPolicyName {
                        description "None";
                        config "false";
                        type string {
                            length "1..39";
                        }
                    }
                    container tnlVsiRefInfos {

                        list tnlVsiRefInfo {

                             key "instanceName";

                             leaf instanceName {
                                config "false";
                                type string {
                                    length "1..31";
                                }
                            }
                        }

                    }

                }

            }
```

```
            container vplsLoopDetectStaticInfo {

                 leaf totalVplsLoopDetectNum {
                    config "false";
                    type uint32;
                }
            }

        }

        container vplsInstances {

            list vplsInstance {

                key "instanceName";

                leaf instanceName {
                    description
                        "Specifies VPLS instance name.";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf description {
                    description
                        "Specify the VPLS instance description.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf memberDiscoveryMode {
                    description
                        "The VPLS member discovery mode for a created VSI.";
                    config "true";
                    default "default";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum auto {
                            value "1";
                            description "auto:";
                        }
                        enum static {
                            value "2";
```

```
                        description "static:";
                    }
                    enum bdmode {
                        value "3";
                        description "bd mode:";
                    }
                }
            }
            leaf encapsulateType {
                description "VPWS instance encapsulation type.";
                config "true";
                default "vlan";
                type pw-encapsulation;
            }
            leaf mtuValue {
                description
                    "MTU specified in dynamic PW signaling negotiation.";
                config "true";
                default "1500";
                type uint16 {
                    range "328..65535";
                }
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the VSI. If no
                     name is specified for a tunnel policy, the default
                     tunnel policy is adopted. The LSP tunnel is
                     preferred and only one LSP is used for load balancing
                     in the default tunnel policy. If the name of the
                     tunnel policy is specified but no tunnel policy is
                     configured, the default tunnel policy is still adopted.
                     ";
                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf shutdown {
                description
                    "Sometimes, because of service debugging or service
                     halt, a VSI can be disabled for function
                     modification.";
                config "true";
                default "false";
                type boolean;
            }
            leaf isolateSpoken {
```

```
                    description
                        "The isolate spoken command enables forwarding
                         isolation between AC interfaces, between UPE
                         PWs, and between ACs and UPE PWs on a VSI. The
                         undo isolate spoken command disables forwarding
                         isolation";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf unknownUnicastAction {
                    description
                        "Specifies the processing mode for received unknown
                         unicast frames.";
                    config "true";
                    type enumeration {
                        enum broadcast {
                            value "0";
                            description "broadcast:";
                        }
                        enum drop {
                            value "1";
                            description "drop:";
                        }
                        enum local-handle {
                            value "2";
                            description "local-handle:";
                        }
                        enum drop-learn {
                            value "3";
                            description "drop-learn:";
                        }
                    }
                }
                leaf macLearnEnable {
                    description
                        "Enables MAC address learning on a VSI.";
                    config "true";
                    default "true";
                    type boolean;
                }
                leaf macLearnStyle {
                    description
                        "Sets the MAC address learning style of a VSI.By
                         default, MAC address learning style is unqualify.
                         Currently, the VRP supports only the unqualified
                         mode.";
```

```
                    config "true";
                    default "unqualify";
                    type enumeration {
                        enum qualify {
                            value "0";
                            description "qualify:";
                        }
                        enum unqualify {
                            value "1";
                            description "unqualify:";
                        }
                    }
                }
                leaf macAgeTimer {
                    description
                        "Sets the aging time of MAC address entries in a VSI.
                         By default, the aging time of MAC address entries in
                         a VSI is the global aging time. You can set the global
                         aging time by the command mac-address aging-time
                         (system view).";
                    config "true";
                    type uint32 {
                        range "0..1000000";
                    }
                }
                leaf totalAcService {
                    description
                        "Total number of interface in the instance.";
                    config "false";
                    type uint32;
                }
                leaf createTime {
                    description
                        "Indicates how long the VSI has been created for.";
                    config "false";
                    type string {
                        length "1..60";
                    }
                }
                leaf vsiState {
                    description "VPLS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
```

