

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 25 December 2021

G. Harris, Ed.  
M. Richardson  
Sandelman  
23 June 2021

PCAP Capture File Format  
draft-gharris-opsawg-pcap-02

Abstract

This document describes the format used by the libpcap library to record captured packets to a file. Programs using the libpcap library to read and write those files, and thus reading and writing files in that format, include tcpdump.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the OPSAWG Working Group mailing list ([opsawg@ietf.org](mailto:opsawg@ietf.org)), which is archived at <https://mailarchive.ietf.org/arch/browse/opsawg/>.

Source for this draft and an issue tracker can be found at <https://github.com/pcapng/pcapng>.

Status of This Memo

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

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

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

This Internet-Draft will expire on 25 December 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. General File Structure . . . . .	3
4. File Header . . . . .	3
5. Packet Record . . . . .	5
6. Recommended File Name Extension: .pcap . . . . .	7
7. Security Considerations . . . . .	7
8. IANA Considerations . . . . .	7
8.1. Media-Type Registry . . . . .	7
8.1.1. application/pcap . . . . .	7
8.2. LinkType Registry . . . . .	8
9. Contributors . . . . .	34
10. Acknowledgments . . . . .	35
11. References . . . . .	35
11.1. Normative References . . . . .	35
11.2. Informative References . . . . .	35
Authors' Addresses . . . . .	35

## 1. Introduction

In the late 1980's, Van Jacobson, Steve McCanne, and others at the Network Research Group at Lawrence Berkeley National Laboratory developed the tcpdump program to capture and dissect network traces. The code to capture traffic, using low-level mechanisms in various operating systems, and to read and write network traces to a file was later put into a library named libpcap.

This document describes the format used by tcpdump, and other programs using libpcap, to read and write network traces.

## 2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. General File Structure

A capture file begins with a File Header, followed by zero or more Packet Records, one per packet.

All fields in the File Header and in Packet Records will always be saved according to the characteristics (little endian / big endian) of the capturing machine. This refers to all the fields that are saved as numbers and that span over two or more octets.

The approach of having the file saved in the native format of the generating host is more efficient because it avoids translation of data when reading / writing on the host itself, which is the most common case when generating/processing capture captures.

The packets are shown in traditional IETF diagram, with the bits numbered from the left to the right. The bit numbering does not reflect the binary value position, as IETF protocols are traditionally in big-endian network-byte order. The most significant bit is therefore on the left in this diagram as if the file is being stored on a big-endian system.

## 4. File Header

The File Header has the following format:

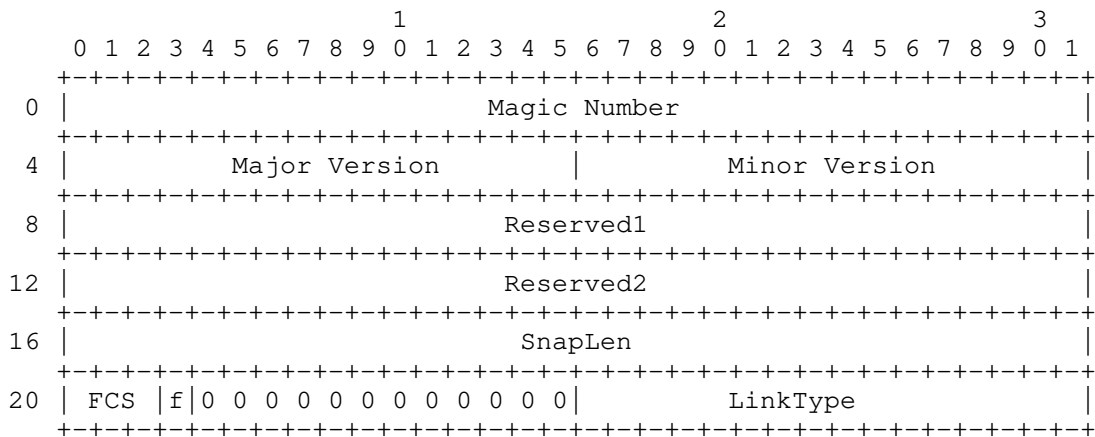


Figure 1: File Header

The File Header length is 24 octets.

The meaning of the fields in the File Header is:

**Magic Number (32 bits):** an unsigned magic number, whose value is either the hexadecimal number 0xA1B2C3D4 or the hexadecimal number 0xA1B23C4D.

If the value is 0xA1B2C3D4, time stamps in Packet Records (see Figure 2) are in seconds and microseconds; if it is 0xA1B23C4D, time stamps in Packet Records are in seconds and nanoseconds.

These numbers can be used to distinguish sessions that have been saved on little-endian machines from the ones saved on big-endian machines, and to heuristically identify pcap files.

**Major Version (16 bits):** an unsigned value, giving the number of the current major version of the format. The value for the current version of the format is 2. This value should change if the format changes in such a way that code that reads the new format could not read the old format (i.e., code to read both formats would have to check the version number and use different code paths for the two formats) and code that reads the old format could not read the new format.

**Minor Version (16 bits):** an unsigned value, giving the number of the

current minor version of the format. The value is for the current version of the format is 4. This value should change if the format changes in such a way that code that reads the new format could read the old format without checking the version number but code that reads the old format could not read all files in the new format.

Reserved1 (32 bits): not used - SHOULD be filled with 0 by pcap file writers, and MUST be ignored by pcap file readers. This value was documented by some older implementations as "gmt to local correction". Some older pcap file writers stored non-zero values in this field.

Reserved2 (32 bits): not used - SHOULD be filled with 0 by pcap file writers, and MUST be ignored by pcap file readers. This value was documented by some older implementations as "accuracy of timestamps". Some older pcap file writers stored non-zero values in this field.

SnapLen (32 bits): an unsigned value indicating the maximum number of octets captured from each packet. The portion of each packet that exceeds this value will not be stored in the file. This value MUST NOT be zero; if no limit was specified, the value should be a number greater than or equal to the largest packet length in the file.

LinkType (16 bits): a 16-bit unsigned value that defines the link layer type of packets in the file. This field is defined in the Section 8.2 IANA registry.

Frame Cyclic Sequence (FCS) present (4 bits): if the "f" bit is set, then the 3 FCS bits provide the number of 16-bit (2 byte) words of FCS that are appended to each packet.

valid values are between 0 and 7, with ethernet typically having a length of 4 bytes, or a value of 2.

The bits marked as zero MUST be set to zero by pcap writers, and MUST be ignored by pcap readers.

## 5. Packet Record

A Packet Record is the standard container for storing the packets coming from the network.

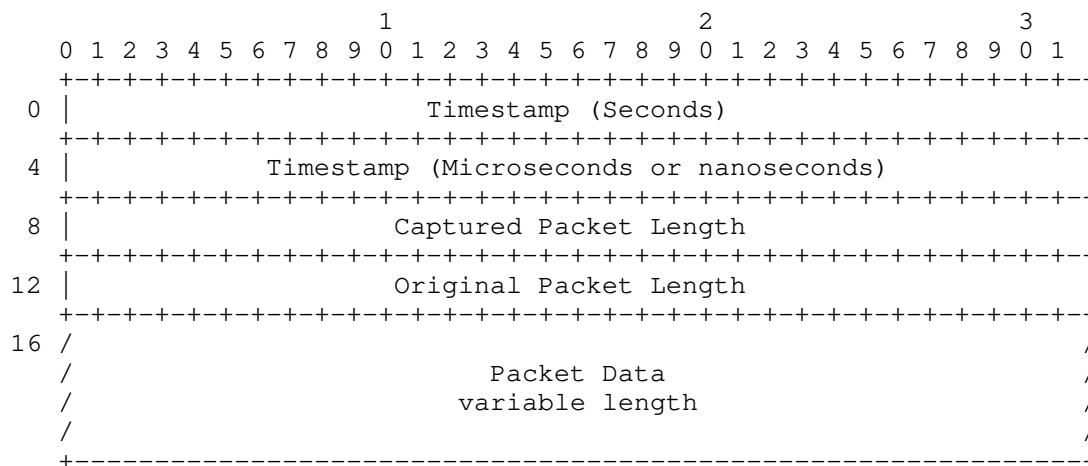


Figure 2: Packet Record

The Packet Header length is 16 octets.

The meaning of the fields in the Packet Record is:

Timestamp (Seconds) and Timestamp (Microseconds or nanoseconds): seconds and fraction of a seconds values of a timestamp.

The seconds value is a 32-bit unsigned integer that represents the number of seconds that have elapsed since 1970-01-01 00:00:00 UTC, and the microseconds or nanoseconds value represents the number of microseconds or nanoseconds that have elapsed since that seconds.

Whether the value represents microseconds or nanoseconds is specified by the magic number in the File Header.

Captured Packet Length (32 bits): an unsigned value that indicates the number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 1).

Original Packet Length (32 bits): an unsigned value that indicates the actual length of the packet when it was transmitted on the network. It can be different from the Captured Packet Length if the packet has been truncated by the capture process.

Packet Data: the data coming from the network, including link-layer

headers. The actual length of this field is Captured Packet Length. The format of the link-layer headers depends on the LinkType field specified in the file header (see Figure 1) and it is specified in the entry for that format in [LINKTYPES].

## 6. Recommended File Name Extension: .pcap

The recommended file name extension for the "PCAP Capture File Format" specified in this document is ".pcap".

On Windows and macOS, files are distinguished by an extension to their filename. Such an extension is technically not actually required, as applications should be able to automatically detect the pcap file format through the "magic bytes" at the beginning of the file, as some other UN\*X desktop environments do. However, using name extensions makes it easier to work with files (e.g. visually distinguish file formats) so it is recommended - though not required - to use .pcap as the name extension for files following this specification.

Please note: To avoid confusion (such as the current usage of .cap for a plethora of different capture file formats) file name extensions other than .pcap should be avoided.

There is new work to create the PCAP Next Generation capture File Format (see [I-D.tuexen-opsawg-pcapng]). The new file format is not compatible with this specification, but many programs read both transparently. Files of that type will usually start with a Section Header Block, with a magic number of 0x0A0D0D0A.

## 7. Security Considerations

TBD.

## 8. IANA Considerations

This document requires the following IANA actions:

### 8.1. Media-Type Registry

This section registers the the 'application/pcap' in the "Media Types" registry. These media types are used to indicate that the content is packet capture as described in this document.

#### 8.1.1. application/pcap

Type name: application  
Subtype name: pcap  
Required parameters: none  
Optional parameters: none  
Encoding considerations: PCAP files contain network packets  
Security considerations: See Security Considerations, Section  
Interoperability considerations: The format is designed to be broadly interoperable.  
Published specification: THIS RFC.  
Applications that use this media type: tcpdump, Wireshark, others.  
Additional information:  
    Magic number(s): 0xA1B2C3D4, and 0xA1B23C4D in both endian orders  
    File extension(s): .pcap  
    Macintosh file type code(s): none  
Person & email address to contact for further information: The Tcpdump Group, [www.tcpdump.org](http://www.tcpdump.org)  
Intended usage: LIMITED  
Restrictions on usage: NONE  
Author: Guy Harris and Michael Richardson  
Change controller: The Tcpdump Group  
Provisional registration? (standards tree only): NO

## 8.2. LinkType Registry

IANA is requested to create a new Registry entitled: "The PCAP Registry", and within that Registry to create a table called: "PCAP LinkType List".

The LinkType Registry is a table of 16-bit numbers. The Registry has three sections with different [RFC8126] rules:

- \* values from 0 to 32767 are marked as Specification Required.
- \* values from 32768 to 65000 are marked as First-Come First-Served.
- \* values from 65000 to 65535 are marked as Private Use.

The Registry has four columns: the symbolic name (LINKTYPE\_something), the integer value, a very short description, and the document/requestor reference.

The Registry shall be populated as follows in the table below. In each case here, the reference should be <http://www.tcpdump.org/linktypes.html>, which is not repeated.

The initial value of table is based upon the Link type list maintained by libpcap, and published on the [tcpdump.org](http://www.tcpdump.org) web site as <http://www.tcpdump.org/linktypes.html>.



There is often an associated DLT value which are often identical in value, but not universally so. DLT values are associated with specific operation system captures, and are operating system specific.

LINKTYPE name	LINKTYPE value	description
LINKTYPE_NULL	0	BSD loopback encapsulation
LINKTYPE_ETHERNET	1	IEEE 802.3 Ethernet
LINKTYPE_EXP_ETHERNET	2	Xerox experimental 3Mb Ethernet
LINKTYPE_AX25	3	AX.25 packet
LINKTYPE_PRONET	4	Reserved for PRONET
LINKTYPE_CHAOS	5	Reserved for MIT CHAOSNET
LINKTYPE_IEEE802_5	6	IEEE 802.5 Token Ring
LINKTYPE_ARCNET_BSD	7	ARCNET Data Packets with BSD encapsulation
LINKTYPE_SLIP	8	SLIP, w/ LINKTYPE_SLIP header.
LINKTYPE_PPP	9	PPP, as per RFC 1661/RFC 1662
LINKTYPE_FDDI	10	FDDI: per ANSI INCITS 239-1994.
not to be used	11-49	Do not use these values
LINKTYPE_PPP_HDLC	50	PPP in HDLC-like framing, as per RFC 1662

LINKTYPE_PPP_ETHER	51	PPPoE; per RFC 2516
not to be used	52-98	Do not use these values
LINKTYPE_SYMANTEC_FIREWALL	99	Reserved for Symantec Enterprise Firewall
LINKTYPE_ATM_RFC1483	100	RFC 1483 LLC/SNAP-encapsulated ATM
LINKTYPE_RAW	101	Raw IP; begins with an IPv4 or IPv6 header
LINKTYPE_SLIP_BSDOS	102	Reserved for BSD/OS SLIP BPF header
LINKTYPE_PPP_BSDOS	103	Reserved for BSD/OS PPP BPF header
LINKTYPE_C_HDLC	104	Cisco PPP with HDLC framing, as per section 4.3.1 of RFC 1547
LINKTYPE_IEEE802_11	105	IEEE 802.11 wireless LAN.
LINKTYPE_ATM_CLIP	106	ATM Classical IP, with no header preceding IP
LINKTYPE_FRELAY	107	Frame Relay LAPF frames
LINKTYPE_LOOP	108	OpenBSD loopback encapsulation
LINKTYPE_ENC	109	Reserved for OpenBSD IPSEC encapsulation
LINKTYPE_LANE8023	110	Reserved for ATM LANE + 802.3
LINKTYPE_HIPPI	111	Reserved for NetBSD HIPPI

LINKTYPE_HDLC	112	Reserved for NetBSD HDLC framing
LINKTYPE_LINUX_SLL	113	Linux "cooked" capture encapsulation
LINKTYPE_LTALK	114	Apple LocalTalk
LINKTYPE_ECONET	115	Reserved for Acorn Econet
LINKTYPE_IPFILTER	116	Reserved for OpenBSD ipfilter
LINKTYPE_PFLOG	117	OpenBSD pflog; "struct pfloghdr" structure
LINKTYPE_CISCO_IOS	118	Reserved for Cisco-internal use
LINKTYPE_IEEE802_11_PRISM	119	Prism monitor mode
LINKTYPE_IEEE802_11_AIRONET	120	Reserved for 802.11 + FreeBSD Aironet radio metadata
LINKTYPE_HHDLC	121	Reserved for Siemens HiPath HDLC
LINKTYPE_IP_OVER_FC	122	RFC 2625 IP-over-Fibre Channel
LINKTYPE_SUNATM	123	ATM traffic, / per SunATM devices
LINKTYPE_RIO	124	Reserved for RapidIO
LINKTYPE_PCI_EXP	125	Reserved for PCI Express
LINKTYPE_AURORA	126	Reserved for Xilinx Aurora link layer
LINKTYPE_IEEE802_11_RADIOTAP	127	Radiotap

		header[Radiotap], followed by an 802.11 header
LINKTYPE_TZSP	128	Reserved for Tazmen Sniffer Protocol
LINKTYPE_ARCNET_LINUX	129	ARCNET Data Packets, per RFC 1051 frames w/variations
LINKTYPE_JUNIPER_MLPPP	130	Reserved for Juniper Networks
LINKTYPE_JUNIPER_MLFR	131	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ES	132	Reserved for Juniper Networks
LINKTYPE_JUNIPER_GGSN	133	Reserved for Juniper Networks
LINKTYPE_JUNIPER_MFR	134	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ATM2	135	Reserved for Juniper Networks
LINKTYPE_JUNIPER_SERVICES	136	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ATM1	137	Reserved for Juniper Networks
LINKTYPE_APPLE_IP_OVER_IEEE1394	138	Apple IP-over-IEEE 1394 cooked header
LINKTYPE_MTP2_WITH_PHDR	139	Signaling System 7 (SS7) Message Transfer Part Level ITU-T Q.703
LINKTYPE_MTP2	140	SS7 Level 2, Q.703
LINKTYPE_MTP3	141	SS7 Level 3, Q.704
LINKTYPE_SCCP	142	SS7 Control Part,

		ITU-T Q.711/Q.712/Q.713/ Q.714
LINKTYPE_DOCSIS	143	DOCSIS MAC frames, DOCSIS 3.1
LINKTYPE_LINUX_IRDA	144	Linux-IrDA packets w/ LINKTYPE_LINUX_IRDA header
LINKTYPE_IBM_SP	145	Reserved for IBM SP switch
LINKTYPE_IBM_SN	146	Reserved for IBM Next Federation switch
LINKTYPE_RESERVED_01	147	For private use
LINKTYPE_RESERVED_02	148	For private use
LINKTYPE_RESERVED_03	149	For private use
LINKTYPE_RESERVED_04	150	For private use
LINKTYPE_RESERVED_05	151	For private use
LINKTYPE_RESERVED_06	152	For private use
LINKTYPE_RESERVED_07	153	For private use
LINKTYPE_RESERVED_08	154	For private use
LINKTYPE_RESERVED_09	155	For private use
LINKTYPE_RESERVED_10	156	For private use
LINKTYPE_RESERVED_11	157	For private use
LINKTYPE_RESERVED_12	158	For private use
LINKTYPE_RESERVED_13	159	For private use
LINKTYPE_RESERVED_14	160	For private use
LINKTYPE_RESERVED_15	161	For private use

LINKTYPE_RESERVED_16	162	For private use
LINKTYPE_IEEE802_11_AVS	163	AVS header[AVS], followed by an 802.11 header
LINKTYPE_JUNIPER_MONITOR	164	Reserved for Juniper Networks
LINKTYPE_BACNET_MS_TP	165	BACnet MS/TP frames, per 9.3 MS/TP Frame Format ANSI 135
LINKTYPE_PPP_PPPD	166	PPP in HDLC-like encapsulation, like LINKTYPE_PPP_HDLC, different stuffing
LINKTYPE_JUNIPER_PPPOE	167	Reserved for Juniper Networks
LINKTYPE_JUNIPER_PPPOE_ATM	168	Reserved for Juniper Networks
LINKTYPE_GPRS_LLC	169	General Packet Radio Service Logical Link Control, as per 3GPP TS 04.64
LINKTYPE_GPF_T	170	Transparent-mapped generic framing procedure, as specified by ITU-T Recommendation G.7041/Y.1303
LINKTYPE_GPF_F	171	Frame-mapped generic framing procedure, as specified by ITU-T Recommendation G.7041/Y.1303
LINKTYPE_GCOM_T1E1	172	Reserved for Gcom T1/E1 line monitoring equipment
LINKTYPE_GCOM_SERIAL	173	Reserved for Gcom

		T1/E1 line monitoring equipment
LINKTYPE_JUNIPER_PIC_PEER	174	Reserved for Juniper Networks
LINKTYPE_ERF_ETH	175	Endace ERF header followed by 802.3 Ethernet
LINKTYPE_ERF_POS	176	Endace ERF header followed by Packet-over-SONET
LINKTYPE_LINUX_LAPD	177	Link Access Procedures on the D Channel (LAPD) frames, as specified by ITU-T Recommendation Q.920 and ITU-T Recommendation Q.921, captured via vISDN, with a LINKTYPE_LINUX_LAPD header, followed by the Q.921 frame, starting with the address field.
LINKTYPE_JUNIPER_ETHER	178	Reserved for Juniper Networks
LINKTYPE_JUNIPER_PPP	179	Reserved for Juniper Networks
LINKTYPE_JUNIPER_FRELAY	180	Reserved for Juniper Networks
LINKTYPE_JUNIPER_CHDLC	181	Reserved for Juniper Networks
LINKTYPE_MFR	182	FRF.16.1 Multi-Link Frame Relay frames, beginning with an FRF.12 Interface fragmentation format fragmentation

		header.
LINKTYPE_JUNIPER_VP	182	Reserved for Juniper Networks
LINKTYPE_A653_ICM	185	Reserved for Arinc 653 Interpartition Communication messages
LINKTYPE_USB_FREEBSD	186	USB packets, beginning with a FreeBSD USB header
LINKTYPE_BLUETOOTH_HCI_H4	187	Bluetooth HCI UART transport layer; the frame contains an HCI packet indicator byte, as specified by the UART Transport Layer portion of the most recent Bluetooth Core specification , followed by an HCI packet of the specified packet type, as specified by the Host Controller Interface Functional Specification portion of the most recent Bluetooth Core Specification.
LINKTYPE_IEEE802_16_MAC_CPS	188	Reserved for IEEE 802.16 MAC Common Part Sublayer
LINKTYPE_USB_LINUX	189	USB packets, beginning with a Linux USB header, as specified by the struct usbmon_packet in the Documentation/usb/usbmon.txt file in



		the Linux source tree. Only the first 48 bytes of that header are present. All fields in the header are in host byte order. When performing a live capture, the host byte order is the byte order of the machine on which the packets are captured. When reading a pcap file, the byte order is the byte order for the file, as specified by the file's magic number; when reading a pcapng file, the byte order is the byte order for the section of the pcapng file, as specified by the Section Header Block.
LINKTYPE_CAN20B	190	Reserved for Controller Area Network (CAN) v. 2.0B packets
LINKTYPE_IEEE802_15_4_LINUX	191	IEEE 802.15.4, with address fields padded, as is done by Linux drivers
LINKTYPE_PPI	192	Per-Packet Information information, as specified by the Per-Packet Information Header Specification , followed by a packet

		with the LINKTYPE_ value specified by the pph_dlt field of that header.
LINKTYPE_IEEE802_16_MAC_CPS_RADIO	193	Reserved for 802.16 MAC Common Part Sublayer plus radio header
LINKTYPE_JUNIPER_ISM	194	Reserved for Juniper Networks
LINKTYPE_IEEE802_15_4_WITHFCS	195	IEEE 802.15.4 Low-Rate Wireless Networks, with each packet having the FCS at the end of the frame.
LINKTYPE_SITA	196	Various link-layer types, with a pseudo-header , for SITA
LINKTYPE_ERF	197	Various link-layer types, with a pseudo-header, for Endace DAG cards; encapsulates Endace ERF records.
LINKTYPE_RAIF1	198	Reserved for Ethernet packets captured from a u10 Networks board
LINKTYPE_IPMB_KONTRON	199	Reserved for IPMB packet for IPMI, with a 2-byte header
LINKTYPE_JUNIPER_ST	200	Reserved for Juniper Networks
LINKTYPE_BLUETOOTH_HCI_H4_WITH_PHDR	201	Bluetooth HCI UART transport layer; the frame contains a 4-byte direction

		field, in network byte order (big-endian), the low-order bit of which is set if the frame was sent from the host to the controller and clear if the frame was received by the host from the controller, followed by an HCI packet indicator byte, as specified by the UART Transport Layer portion of the most recent Bluetooth Core specification, followed by an HCI packet of the specified packet type, as specified by the Host Controller Interface Functional Specification portion of the most recent Bluetooth Core Specification.
LINKTYPE_AX25_KISS	202	AX.25 packet, with a 1-byte KISS header containing a type indicator.
LINKTYPE_LAPD	203	Link Access Procedures on the D Channel (LAPD) frames, as specified by ITU-T Recommendation Q.920 and ITU-T Recommendation Q.921, starting with the address field, with no pseudo-header.