```
                        value "1";
                        description "up:";
                    }
                    enum adminDown {
                        value "2";
                        description "adminDown:";
                    }
                }
            }
            leaf ignoreAcStateEffect {
                description
                    "After the ignore-ac-state command is configured,
                     the VSI status is not subject to changes in the
                     status of the AC. That is, a VSI can go Up even
                     if no AC is connected to the VSI.";

                config "false";
                default "false";
                type boolean;
            }

            container vsiPipe {

                leaf pipeMode {
                    description "Pipe mode";
                    config "true";
                    default "uniform";
                    type enumeration {
                        enum pipe {
                            value "0";
                            description "pipe:";
                        }
                        enum shortPipe {
                            value "1";
                            description "shortPipe:";
                        }
                        enum uniform {
                            value "2";
                            description "uniform:";
                        }
                    }
                }
                leaf serviceClass {
                    description "service class";
                    config "true";
                    default "be";
                    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:";
                    }
                }
            }
            leaf color {
                description "service color";
                config "true";
                default "green";
                type enumeration {
                    enum green {
                        value "0";
                        description "green:";
                    }
                    enum yellow {
                        value "1";
                        description "yellow:";
                    }
                    enum red {
                        value "2";
```

```
                            description "red:";
                        }
                    }
                }
                leaf dsName {
                    description "domain name";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
            }

            container vplsLdpInst {

                leaf vsiId {
                    description
                        "After an ID is set for a VSI, it cannot be
                         changed. Different VSIs have different IDs.";
                    config "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf macWithdraw {
                    description
                        "Configures a VSI to delete the local MAC
                         addresses and informs all the remote peers
                         of the deletion when an AC fault or a UPE
                         fault occurs and the VSI remains Up.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf ifChgWithdraw {
                    description
                        "Configures PEs to send LDP MAC Withdraw
                         messages to all peers when the status of
                         the AC interface bound to the VSI changes.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf upeUpeMacWithdraw {
                    description
                        "Configures an NPE to forward the LDP MAC
                         Withdraw messages received from a UPE to
                         other UPEs.";
```

```
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf upeNpeMacWithdraw {
                        description
                            "Configures an NPE to forward the LDP MAC
                             Withdraw messages received from UPEs to
                             other NPEs.";
                        config "true";
                        default "false";
                        type boolean;
                    }
                    leaf npeUpeMacWithdraw {
                        description
                            "Configures an NPE to forward the LDP MAC
                             Withdraw messages received from other NPEs
                             to UPEs.";
                        config "true";
                        type boolean;
                    }
                    container vplsLdpPws {

                        list vplsLdpPw {

                            key "peerIp pwId pwEncapType";

                            leaf peerIp {
                                description "Specifies the LSR ID of the peer PE
.";
                                config "true";
                                type inet:ip-address;
                            }
                            leaf pwId {
                                description
                                    "Indicates the identifier of the PW. Default
,
                                     we may use vsiId as the pwId. Sometimes we
may
                                     create pw to different pw that the pwId not
                                     match our vsiId, so it must specify the pwI
d.";

                                config "true";
                                type uint32 {
                                    range "1..4294967295";
                                }
                            }
                            leaf pwEncapType {
                                description
                                    "Indicates the encapsulation of the PW.
                                     Default, we may use encapsulateType as
```

```
                    the pwEncapType. Sometimes we may create
                    pw that the pwEncapType not match our
                    encapsulateType, so it must specify the
                    pwEncapType.";
                config "true";
                type pw-encapsulation;
            }
            leaf pwRole {
                description
                    "VPLS pw role:primary, secondary.";
                config "true";
                default "primary";
                type pw-role;
            }
            leaf pwName {
                description
                    "Specifies the name of a PW, which is used
                     to distinguish the PW from other PWs. The
                     PW name must be unique in the same VSI,
                     but can be the same as the PW names in
                     other VSIs. ";
                config "true";
                type string {
                    length "1..15";
                }
            }
            leaf ifParaVCCV {
                description
                    "Deletes the VCCV byte (an interface
                     parameter) in the Mapping packet.";
                config "true";
                default "true";
                type boolean;
            }
            leaf isUpe {
                description "set VPLS PW as upe.";
                config "true";
                default "false";
                type boolean;
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the
                     VPLS PW. If no name is specified for a
                     tunnel policy, the default tunnel policy
                     is adopted. The LSP tunnel is preferred
                     and only one LSP is used for load
                     balancing in the default tunnel policy.
```

```
                                    If the name of the tunnel policy is
                                    specified but no tunnel policy is
                                    configured, the default tunnel policy is
                                    still adopted.";
                                config "true";
                                type string {
                                    length "1..39";
                                }
                            }