LINKTYPE_PPP_WITH_DIR	204	PPP, as per RFC 1661 and RFC 1662 , preceded with a one-byte pseudo-header with a zero value meaning received by this host and a non-zero value meaning sent by this host; if the first 2 bytes are 0xff and 0x03, it's PPP in HDLC-like framing, with the PPP header following those two bytes, otherwise it's PPP without framing, and the packet begins with the PPP header. The data in the frame is not octet-stuffed or bit-stuffed.
LINKTYPE_C_HDLC_WITH_DIR	205	Cisco PPP with HDLC framing, as per section 4.3.1 of RFC 1547 , preceded with a one-byte pseudo-header with a zero value meaning received by this host and a non-zero value meaning sent by this host.
LINKTYPE_FRELAY_WITH_DIR	206	Frame Relay LAPF frames, beginning with a one-byte pseudo-header with a zero value meaning received by this host (DCE->DTE) and a non-zero value meaning sent by this host (DTE->DCE), followed by an ITU-T Recommendation Q.922

		LAPF header starting with the address field, and without an FCS at the end of the frame.
LINKTYPE_LAPB_WITH_DIR	207	Link Access Procedure, Balanced (LAPB), as specified by ITU-T Recommendation X.25, preceded with a one-byte pseudo-header with a zero value meaning received by this host (DCE->DTE) and a non-zero value meaning sent by this host (DTE->DCE).
Reserved	208	Reserved for an unspecified link-layer type
LINKTYPE_IPMB_LINUX	209	IPMB over an I2C circuit, with a Linux-specific pseudo-header
LINKTYPE_FLEXRAY	210	Reserved for FlexRay automotive bus
LINKTYPE_MOST	211	Reserved for Media Oriented Systems Transport (MOST) bus
LINKTYPE_LIN	212	Reserved for Local Interconnect Network (LIN) bus for vehicle networks
LINKTYPE_X2E_SERIAL	213	Reserved for X2E serial line captures
LINKTYPE_X2E_XORAYA	214	Reserved for X2E Xoraya data loggers

LINKTYPE_IEEE802_15_4_NONASK_PHY	215	IEEE 802.15.4 Low-Rate Wireless Networks, with each packet having the FCS at the end of the frame, and with the PHY-level data for the O-QPSK, BPSK, GFSK, MSK, and RCC DSS BPSK PHYs (4 octets of 0 as preamble, one octet of SFD, one octet of frame length + reserved bit) preceding the MAC-layer data (starting with the frame control field).
LINKTYPE_LINUX_EVDEV	216	Reserved for Linux evdev messages
LINKTYPE_GSMTAP_UM	217	Reserved for GSM Um interface, with gsmtap header
LINKTYPE_GSMTAP_ABIS	218	Reserved for GSM Abis interface, with gsmtap header
LINKTYPE_MPLS	219	MPLS packets with MPLS label as the header
LINKTYPE_USB_LINUX_MMAPPED	220	USB packets, beginning with a Linux USB header, as specified by the struct usbmon_packet in the Documentation/usb/usbmon.txt file in the Linux source tree. All 64 bytes of the header are present. All fields in the header are in

		host byte order. When performing a live capture, the host byte order is the byte order of the machine on which the packets are captured. When reading a pcap file, the byte order is the byte order for the file, as specified by the file's magic number; when reading a pcapng file, the byte order is the byte order for the section of the pcapng file, as specified by the Section Header Block. For isochronous transfers, the ndesc field specifies the number of isochronous descriptors that follow.
LINKTYPE_DECT	221	Reserved for DECT packets, with a pseudo-header
LINKTYPE_AOS	222	Reserved for OS Space Data Link Protocol
LINKTYPE_WIHART	223	Reserved for Wireless HART (Highway Addressable Remote Transducer)
LINKTYPE_FC_2	224	Fibre Channel FC-2 frames, beginning with a Frame_Header.

LINKTYPE_FC_2_WITH_FRAME_DELIMS	225	Fibre Channel FC-2 frames, beginning an encoding of the SOF, followed by a Frame_Header, and ending with an encoding of the SOF. The encodings represent the frame delimiters as 4-byte sequences representing the corresponding ordered sets, with K28.5 represented as 0xBC, and the D symbols as the corresponding byte values; for example, SOFi2, which is K28.5 - D21.5 - D1.2 - D21.2, is represented as 0xBC 0xB5 0x55 0x55.
LINKTYPE_IPNET	226	Solaris ipnet pseudo-header , followed by an IPv4 or IPv6 datagram.
LINKTYPE_CAN_SOCKETCAN	227	CAN (Controller Area Network) frames, with a pseudo-header followed by the frame payload.
LINKTYPE_IPV4	228	Raw IPv4; the packet begins with an IPv4 header.
LINKTYPE_IPV6	229	Raw IPv6; the packet begins with an IPv6 header.
LINKTYPE_IEEE802_15_4_NOFCS	230	IEEE 802.15.4 Low-Rate Wireless Network, without the FCS at the end of



		the frame.
LINKTYPE_DBUS	231	Raw D-Bus messages , starting with the endianness flag, followed by the message type, etc., but without the authentication handshake before the message sequence.
LINKTYPE_JUNIPER_VS	232	Reserved for Juniper Networks
LINKTYPE_JUNIPER_SRX_E2E	233	Reserved for Juniper Networks
LINKTYPE_JUNIPER_FIBRECHANNEL	234	Reserved for Juniper Networks
LINKTYPE_DVB_CI	235	DVB-CI (DVB Common Interface for communication between a PC Card module and a DVB receiver), with the message format specified by the PCAP format for DVB-CI specification
LINKTYPE_MUX27010	236	Variant of 3GPP TS 27.010 multiplexing protocol (similar to, but not the same as, 27.010).
LINKTYPE_STANAG_5066_D_PDU	237	D_PDUs as described by NATO standard STANAG 5066, starting with the synchronization sequence, and including both header and data CRCs. The current version of STANAG

		5066 is backwards-compatible with the 1.0.2 version , although newer versions are classified.
LINKTYPE_JUNIPER_ATM_CEMIC	238	Reserved for Juniper Networks
LINKTYPE_NFLOG	239	Linux netlink NETLINK NFLOG socket log messages.
LINKTYPE_NETANALYZER	240	Pseudo-header for Hilscher Gesellschaft fuer Systemautomation mbH netANALYZER devices , followed by an Ethernet frame, beginning with the MAC header and ending with the FCS.
LINKTYPE_NETANALYZER_TRANSPARENT	241	Pseudo-header for Hilscher Gesellschaft fuer Systemautomation mbH netANALYZER devices , followed by an Ethernet frame, beginning with the preamble, SFD, and MAC header, and ending with the FCS.
LINKTYPE_IPOIB	242	IP-over-InfiniBand, as specified by RFC 4391 section 6
LINKTYPE_MPEG_2_TS	243	MPEG-2 Transport Stream transport packets, as specified by ISO 13818-1/ ITU-T Recommendation H.222.0 (see table

		2-2 of section 2.4.3.2 Transport Stream packet layer).
LINKTYPE_NG40	244	Pseudo-header for ng4T GmbH's UMTS Iub/Iur-over-ATM and Iub/Iur-over-IP format as used by their ng40 protocol tester , followed by frames for the Frame Protocol as specified by 3GPP TS 25.427 for dedicated channels and 3GPP TS 25.435 for common/shared channels in the case of ATM AAL2 or UDP traffic, by SSCOP packets as specified by ITU-T Recommendation Q.2110 for ATM AAL5 traffic, and by NBAP packets for SCTP traffic.
LINKTYPE_NFC_LLCP	245	Pseudo-header for NFC LLCP packet captures , followed by frame data for the LLCP Protocol as specified by NFCForum-TS-LLCP_1.1
LINKTYPE_PFSYNC	246	Reserved for pfsync output
LINKTYPE_INFINIBAND	247	Raw InfiniBand frames, starting with the Local Routing Header, as specified in Chapter 5 Data packet format of InfiniBand[TM] Architectural

		Specification Release 1.2.1 Volume 1 - General Specifications
LINKTYPE_SCTP	248	SCTP packets, as defined by RFC 4960 , with no lower- level protocols such as IPv4 or IPv6.
LINKTYPE_USBPCAP	249	USB packets, beginning with a USBPcap header
LINKTYPE_RTAC_SERIAL	250	Serial-line packet header for the Schweitzer Engineering Laboratories RTAC product , followed by a payload for one of a number of industrial control protocols.
LINKTYPE_BLUETOOTH_LE_LL	251	Bluetooth Low Energy air interface Link Layer packets, in the format described in section 2.1 PACKET FORMAT of volume 6 of the Bluetooth Specification Version 4.0 (see PDF page 2200), but without the Preamble.
LINKTYPE_WIRESHARK_UPPER_PDU	252	Reserved for Wireshark
LINKTYPE_NETLINK	253	Linux Netlink capture encapsulation
LINKTYPE_BLUETOOTH_LINUX_MONITOR	254	Bluetooth Linux

		Monitor encapsulation of traffic for the BlueZ stack
LINKTYPE_BLUETOOTH_BREDR_BB	255	Bluetooth Basic Rate and Enhanced Data Rate baseband packets
LINKTYPE_BLUETOOTH_LE_LL_WITH_PHDR	256	Bluetooth Low Energy link-layer packets
LINKTYPE_PROFIBUS_DL	257	PROFIBUS data link layer packets, as specified by IEC standard 61158-4-3, beginning with the start delimiter, ending with the end delimiter, and including all octets between them.
LINKTYPE_PKTAP	258	Apple PKTAP capture encapsulation
LINKTYPE_EPON	259	Ethernet-over-passive-optical-network packets, starting with the last 6 octets of the modified preamble as specified by 65.1.3.2 Transmit in Clause 65 of Section 5 of IEEE 802.3 , followed immediately by an Ethernet frame.
LINKTYPE_IPMI_HPM_2	260	IPMI trace packets, as specified by Table 3-20 Trace Data Block Format in the PICMG HPM.2 specification The time stamps for

		packets in this format must match the time stamps in the Trace Data Blocks.
LINKTYPE_ZWAVE_R1_R2	261	Z-Wave RF profile R1 and R2 packets , as specified by ITU-T Recommendation G.9959 , with some MAC layer fields moved.
LINKTYPE_ZWAVE_R3	262	Z-Wave RF profile R3 packets , as specified by ITU-T Recommendation G.9959 , with some MAC layer fields moved.
LINKTYPE_WATTSTOPPER_DLM	263	Formats for WattStopper Digital Lighting Management (DLM) and Legrand Nitoo Open protocol common packet structure captures.
LINKTYPE_ISO_14443	264	Messages between ISO 14443 contactless smartcards (Proximity Integrated Circuit Card, PICC) and card readers (Proximity Coupling Device, PCD), with the message format specified by the PCAP format for ISO14443 specification
LINKTYPE_RDS	265	Radio data system (RDS) groups, as per IEC 62106,

		encapsulated in this form
LINKTYPE_USB_DARWIN	266	USB packets, beginning with a Darwin (macOS, etc.) USB header
LINKTYPE_OPENFLOW	267	Reserved for OpenBSD DLT_OPENFLOW
LINKTYPE_SDL	268	SDL packets, as specified by Chapter 1, DLC Links, section Synchronous Data Link Control (SDL) of Systems Network Architecture Formats, GA27-3136-20 , without the flag fields, zero-bit insertion, or Frame Check Sequence field, containing SNA path information units (PIUs) as the payload.
LINKTYPE_TI_LL_N_SNIFFER	269	Reserved for Texas Instruments protocol sniffer
LINKTYPE_LORATAP	270	LoRaTap pseudo-header , followed by the payload, which is typically the PHYPayload from the LoRaWan specification
LINKTYPE_VSOCK	271	Protocol for communication between host and guest machines in VMware and KVM hypervisors.

LINKTYPE_NORDIC_BLE	272	Messages to and from a Nordic Semiconductor nRF Sniffer for Bluetooth LE packets, beginning with a pseudo-header
LINKTYPE_DOCSIS31_XRA31	273	DOCSIS packets and bursts, preceded by a pseudo-header giving metadata about the packet
LINKTYPE_ETHERNET_MPACKET	274	mPackets, as specified by IEEE 802.3br Figure 99-4, starting with the preamble and always ending with a CRC field.
LINKTYPE_DISPLAYPORT_AUX	275	DisplayPort AUX channel monitoring data as specified by VESA DisplayPort (DP) Standard preceded by a pseudo-header
LINKTYPE_LINUX_SLL2	276	Linux cooked capture encapsulation v2
LINKTYPE_SERCOS_MONITOR	277	Reserved for Sercos Monitor
LINKTYPE_OPENVIZSLA	278	Openvizsla FPGA-based USB sniffer
LINKTYPE_EBHSCR	279	Elektrobit High Speed Capture and Replay (EBHSCR) format
LINKTYPE_VPP_DISPATCH	280	Records in traces from the <a href="http://fd.io">http://fd.io</a> VPP graph dispatch tracer, in the the



		graph dispatcher trace format
LINKTYPE_DSA_TAG_BRCM	281	Ethernet frames, with a switch tag inserted between the source address field and the type/length field in the Ethernet header.
LINKTYPE_DSA_TAG_BRCM_PREPEND	282	Ethernet frames, with a switch tag inserted before the destination address in the Ethernet header.
LINKTYPE_IEEE802_15_4_TAP	283	IEEE 802.15.4 Low- Rate Wireless Networks, with a pseudo-header containing TLVs with metadata preceding the 802.15.4 header.
LINKTYPE_DSA_TAG_DSA	284	Ethernet frames, with a switch tag inserted between the source address field and the type/length field in the Ethernet header.
LINKTYPE_DSA_TAG_EDSA	285	Ethernet frames, with a programmable Ethernet type switch tag inserted between the source address field and the type/ length field in the Ethernet header.
LINKTYPE_ELEE	286	Payload of lawful intercept packets using the ELEE protocol The packet begins with the ELEE

		header; it does not include any transport-layer or lower-layer headers for protocols used to transport ELEE packets.
LINKTYPE_Z_WAVE_SERIAL	287	Serial frames transmitted between a host and a Z-Wave chip over an RS-232 or USB serial connection, as described in section 5 of the Z-Wave Serial API Host Application Programming Guide
LINKTYPE_USB_2_0	288	USB 2.0, 1.1, or 1.0 packet, beginning with a PID, as described by Chapter 8 Protocol Layer of the the Universal Serial Bus Specification Revision 2.0
LINKTYPE_ATSC_ALP	289	ATSC Link-Layer Protocol frames, as described in section 5 of the A/330 Link-Layer Protocol specification, found at the ATSC 3.0 standards page , beginning with a Base Header

Table 1

## 9. Contributors

[Insert pcap developers etc. here].

## 10. Acknowledgments

The authors wish to thank [insert list here] and many others for their invaluable comments.

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 11.2. Informative References

- [I-D.tuexen-opsawg-pcapng] Tuexen, M., Risso, F., Bongertz, J., Combs, G., Harris, G., and M. Richardson, "PCAP Next Generation (pcapng) Capture File Format", Work in Progress, Internet-Draft, draft-tuexen-opsawg-pcapng-02, 28 September 2020, <<https://www.ietf.org/archive/id/draft-tuexen-opsawg-pcapng-02.txt>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [Radiotap] radiotap.org, "Radiotap Web site", n.d., <<http://www.radiotap.org/>>.
- [AVS] Peachy, S., "Archived AVS specification", n.d., <<http://web.archive.org/web/20040803232023/http://www.shaftnet.org/~pizza/software/capturefrm.txt>>.

## Authors' Addresses

Guy Harris (editor)

Email: [gharris@sonic.net](mailto:gharris@sonic.net)

Michael C. Richardson  
Sandelman Software Works Inc

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)  
URI: <http://www.sandelman.ca/>

OPSAWG  
Internet-Draft  
Intended status: Standards Track  
Expires: 14 November 2022

M. Boucadair, Ed.  
Orange  
O. Gonzalez de Dios, Ed.  
S. Barguil  
Telefonica  
L. Munoz  
Vodafone  
13 May 2022

A YANG Network Data Model for Layer 2 VPNs  
draft-ietf-opsawg-l2nm-16

Abstract

This document defines an L2VPN Network YANG Model (L2NM) which can be used to manage the provisioning of Layer 2 Virtual Private Network services within a network (e.g., service provider network). The L2NM complements the Layer 2 Service Model (L2SM) by providing a network-centric view of the service that is internal to a service provider. The L2NM is particularly meant to be used by a network controller to derive the configuration information that will be sent to relevant network devices.

Also, this document defines a YANG module to manage Ethernet segments and the initial versions of two IANA-maintained modules that defines a set of identities of BGP Layer 2 encapsulation types and pseudowire types.

Editorial Note (To be removed by RFC Editor)

Please update these statements within the document with the RFC number to be assigned to this document:

- \* "This version of this YANG module is part of RFC XXXX;"
- \* "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs";
- \* reference: RFC XXXX

Also, please update the "revision" date of the YANG modules.

Status of This Memo

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

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

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

This Internet-Draft will expire on 14 November 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Acronyms and Abbreviations . . . . .	6
4. Reference Architecture . . . . .	6
5. Relationship to Other YANG Data Models . . . . .	10
6. Description of the Ethernet Segment YANG Module . . . . .	11
7. Description of the L2NM YANG Module . . . . .	14
7.1. Overall Structure of the Module . . . . .	14
7.2. VPN Profiles . . . . .	14
7.3. VPN Services . . . . .	16
7.4. Global Parameters Profiles . . . . .	20
7.5. VPN Nodes . . . . .	23
7.5.1. BGP Auto-Discovery . . . . .	26
7.5.2. Signaling Options . . . . .	27
7.5.2.1. BGP . . . . .	29
7.5.2.2. LDP . . . . .	30
7.5.2.3. L2TP . . . . .	31
7.6. VPN Network Accesses . . . . .	32
7.6.1. Connection . . . . .	34
7.6.2. EVPN-VPWS Service Instance . . . . .	37

7.6.3. Ethernet OAM . . . . .	38
7.6.4. Services . . . . .	40
8. YANG Modules . . . . .	45
8.1. IANA-Maintained Module for BGP Layer 2 Encapsulation Types . . . . .	45
8.2. IANA-Maintained Module for Pseudowire Types . . . . .	51
8.3. Ethernet Segments . . . . .	58
8.4. L2NM . . . . .	66
9. Security Considerations . . . . .	119
10. IANA Considerations . . . . .	120
10.1. Registering YANG Modules . . . . .	121
10.2. BGP Layer 2 Encapsulation Types . . . . .	122
10.3. Pseudowire Types . . . . .	122
11. References . . . . .	123
11.1. Normative References . . . . .	123
11.2. Informative References . . . . .	126
Appendix A. Examples . . . . .	132
A.1. BGP-based VPLS . . . . .	132
A.2. BGP-based VPWS with LDP Signaling . . . . .	137
A.3. LDP-based VPLS . . . . .	141
A.4. VPWS-EVPN Service Instance . . . . .	145
A.5. Automatic ESI Assignment . . . . .	151
A.6. VPN Network Access Precedence . . . . .	154
Acknowledgements . . . . .	157
Contributors . . . . .	157
Authors' Addresses . . . . .	157

## 1. Introduction

[RFC8466] defines an L2VPN Service Model (L2SM) YANG data model that can be used between customers and service providers for ordering Layer 2 Virtual Private Network (L2VPN) services. This document complements the L2SM by creating a network-centric view of the service: the L2VPN Network Model (L2NM).

The L2NM (Section 8.4) can be exposed, for example, by a network controller to a service controller within the service provider's network. In particular, the model can be used in the communication interface between the entity that interacts directly with the customer (i.e., the service orchestrator) and the entity in charge of network orchestration and control (a.k.a., network controller/orchestrator) by allowing for more network-centric information to be included.

The L2NM supports capabilities, such as exposing operational parameters, transport protocols selection, and precedence. It can also serve as a multi-domain orchestration interface.

The L2NM is scoped for a variety of Layer 2 Virtual Private Networks, such as:

- \* Virtual Private LAN Service (VPLS) [RFC4761][RFC4762]
- \* Virtual Private Wire Service (VPWS) (Section 3.1.1 of [RFC4664])
- \* Various flavors of Ethernet VPNs (EVPNs):
  - VPWS EVPN [RFC8214],
  - Provider Backbone Bridging Ethernet VPNs (PBB EVPNs) [RFC7623],
  - EVPN over MPLS [RFC7432], and
  - EVPN over Virtual eXtensible Local Area Network (VXLAN) [RFC8365].

The L2NM is designed to easily support future Layer 2 VPN flavors and procedures (e.g., advanced configuration such as pseudowires resilience or Multi-Segment pseudowires [RFC7267]). A set of examples to illustrate the use of the L2NM are provided in Appendix A.

Also, this document defines the initial versions of two IANA-maintained modules that define a set of identities of BGP Layer 2 encapsulation types (Section 8.1) and pseudowire types (Section 8.2). Relying upon these IANA-maintained modules is meant to provide more flexibility in handling new types rather than being limited by a set of identities defined in the L2NM itself. Section 8.3 defines a YANG module to manage Ethernet Segments (ESes) that are required for instantiating EVPNs.