                            container vplsLdpPwInfo {
                                config "false";
                                uses vplsPwInfo;
                            }
                        }

                    }
                }

                container vplsBgpAdInst {

                    leaf vplsId {
                        description
                            "Specifies the vpls id. The value is a
                             case-sensitive string of 3 to 21 characters
                             without blank space.";
                        config "true";
                        type string {
                            length "1..21";
                        }
                    }

                    leaf bgpAdRd {
                        description
                            "Specifies the Route Distinguisher. The value is
                             a case-sensitive string of 3 to 21 characters
                             without blank space.";
                        config "false";
                        type string {
                            length "1..21";
                        }
                    }

                    leaf vsiId {
                        description
                            "Specifies the negotiation ip address of the
                             local PE.";
                        config "false";
```

```
                    type inet:ip-address;
                }

                container vpnTargets {
                    description "BGP-AD vpn-targets.";
                    list vpnTarget {
                        key "vpnRTValue";
                        description "BGP AD vpn targets.";

                        leaf vpnRTValue {

                          description
                            "Vpn-target:
                             adds VPN target extended community attribute
                             to the export or import VPN target extended
                             community list. The vpn-target can be
                             expressed in either of the following formats:
                             (1)16-bit AS number:32-bit user-defined number
                                For example, 1:3. The AS number ranges from
                                0 to 65535. The user-defined number ranges
                                from 0 to 4294967295. The AS number and the
                                user-defined number cannot be 0s at the same
                                time. That is, a VPN target cannot be 0:0.
                             (2)32-bit IP address:16-bit user-defined number
                                For example, 192.168.122.15:1. The IP addres
s

                                ranges from 0.0.0.0 to 255.255.255.255. The
                                user-defined number ranges from 0 to 65535."
;

                          mandatory "true";
                          type string {
                            length "3..21";
                          }
                        }

                        leaf vrfRTType {
                          description
                            "Specifies the vpn target type,
                             export-extcommunity: specifies the extended
                             community attributes carried in routing
                             information to be sent. import-extcommunity:
                             receives routing information carrying
                             specified extended community attributes.";
                           mandatory "true";
                           type enumeration {
                           enum export_extcommunity {
                             value "0";
                             description "export-extcommunity:";
                           }
```

```
                        enum import_extcommunity {
                          value "1";
                          description "import-extcommunity:";
                        }
                        enum both {
                          value "2";
                          description
                            "export-extcommunity &
                             import-extcommunity:";
                        }
                      }
                    }
                  }
                }

                container bgpAdPeerInfos {
                    config "false";

                    list bgpAdPeerInfo {

                        key "peerRouterID";
                        leaf peerRouterID {
                            description
                                "The Router ID of the remote router.";
                            type inet:ip-address;
                        }

                        leaf vplsId {
                            description
                                "The vpls id. The value is a case-sensitive
                                 string of 3 to 21 characters without blank
                                 space.";
                            type string {
                                length "1..21";
                            }
                        }

                        leaf sourceAII {
                            description
                                "The source AII of the remote PE.";
                            type inet:ip-address;
                        }

                        leaf targetAII {
                            description
                                "The target AII of the remote PE.";
                            type inet:ip-address;
                        }
```

```
                            leaf peerType {
                              description
                                "Specifies the peer type.Static,Dynamic.";

                                 type enumeration {
                                   enum static {
                                     value "0";
                                     description "Static pw";
                                   }
                                   enum dynamic {
                                     value "1";
                                     description "Dynamic pw";
                                   }
                              }
                            }

                            container bgpAdPwInfo {
                                config "false";

                                uses vplsPwInfo;
                            }

                        }
                    }

                }

                container vplsBgpInst {

                    leaf bgpRd {
                        description
                            "Specifies the Route Distinguisher. The value is a
                            case-sensitive string of 3 to 21 characters without
                            blank space.";
                        type string {
                            length "1..21";
                        }
                    }

                    leaf ignoreMtu {
                        description
                            "Ignore the mtu when negotiate pw.";
                        type boolean;
                    }

                    container vpnTargets {
                        description "BGP vpn-targets";
                        list vpnTarget {
```