This document uses the common Virtual Private Network (VPN) YANG module defined in [RFC9181].

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

## 2. Terminology

This document assumes that the reader is familiar with [RFC6241], [RFC7950], [RFC8466], [RFC4026], and [RFC8309]. This document uses terminology from those documents.

This document uses the term "network model" as defined in Section 2.1 of [RFC8969].

The meanings of the symbols in YANG tree diagrams is defined in [RFC8340].

This document makes use of the following terms:

Ethernet segment (ES): Refers to the set of the Ethernet links that



are used by a customer site (device or network) to connect to one or more Provider Edges (PEs).

**Layer 2 VPN Service Model (L2SM):** Describes the service characterization of an L2VPN that interconnects a set of sites from the customer's perspective. The customer service model does not provide details on the service provider network. An L2VPN customer service model is defined in [RFC8466].

**Layer 2 VPN Network Model (L2NM):** Refers to the YANG data model that describes an L2VPN service with a network-centric view. It contains information on the service provider network and might include allocated resources. Network controllers can use it to manage the Layer 2 VPN service configuration in the service provider's network. The corresponding YANG module can be used by a service orchestrator to request a VPN service to a network controller or to expose the list of active L2VPN services.

**MAC-VRF:** Refers to a Virtual Routing and Forwarding (VRF) table for Media Access Control (MAC) addresses on a PE.

**Network controller:** Denotes a functional entity responsible for the management of the service provider network.

**Service orchestrator:** Refers to a functional entity that interacts with the customer of an L2VPN relying upon, e.g., the L2SM. The service orchestrator is responsible for the Customer Edge - to Provider Edge (CE-PE) attachment circuits, the PE selection, and requesting the activation of the L2VPN service to a network controller.

**Service provider network:** Is a network able to provide L2VPN-related services.

**VPN node:** Is an abstraction that represents a set of policies applied on a PE and belonging to a single VPN service. A VPN service involves one or more VPN nodes. The VPN node will identify the service providers' node on which the VPN is deployed.

**VPN network access:** Is an abstraction that represents the network interfaces that are associated with a given VPN node. Traffic coming from the VPN network access belongs to the VPN. The attachment circuits (bearers) between Customer Edges (CEs) and Provider Edges (PEs) are terminated in the VPN network access.

**VPN service provider:** Is a service provider that offers L2VPN-related services.

### 3. Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document:

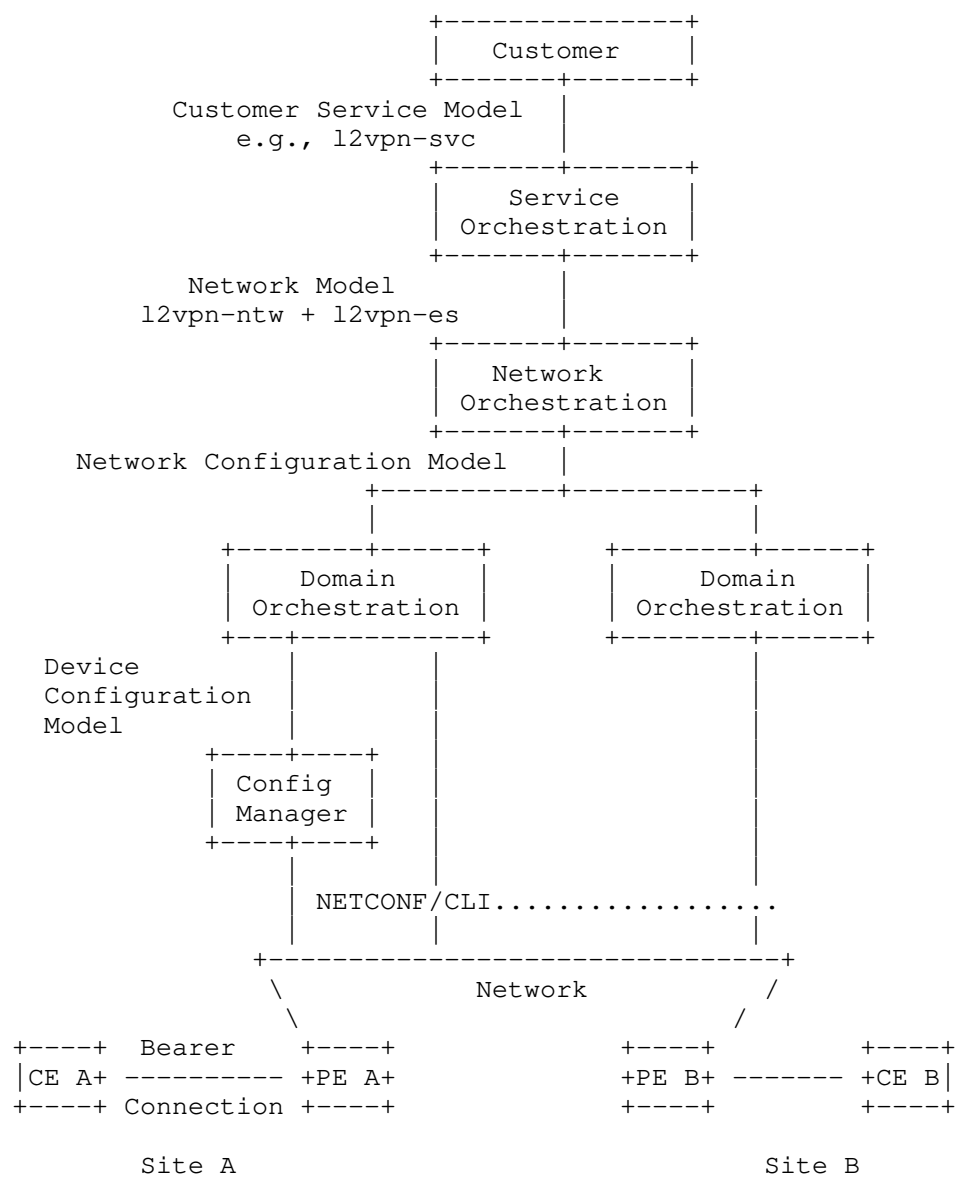
ACL	Access Control List
BGP	Border Gateway Protocol
BUM	Broadcast, unknown unicast, or multicast
CE	Customer Edge
ES	Ethernet Segment
ESI	Ethernet Segment Identifier
EVPN	Ethernet VPN
L2VPN	Layer 2 Virtual Private Network
L2SM	L2VPN Service Model
L2NM	L2VPN Network Model
MAC	Media Access Control
PBB	Provider Backbone Bridging
PCP	Priority Code Point
PE	Provider Edge
QoS	Quality of Service
RD	Route Distinguisher
RT	Route Target
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VPWS	Virtual Private Wire Service
VRF	Virtual Routing and Forwarding

### 4. Reference Architecture

Figure 1 illustrates how the L2NM is used. As a reminder, this figure is an expansion of the architecture presented in Section 3 of [RFC8466] and decomposes the box marked "orchestration" in that figure into three separate functional components called "Service Orchestration", "Network Orchestration", and "Domain Orchestration".

Similar to Section 3 of [RFC8466], CE to PE attachment is achieved through a bearer with a Layer 2 connection on top. The bearer refers to properties of the attachment that are below Layer 2, while the connection refers to Layer 2 protocol-oriented properties.

The reader may refer to [RFC8309] for the distinction between the "Customer Service Model", the "Service Delivery Model", the "Network Configuration Model", and the "Device Configuration Model". The "Domain Orchestration" and "Config Manager" roles may be performed by "SDN Controllers".



NETCONF: Network Configuration Protocol  
CLI: Command-Line Interface

Figure 1: L2SM and L2NM Interaction

The customer may use various means to request a service that may trigger the instantiation of an L2NM. The customer may use the L2SM or may rely upon more abstract models to request a service that relies upon an L2VPN service. For example, the customer may supply an IP Connectivity Provisioning Profile (CPP) that characterizes the requested service [RFC7297], an enhanced VPN (VPN+) service [I-D.ietf-teas-enhanced-vpn], or an IETF network slice service [I-D.ietf-teas-ietf-network-slices].

Note also that both the L2SM and the L2NM may be used in the context of the Abstraction and Control of TE Networks (ACTN) framework [RFC8453]. Figure 2 shows the Customer Network Controller (CNC), the Multi-Domain Service Coordinator (MDSC), and the Provisioning Network Controller (PNC).

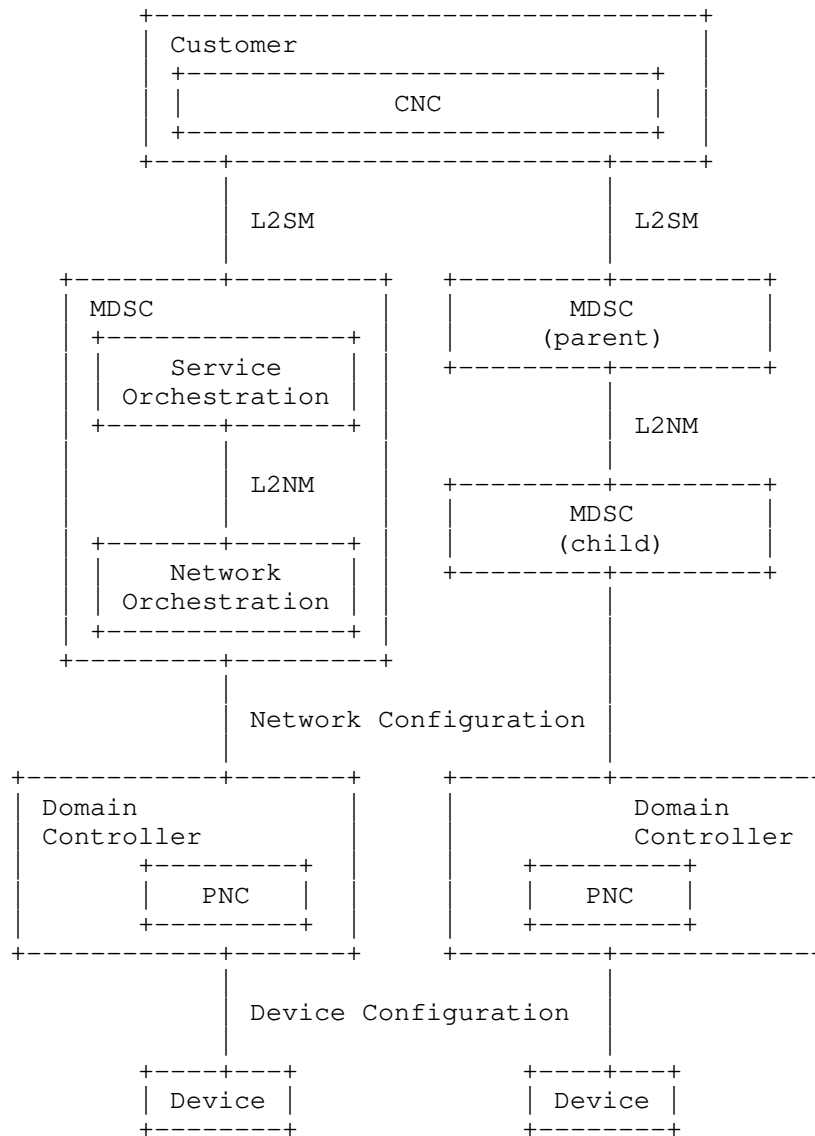


Figure 2: L2SM and L2NM in the Context of ACTN

## 5. Relationship to Other YANG Data Models

The "ietf-vpn-common" module [RFC9181] includes a set of identities, types, and groupings that are meant to be reused by VPN-related YANG modules independently of the layer (e.g., Layer 2, Layer 3) and the type of the module (e.g., network model, service model) including future revisions of existing models (e.g., [RFC8466]). The L2NM reuses these common types and groupings.

Also, the L2NM uses the IANA-maintained modules "iana-bgp-l2-encaps" (Section 8.1) and "iana-pseudowire-types" (Section 8.2) to identify a Layer 2 encapsulation type. More details are provided in Sections 7.5.2.1 and 7.5.2.3.

For the particular case of EVPN, the L2NM includes a name that refers to an Ethernet segment that is created using the "ietf-ethernet-segment" module (Section 8.3). Some ES-related examples are provided in Appendices A.4 and A.5.

As discussed in Section 4, the L2NM is used to manage L2VPN services within a service provider network. The module provides a network view of the L2VPN service. Such a view is only visible within the service provider and is not exposed outside (to customers, for example). The following discusses how the L2NM interfaces with other YANG modules:

**L2SM:** The L2NM is not a customer service model.

The internal view of the service (i.e., the L2NM) may be mapped to an external view which is visible to customers: L2VPN Service Model (L2SM) [RFC8466].

The L2NM can be fed with inputs that are requested by customers, typically, relying upon an L2SM template. Concretely, some parts of the L2SM module can be directly mapped into the L2NM while other parts are generated as a function of the requested service and local guidelines. Finally, there are parts local to the service provider and do not map directly to the L2SM.

Note that using the L2NM within a service provider does not assume, nor does it precludes exposing the VPN service via the L2SM. This is deployment specific. Nevertheless, the design of L2NM tries to align as much as possible with the features supported by the L2SM to ease the grafting of both the L2NM and the L2SM for the sake of highly automated VPN service provisioning and delivery.

**Network Topology Modules:** An L2VPN involves nodes that are part of a

topology managed by the service provider network. Such a topology can be represented using the network topology module in [RFC8345] or its extension, such as a network YANG module for Service Attachment Points (SAPs) [I-D.ietf-opsawg-sap].

Device Modules: The L2NM is not a device model.

Once a global VPN service is captured by means of the L2NM, the actual activation and provisioning of the VPN service will involve a variety of device modules to tweak the required functions for the delivery of the service. These functions are supported by the VPN nodes and can be managed using device YANG modules. A non-comprehensive list of such device YANG modules is provided below:

- \* Interfaces [RFC8343].
- \* BGP [I-D.ietf-idr-bgp-model].
- \* MPLS [RFC8960].
- \* Access Control Lists (ACLs) [RFC8519].

How the L2NM is used to derive device-specific actions is implementation specific.

## 6. Description of the Ethernet Segment YANG Module

The 'ietf-ethernet-segment' module (Figure 3) is used to manage a set of Ethernet segments in the context of an EVPN service.

```

module: ietf-ethernet-segment
  +--rw ethernet-segments
    +--rw ethernet-segment* [name]
      +--rw name string
      +--rw esi-type? identityref
      +--rw (esi-choice)?
        +--:(directly-assigned)
        |   +--rw ethernet-segment-identifier? yang:hex-string
        +--:(auto-assigned)
        |   +--rw esi-auto
        |     +--rw (auto-mode)?
        |       +--:(from-pool)
        |       |   +--rw esi-pool-name? string
        |       +--:(full-auto)
        |       |   +--rw auto? empty
        |       +--ro auto-ethernet-segment-identifier?
        |           yang:hex-string
        +--rw esi-redundancy-mode? identityref
      +--rw df-election
        +--rw df-election-method? identityref
        +--rw revertive? boolean
        +--rw election-wait-time? uint32
      +--rw split-horizon-filtering? boolean
      +--rw pbb
        +--rw backbone-src-mac? yang:mac-address
      +--rw member* [ne-id interface-id]
        +--rw ne-id string
        +--rw interface-id string

```

Figure 3: Ethernet Segments Tree Structure

The descriptions of the data nodes depicted in Figure 3 are as follows:

'name': Sets a name to uniquely identify an ES within a service provider network. This name is referenced in the VPN network access level of the L2NM (Section 7.6).

'esi-type': Indicates the Ethernet Segment Identifier (ESI) type as discussed in Section 5 of [RFC7432]. ESIs can be automatically assigned either with or without indicating a pool from which an ESI should be taken ('esi-pool-name'). The following types are supported:

'esi-type-0-operator': The ESI is directly configured by the VPN service provider. The configured value is provided in 'ethernet-segment-identifier'.



'esi-type-1-lacp': The ESI is auto-generated from the IEEE 802.1AX Link Aggregation Control Protocol (LACP) [IEEE802.1AX].

'esi-type-2-bridge': The ESI is auto-generated and determined based on the Layer 2 bridge protocol.

'esi-type-3-mac': The ESI is a MAC-based ESI value that can be auto-generated or configured by the VPN service provider.

'esi-type-4-router-id': The ESI is auto-generated or configured by the VPN service provider based on the Router ID. The 'router-id' supplied in Section 7.5 can be used to auto-derive an ESI when this type is used.

'esi-type-5-asn': The ESI is auto-generated or configured by the VPN service provider based on the Autonomous System (AS) number. The 'local-autonomous-system' supplied in Section 7.4 can be used to auto-derive an ESI when this type is used.

Auto-generated values can be retrieved using 'auto-ethernet-segment-identifier'.

'esi-redundancy-mode': Specifies the EVPN redundancy mode for a given ES. The following modes are supported: Single-Active (Section 14.1.1 of [RFC7432]) or All-Active (Section 14.1.2 of [RFC7432]).

'df-election': Specifies a set of parameters related to the Designated Forwarder (DF) election (Section 8.5 of [RFC7432]). For example, this data node can be used to indicate an election method (e.g., [RFC8584] or [I-D.ietf-bess-evpn-pref-df]). If no election method is indicated, the default one defined in Section 8.5 of [RFC7432] is used.

'split-horizon-filtering': Controls the activation of the split-horizon filtering for an ES (Section 8.3 of [RFC7432]).

'pbb': Indicates data nodes that are specific to PBB [IEEE-802-1ah]:

'backbone-src-mac': Associates a Provider Backbone MAC (B-MAC) address with an ES. This is particularly useful for All-Active multihomed ESes (Section 9.1 of [RFC7623]).

'member': Lists the members of an ES in a service provider network.

## 7. Description of the L2NM YANG Module

The L2NM ('ietf-l2vpn-ntw', Section 8.4) is used to manage L2VPNs within a service provider network. In particular, the 'ietf-l2vpn-ntw' module can be used to create, modify, delete and retrieve L2VPN services in a network controller. The module is designed to minimize the amount of customer-related information.

The full tree diagram of the module can be generated using the "pyang" tool [PYANG]. That tree is not included here because it is too long (Section 3.3 of [RFC8340]). Instead, subtrees are provided for the reader's convenience.

### 7.1. Overall Structure of the Module

The 'ietf-l2vpn-ntw' module uses two main containers: 'vpn-profiles' and 'vpn-services' (see Figure 4).

The 'vpn-profiles' container is used by the provider to maintain a set of common VPN profiles that apply to VPN services (Section 7.2).

The 'vpn-services' container maintains the set of L2VPN services managed in the service provider network. The module allows creating a new L2VPN service by adding a new instance of 'vpn-service'. The 'vpn-service' is the data structure that abstracts the VPN service (Section 7.3).

```

module: ietf-l2vpn-ntw
  +--rw l2vpn-ntw
    +--rw vpn-profiles
    |   ...
    +--rw vpn-services
      +--rw vpn-service* [vpn-id]
      |   ...
      +--rw vpn-nodes
        +--rw vpn-node* [vpn-node-id]
        |   ...
        +--rw vpn-network-accesses
          +--rw vpn-network-access* [id]
          |   ...
          ...

```

Figure 4: Overall L2NM Tree Structure

### 7.2. VPN Profiles

The 'vpn-profiles' container (Figure 5) is used by a VPN service provider to define and maintain a set of VPN profiles [RFC9181] that apply to one or several VPN services.

```

+--rw l2vpn-ntw
  +--rw vpn-profiles
    +--rw valid-provider-identifiers
      +--rw external-connectivity-identifier* [id]
        {external-connectivity}?
        +--rw id      string
      +--rw encryption-profile-identifier* [id]
        +--rw id      string
      +--rw qos-profile-identifier* [id]
        +--rw id      string
      +--rw bfd-profile-identifier* [id]
        +--rw id      string
      +--rw forwarding-profile-identifier* [id]
        +--rw id      string
      +--rw routing-profile-identifier* [id]
        +--rw id      string
    +--rw vpn-services
      ...

```

Figure 5: VPN Profiles Subtree Structure

This document does not make any assumption about the exact definition of these profiles. The exact definition of the profiles is local to each VPN service provider. The model only includes an identifier for these profiles in order to ease identifying and binding local policies when building a VPN service. As shown in Figure 5, the following identifiers can be included:

'external-connectivity-identifier': This identifier refers to a profile that defines the external connectivity provided to a VPN service (or a subset of VPN sites). External connectivity may be access to the Internet or restricted connectivity, such as access to a public/private cloud.

'encryption-profile-identifier': An encryption profile refers to a set of policies related to the encryption schemes and setup that can be applied when building and offering a VPN service.

'qos-profile-identifier': A Quality of Service (QoS) profile refers to as set of policies, such as classification, marking, and actions (e.g., [RFC3644]).

'bfd-profile-identifier': A Bidirectional Forwarding Detection (BFD) profile refers to a set of BFD policies [RFC5880] that can be invoked when building a VPN service.

'forwarding-profile-identifier': A forwarding profile refers to the

policies that apply to the forwarding of packets conveyed within a VPN. Such policies may consist, for example, of applying ACLs.

'routing-profile-identifier': A routing profile refers to a set of routing policies that will be invoked (e.g., BGP policies) when delivering the VPN service.

### 7.3. VPN Services

The 'vpn-service' is the data structure that abstracts an L2VPN service in the service provider network. Each 'vpn-service' is uniquely identified by an identifier: 'vpn-id'. Such a 'vpn-id' is only meaningful locally within the network controller. The subtree of the 'vpn-services' is shown in Figure 6.

```

+--rw vpn-services
  +--rw vpn-service* [vpn-id]
    +--rw vpn-id                vpn-common:vpn-id
    +--rw vpn-name?             string
    +--rw vpn-description?      string
    +--rw customer-name?        string
    +--rw parent-service-id?    vpn-common:vpn-id
    +--rw vpn-type?             identityref
    +--rw vpn-service-topology? identityref
    +--rw bgp-ad-enabled?       boolean
    +--rw signaling-type?       identityref
    +--rw global-parameters-profiles
    |   ...
  +--rw underlay-transport
    +--rw (type)?
    |   +--:(abstract)
    |   |   +--rw transport-instance-id? string
    |   |   +--rw instance-type?        identityref
    |   +--:(protocol)
    |   |   +--rw protocol*              identityref
  +--rw status
    +--rw admin-status
    |   +--rw status?                identityref
    |   +--rw last-change?          yang:date-and-time
    +--ro oper-status
    |   +--ro status?                identityref
    |   +--ro last-change?          yang:date-and-time
  +--rw vpn-nodes
    ...

```

Figure 6: VPN Services Subtree

The descriptions of the VPN service data nodes that are depicted in Figure 6 are as follows:

'vpn-id': An identifier that is used to uniquely identify the L2VPN service within the L2NM scope.