```
                key "vpnRTValue";
                description
                    "BGP vpn targets";

                leaf vpnRTValue {

                  description
                    "Vpn-target: adds VPN target extended community
                     attribute to the export or import VPN target
                     extended community list. The vpn-target can be
                     expressed in either of the following formats:
                    (1)16-bit AS number:32-bit user-defined number
                       For example, 1:3. The AS number ranges from
                       0 to 65535. The user-defined number ranges
                       from 0 to 4294967295. The AS number and the
                       user-defined number cannot be 0s at the same
                       time. That is, a VPN target cannot be 0:0.
                    (2)32-bit IP address:16-bit user-defined number
                       For example, 192.168.122.15:1. The IP address
                       ranges from 0.0.0.0 to 255.255.255.255. The
                       user-defined number ranges from 0 to 65535.";

                  mandatory "true";
                  type string {
                    length "3..21";
                  }
                }

                leaf vrfRTType {
                  description
                    "Specifies the vpn target type,
                     export-extcommunity: specifies the extended
                     community attributes carried in routing
                     information to be sent.
                     import-extcommunity: receives routing
                     information carrying specified extended
                     community attributes.";

                      mandatory "true";
                      type enumeration {
                        enum export_extcommunity {
                          value "0";
                          description "export-extcommunity:";
                        }
                        enum import_extcommunity {
                          value "1";
                          description "import-extcommunity:";
                        }
```

```
                        enum both {
                          value "2";
                          description "export-extcommunity &
                                       import-extcommunity:";
                        }
                    }
                }
            }
        }

        container bgpSite {

            leaf siteId {
                description
                    "Specifies the ID of the site.";
                mandatory "true";
                type uint16 {
                    range "1..65535";
                }
            }
            leaf siteRange {
                description
                    "Specifies the ID of the site range.";
                type uint16 {
                    range "1..65535";
                }
            }
            leaf defaultOffset {
                description
                    "Specifies the default offset of the site ID.";
                type uint8 {
                    range "0..1";
                }
            }
        }

        container bgpPeerInfos {
            config "false";

            list bgpPeerInfo {

                key "siteId";

                leaf siteId {
                    description
                        "The site id of the peer.";
                    type uint16 {
                        range "1..65535";
```

```
                        }
                    }

                    container bgpPwInfo {
                        config "false";

                        uses vplsPwInfo;
                    }

                }
            }

        }

        container vplsAcs {

            list vplsAc {

                key "interfaceName";

                leaf interfaceName {
                    description "Specifies the AC interface name. ";
                    config "true";
                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf hubModeEnable {
                    description
                        "Change the VSI attribute of the local
                         interface from spoke to hub.By default,
                         the AC side of a VSI has the attribute
                         of spoke, and the PW side of a VSI has
                         the attribute of hub.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf state {
                    description
                        "Indicates the status of the AC.";
                    config "false";
                    type ifState;
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the AC keeps the Up state.
                         If the AC is currently in the Down state, the
```

```
                    value is 0.";
                config "false";
                type yang:date-and-time;
            }
            leaf totalUpTime {
                description
                    "Indicates the total duration the AC is Up.";
                config "false";
                type string {
                    length "1..60";
                }
            }
            container ifLinkProtocolTran {

                description "lacp status";

                leaf protocolLacp {
                    description "lacp status";
                    config "true";
                    type enumeration {
                        enum enable {
                            value "0";
                            description "enable:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                    }
                }
                leaf protocolLldp {
                    description "lldp status";
                    config "true";
                    type enumeration {
                        enum enable {
                            value "0";
                            description "enable:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                    }
                }
                leaf protocolBpdu {
                    description "bpdu status";
                    config "true";
                    type enumeration {
```

```
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolCdp {
                            description "cdp status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolUdld {
                            description "udld status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                    }
                }
            }

            container vplsLoopDetectInfo {

                description "L2vpn admin vsi binding class.";

                leaf lastLoopType {
```

```
                        description
                            "Indicates the last mac withdraw loop type.";
                        config "false";
                        type enumeration {
                            enum detect-loop {
                                value "0";
                                description "detect loop:";
                            }
                            enum exceed-max-hop {
                                value "1";
                                description "exceed max hop:";
                            }
                        }
                    }
                    leaf sendPeer {
                        description
                            "Indicates the mac withdraw send peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf receivePeer {
                        description
                            "Indicates the mac withdrawreceive peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf lastLoopTime {
                        description
                            "Indicates the last mac withdraw loop time.";
                        config "false";
                        type string {
                            length "1..80";
                        }
                    }
                }

            }

        }

    }
}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

8. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Haibo Wang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: rainsword.wang@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com