'vpn-name': Associates a name with the service in order to facilitate the identification of the service.

'vpn-description': Includes a textual description of the service.

The internal structure of a VPN description is local to each VPN service provider.

'customer-name': Indicates the name of the customer who ordered the service.

'parent-service-id': Refers to an identifier of the parent service (e.g., the L2SM, IETF network slice, VPN+) that triggered the creation of the L2VPN service. This identifier is used to easily correlate the (network) service as built in the network with a service order. A controller can use that correlation to enrich or populate some fields (e.g., description fields) as a function of local deployments.

'vpn-type': Indicates the L2VPN type. The following types, defined in [RFC9181], can be used for the L2NM:

'vpls': Virtual Private LAN Service (VPLS) as defined in [RFC4761] or [RFC4762]. This type is also used for hierarchical VPLS (H-VPLS) (Section 10 of [RFC4762]).

'vpws': Virtual Private Wire Service (VPWS) as defined in Section 3.1.1 of [RFC4664].

'vpws-evpn': VPWS as defined in [RFC8214].

'pbb-evpn': Provider Backbone Bridging (PBB) EVPNs as defined in [RFC7623].

'mpls-evpn': MPLS-based EVPNs [RFC7432].

'vxlan-evpn': VXLAN based EVPNs [RFC8365].

The type is used as a condition for the presence of some data nodes in the L2NM.

'vpn-service-topology': Indicates the network topology for the

service: hub-spoke, any-to-any, or custom. These types are defined in [RFC9181].

'bgp-ad-enabled': Controls whether BGP auto-discovery is enabled. If so, additional data nodes are included (Section 7.5.1).

'signaling-type': Indicates the signaling that is used for setting up pseudowires. Signaling type values are taken from [RFC9181]. The following signaling options are supported:

'bgp-signaling': The L2NM supports two flavors of BGP-signaled L2VPNs:

'l2vpn-bgp': The service is a Multipoint VPLS that uses a BGP control plane as described in [RFC4761] and [RFC6624].

'evpn-bgp': The service is a Multipoint VPLS that uses also a BGP control plane, but also includes the additional EVPN features and related parameters [RFC7432] and [RFC7209].

'ldp-signaling': A Multipoint VPLS that uses a mesh of LDP-signaled Pseudowires [RFC6074].

'l2tp-signaling': The L2NM uses L2TP-signaled Pseudowires as described in [RFC6074].

Table 1 summarizes the allowed signaling types for each VPN service type ('vpn-type'). See Section 7.5.2 for more details.

VPN Type	Signaling Options
vppls	l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)
vpws	l2tp-signaling, ldp-signaling, bgp-signaling (l2vpn-bgp)
vpws-evpn	bgp-signaling (evpn-bgp)
pbb-evpn	bgp-signaling (evpn-bgp)
mpls-evpn	bgp-signaling (evpn-bgp)
vxlan-evpn	bgp-signaling (evpn-bgp)

Table 1: Signaling Options per VPN  
Service Type

'global-parameters-profiles': Defines reusable parameters for the same L2VPN service.

More details are provided in Section 7.4.

'underlay-transport': Describes the preference for the transport technology to carry the traffic of the VPN service. This preference is especially useful in networks with multiple domains and Network-to-Network Interface (NNI) types. The underlay transport can be expressed as an abstract transport instance (e.g., an identifier of a VPN+ instance, a virtual network identifier, or a network slice name) or as an ordered list of the actual protocols to be enabled in the network.

A rich set of protocol identifiers that can be used to refer to an underlay transport (or how such an underlay is set up) are defined in [RFC9181].

The model defined in Section 6.3.2 of [I-D.ietf-teas-te-service-mapping-yang] may be used if specific protection and availability requirements are needed between PEs.

'status': Used to track the overall status of a given VPN service. Both operational and administrative status are maintained together with a timestamp. For example, a service can be created, but not put into effect.

Administrative and operational status can be used as a trigger to detect service anomalies. For example, a service that is declared at the service layer as being active but still inactive at the network layer is an indication that network provisioning actions are needed to align the observed service status with the expected service status.

'vpn-node': An abstraction that represents a set of policies applied to a network node and belonging to a single 'vpn-service'. An L2VPN service is typically built by adding instances of 'vpn-node' to the 'vpn-nodes' container.

A 'vpn-node' contains 'vpn-network-accesses', which are the interfaces attached to the VPN by which the customer traffic is received. Therefore, the customer sites are connected to the 'vpn-network-accesses'.

Note that, as this is a network data model, the information about customers sites is not required in the model. Such information is rather relevant in the L2SM. Whether that information is included in the L2NM, e.g., to populate the various 'description' data nodes is implementation specific.

More details are provided in Section 7.5.

#### 7.4. Global Parameters Profiles

The 'global-parameters-profile' defines reusable parameters for the same L2VPN service instance ('vpn-service'). Global parameters profiles are defined at the VPN service level, activated at the VPN node level, and then an activated VPN profile may be used at the VPN network access level. Each VPN instance profile is identified by 'profile-id'. Some of the data nodes can be adjusted at the VPN node or VPN network access levels. These adjusted values take precedence over the global values. The subtree of 'global-parameters-profile' is depicted in Figure 7.

```

...
+--rw vpn-services
  +--rw vpn-service* [vpn-id]
    ...
    +--rw global-parameters-profiles
      +--rw global-parameters-profile* [profile-id]
        +--rw profile-id string
        +--rw (rd-choice)?
          +--:(directly-assigned)
          | +--rw rd?

```



```

|         rt-types:route-distinguisher
+---:(directly-assigned-suffix)
|   +---rw rd-suffix?          uint16
+---:(auto-assigned)
|   +---rw rd-auto
|       +---rw (auto-mode)?
|           +---:(from-pool)
|               |   +---rw rd-pool-name?    string
|           +---:(full-auto)
|               |   +---rw auto?            empty
|           +---ro auto-assigned-rd?
|               rt-types:route-distinguisher
+---:(auto-assigned-suffix)
|   +---rw rd-auto-suffix
|       +---rw (auto-mode)?
|           +---:(from-pool)
|               |   +---rw rd-pool-name?    string
|           +---:(full-auto)
|               |   +---rw auto?            empty
|           +---ro auto-assigned-rd-suffix? uint16
+---:(no-rd)
|   +---rw no-rd?              empty
+---rw vpn-target* [id]
|   +---rw id                  uint8
|   +---rw route-targets* [route-target]
|       |   +---rw route-target    rt-types:route-target
+---rw route-target-type
|       rt-types:route-target-type
+---rw vpn-policies
|   +---rw import-policy?     string
|   +---rw export-policy?     string
+---rw local-autonomous-system? inet:as-number
+---rw svc-mtu?               uint32
+---rw ce-vlan-preservation?   boolean
+---rw ce-vlan-cos-preservation? boolean
+---rw control-word-negotiation? boolean
+---rw mac-policies
|   +---rw mac-addr-limit
|       |   +---rw limit-number?    uint16
|       |   +---rw time-interval?   uint32
|       |   +---rw action?          identityref
|   +---rw mac-loop-prevention
|       |   +---rw window?          uint32
|       |   +---rw frequency?       uint32
|       |   +---rw retry-timer?     uint32
|       |   +---rw protection-type? identityref
+---rw multicast {vpn-common:multicast}?
|   +---rw enabled?            boolean

```

```

|      +---rw customer-tree-flavors
|      +---rw tree-flavor*   identityref
|      ...

```

Figure 7: Global Parameters Profiles Subtree

The description of the global parameters profile is as follows:

'profile-id': Uniquely identifies a global parameter profile in the context of an L2VPN service.

'rd': As defined in [RFC9181], these RD assignment modes are supported: direct assignment, automatic assignment from a given pool, full automatic assignment, and no assignment.

Also, the module accommodates deployments where only the Assigned Number subfield of RDs is assigned from a pool while the Administrator subfield is set to, e.g., the Router ID that is assigned to a VPN node. The module supports these modes for managing the Assigned Number subfield: explicit assignment, auto-assignment from a pool, and full auto-assignment.

'vpn-targets': Specifies RT import/export rules for the VPN service.

'local-autonomous-system': Indicates the Autonomous System Number (ASN) that is configured for the VPN node. The ASN can be used to auto-derive some other attributes such as RDs or Ethernet Segment Identifiers (ESIs).

'svc-mtu': Is the service MTU for an L2VPN service (i.e., Layer 2 MTU including L2 frame header/tail). It is also known as the maximum transmission unit or maximum frame size.

'ce-vlan-preservation': Is set to preserve the Customer Edge VLAN IDs (CE-VLAN IDs) from ingress to egress, i.e., CE-VLAN tag of the egress frame are identical to those of the ingress frame that yielded this egress service frame. If all-to-one bundling within a site is enabled, then preservation applies to all ingress service frames. If all-to-one bundling is disabled, then preservation applies to tagged Ingress service frames having CE-VLAN ID 1 through 4094.

'ce-vlan-cos-preservation': Controls the CE VLAN CoS preservation. When set, Priority Code Point (PCP) bits in the CE-VLAN tag of the egress frame are identical to those of the ingress frame that yielded this egress service frame.

'control-word-negotiation': Controls whether control-word

negotiation is enabled (if set to true) or not (if set to false). Refer to Section 7 of [RFC8077] for more details.

'mac-policies': Includes a set of MAC policies that apply to the service:

'mac-addr-limit': Is a container of MAC address limit configuration. It includes the following data nodes:

'limit-number': Maximum number of MAC addresses learned from the customer for a single service instance.

'time-interval': The aging time of the mac address.

'action': Specifies the action when the upper limit is exceeded: drop the packet, flood the packet, or simply send a warning message.

'mac-loop-prevention': Container for MAC loop prevention.

'window': The time interval over which a MAC mobility event is detected and checked.

'frequency': The number of times to detect MAC duplication, where a 'duplicate MAC address' situation has occurred within the 'window' time interval and the duplicate MAC address has been added to a list of duplicate MAC addresses.

'retry-timer': The retry timer. When the retry timer expires, the duplicate MAC address will be flushed from the MAC-VRF.

'protection-type': It defines the loop prevention type (e.g., shut).

'multicast': Controls whether multicast is allowed in the service.

## 7.5. VPN Nodes

The 'vpn-node' (Figure 8) is an abstraction that represents a set of policies/configurations applied to a network node and that belong to a single 'vpn-service'. A 'vpn-node' contains 'vpn-network-accesses', which are the interfaces involved in the creation of the VPN. The customer sites are connected to the 'vpn-network-accesses'.

```

+--rw l2vpn-ntw
+--rw vpn-profiles
|   ...
+--rw vpn-services
+--rw vpn-service* [vpn-id]
    ...
+--rw vpn-nodes
+--rw vpn-node* [vpn-node-id]
+--rw vpn-node-id          vpn-common:vpn-id
+--rw description?         string
+--rw ne-id?               string
+--rw role?                identityref
+--rw router-id?           rt-types:router-id
+--rw active-global-parameters-profiles
|   +--rw global-parameters-profile* [profile-id]
|   |   +--rw profile-id          leafref
|   |   +--rw local-autonomous-system?
|   |   |   inet:as-number
|   |   +--rw svc-mtu?            uint32
|   |   +--rw ce-vlan-preservation? boolean
|   |   +--rw ce-vlan-cos-preservation? boolean
|   |   +--rw control-word-negotiation? boolean
|   |   +--rw mac-policies
|   |   |   +--rw mac-addr-limit
|   |   |   |   +--rw limit-number?    uint16
|   |   |   |   +--rw time-interval?    uint32
|   |   |   |   +--rw action?          identityref
|   |   |   +--rw mac-loop-prevention
|   |   |   |   +--rw window?          uint32
|   |   |   |   +--rw frequency?       uint32
|   |   |   |   +--rw retry-timer?     uint32
|   |   |   |   +--rw protection-type? identityref
|   |   +--rw multicast {vpn-common:multicast}?
|   |   |   +--rw enabled?            boolean
|   |   +--rw customer-tree-flavors
|   |   |   +--rw tree-flavor*        identityref
+--rw status
|   ...
+--rw bgp-auto-discovery
|   ...
+--rw signaling-option
|   ...
+--rw vpn-network-accesses
    ...

```

Figure 8: VPN Nodes Subtree

The descriptions of VPN node data nodes are as follows:

- 'vpn-node-id': Used to uniquely identify a node that enables a VPN network access.
- 'description': Provides a textual description of the VPN node.
- 'ne-id': Includes an identifier of the network element where the VPN node is deployed.
- 'role': Indicates the role of the VPN instance profile in the VPN. Role values are defined in [RFC9181] (e.g., 'any-to-any-role', 'spoke-role', 'hub-role').
- 'router-id': Indicates a 32-bit number that is used to uniquely identify a router within an Autonomous System (AS).
- 'active-global-parameters-profiles': Lists the set of active global VPN parameters profiles for this VPN node. Concretely, one or more global profiles that are defined at the VPN service level (i.e., under 'l2vpn-ntw/vpn-services/vpn-service' level) can be activated at the VPN node level; each of these profiles is uniquely identified by means of 'profile-id'. The structure of 'active-global-parameters-profiles' uses the same data nodes as Section 7.4 except RD and RT related data nodes.
- Values defined in 'active-global-parameters-profiles' overrides the values defined in the VPN service level.
- 'status': Tracks the status of a node involved in a VPN service. Both operational and administrative status are maintained. A mismatch between the administrative status vs. the operational status can be used as a trigger to detect anomalies.
- 'bgp-auto-discovery': See Section 7.5.1.
- 'signaling-option': See Section 7.5.2.
- 'vpn-network-accesses': Represents the point to which sites are connected.

Note that, unlike the L2SM, the L2NM does not need to model the customer site -- only the points that receive traffic from the site are covered (i.e., the PE side of Provider Edge to Customer Edge (PE-CE) connections). Hence, the VPN network access contains the connectivity information between the provider's network and the customer premises. The VPN profiles ('vpn-profiles') have a set of routing policies that can be applied during the service creation.

See Section 7.6 for more details.

#### 7.5.1. BGP Auto-Discovery

The 'bgp-auto-discovery' container (Figure 9) includes the required information for the activation of BGP auto-discovery [RFC4761][RFC6624].

```

+--rw l2vpn-ntw
  +--rw vpn-profiles
  |   ...
  +--rw vpn-services
    +--rw vpn-service* [vpn-id]
    ...
    +--rw vpn-nodes
      +--rw vpn-node* [vpn-node-id]
      ...
      +--rw bgp-auto-discovery
        +--rw (bgp-type)?
        |   +--:(l2vpn-bgp)
        |   |   +--rw vpn-id?
        |   |       vpn-common:vpn-id
        |   +--:(evpn-bgp)
        |   |   +--rw evpn-type?                leafref
        |   |   +--rw auto-rt-enable?            boolean
        |   |   +--ro auto-route-target?
        |   |       rt-types:route-target
        |   +--rw (rd-choice)?
        |   |   +--:(directly-assigned)
        |   |   |   +--rw rd?
        |   |   |       rt-types:route-distinguisher
        |   |   +--:(directly-assigned-suffix)
        |   |   |   +--rw rd-suffix?                uint16
        |   |   +--:(auto-assigned)
        |   |   |   +--rw rd-auto
        |   |   |   |   +--rw (auto-mode)?
        |   |   |   |   |   +--:(from-pool)
        |   |   |   |   |   |   +--rw rd-pool-name?    string
        |   |   |   |   |   +--:(full-auto)
        |   |   |   |   |   |   +--rw auto?            empty
        |   |   |   |   +--ro auto-assigned-rd?
        |   |   |   |       rt-types:route-distinguisher
        |   |   +--:(auto-assigned-suffix)
        |   |   |   +--rw rd-auto-suffix
        |   |   |   |   +--rw (auto-mode)?
        |   |   |   |   |   +--:(from-pool)
        |   |   |   |   |   |   +--rw rd-pool-name?    string
        |   |   |   |   |   +--:(full-auto)

```

```
| | | +--rw auto? empty  
| | | +---ro auto-assigned-rd-suffix? uint16  
+---:(no-rd)  
| | | +--rw no-rd? empty  
+--rw vpn-target* [id]  
| | | +--rw id uint8  
| | | +--rw route-targets* [route-target]  
| | | | +--rw route-target rt-types:route-target  
| | | +--rw route-target-type  
| | | | rt-types:route-target-type  
+--rw vpn-policies  
| | | +--rw import-policy? string  
| | | +--rw export-policy? string  
+--rw signaling-option  
| ...  
+--rw vpn-network-accesses  
| ...
```

Figure 9: BGP Auto-Discovery Subtree

As discussed in Section 1 of [RFC6624], all of BGP-based methods include the notion of a VPN identifier that serves to unify components of a given VPN and the concept of auto-discovery; hence the support of the data node 'vpn-id'.

For the particular case of EVPN, the L2NM supports RT auto-derivation based on the Ethernet Tag ID specified in Section 7.10.1 of [RFC7432]. A VPN service provider can enable/disable this functionality by means of 'auto-rt-enable'. The assigned RT can be retrieved using 'auto-route-target'.

For all BGP-based L2VPN flavors, other data nodes such as RD and RT are used. These data nodes have the same structure as the one discussed in Section 7.4.

### 7.5.2. Signaling Options

The 'signaling-option' container (Figure 10) defines a set of data nodes for a given signaling protocol that is used for an L2VPN service. As discussed in Section 7.3, several signaling options to exchange membership information between PEs of an L2VPN are supported. The signaling type to be used for an L2VPN service is controlled at the VPN service level by means of 'signaling-type'.

```

...
+--rw vpn-nodes
  +--rw vpn-node* [vpn-node-id]
    ...
  +--rw signaling-option
    +--rw advertise-mtu?          boolean
    +--rw mtu-allow-mismatch?    boolean
    +--rw signaling-type?        leafref
    +--rw (signaling-option)?
      +--:(bgp)
      |   ...
      +--:(ldp-or-l2tp)
        +--rw (ldp-or-l2tp)?
          +--:(ldp)
          |   ...
          +--:(l2tp)
          |   ...

```

Figure 10: Signaling Option Overall Subtree

The following signaling data nodes are supported:

- 'advertise-mtu': Controls whether MTU is advertised when setting a pseudowire (e.g., Section 4.3 of [RFC4667], Section 5.1 of [RFC6624], or Section 6.1 of [RFC4762]).
- 'mtu-allow-mismatch': When set to true, it allows MTU mismatch for a pseudowire (see, e.g., Section 4.3 of [RFC4667]).
- 'signaling-type': Indicates the signaling type. This type inherits the value of 'signaling-type' defined at the service level (Section 7.3).
- 'bgp': Is provided when BGP is used for L2VPN signaling. Refer to Section 7.5.2.1 for more details.
- 'ldp': The model supports the configuration of the parameters that are discussed in Section 6 of [RFC4762]. Refer to Section 7.5.2.2 for more details.
- 'l2tp': The model supports the configuration of the parameters that are discussed in Section 4 of [RFC4667]. Refer to Section 7.5.2.3 for more details.



## 7.5.2.1. BGP

The structure of the BGP-related data nodes is provided in Figure 11.

```

...
+--rw (signaling-option)?
  ...
  +--:(bgp)
    +--rw (bgp-type)?
      +--:(l2vpn-bgp)
        +--rw ce-range?          uint16
        +--rw pw-encapsulation-type?
          |
          | identityref
        +--rw vpls-instance
          |
          | +--rw vpls-edge-id?          uint16
          | +--rw vpls-edge-id-range?    uint16
      +--:(evpn-bgp)
        +--rw evpn-type?          leafref
        +--rw service-interface-type?
          |
          | identityref
        +--rw evpn-policies
          +--rw mac-learning-mode?
            |
            | identityref
          +--rw ingress-replication?
            |
            | boolean
          +--rw p2mp-replication?
            |
            | boolean
          +--rw arp-proxy {vpn-common:ipv4}?
            +--rw enable?          boolean
            +--rw arp-suppression?
              |
              | boolean
            +--rw ip-mobility-threshold?
              |
              | uint16
            +--rw duplicate-ip-detection-interval?
              |
              | uint16
          +--rw nd-proxy {vpn-common:ipv6}?
            +--rw enable?          boolean
            +--rw nd-suppression?
              |
              | boolean
            +--rw ip-mobility-threshold?
              |
              | uint16
            +--rw duplicate-ip-detection-interval?
              |
              | uint16
          +--rw underlay-multicast?
            |
            | boolean
          +--rw flood-unknown-unicast-supression?
            |
            | boolean
          +--rw vpws-vlan-aware?    boolean

```

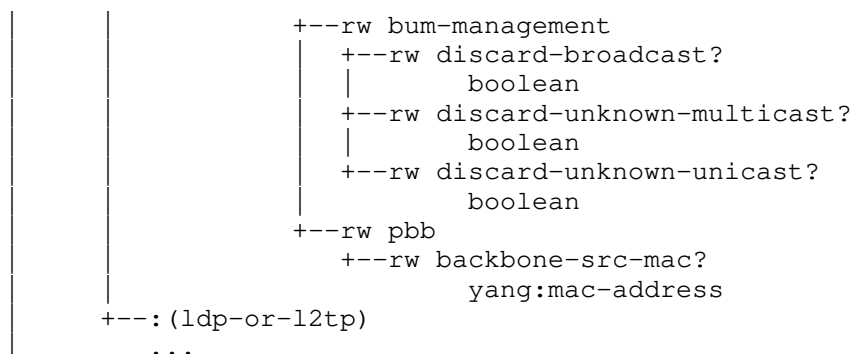


Figure 11: Signaling Option Subtree (BGP)

Remote CEs that are entitled to connect to the same VPN should fit with the CE range ('ce-range') as discussed in Section 2.2.3 of [RFC6624]. 'pw-encapsulation-type' is used to control the pseudowire encapsulation type (Section 3 of [RFC6624]). The value of the 'pw-encapsulation-type' are taken from the IANA-maintained "iana-bgp-l2-encaps" module (Section 8.1).

For the specific case of VPLS, the VPLS Edge ID (VE ID, 'vpls-edge-id') and a VE ID range ('vpls-edge-id-range') are provided as per Section 3.2 of [RFC4761]. If different VE IDs are required (e.g., multihoming as per Section 3.5 of [RFC4761]), these IDs are configured at the VPN network access level (under 'signaling-option' in Section 7.6).

For EVPN-related L2VPNs, 'service-interface-type' indicates whether this is a VLAN-based, VLAN bundle, or VLAN-aware bundle service interface (Section 6 of [RFC7432]). Moreover, a set of policies can be provided such as MAC address learning mode (Section 9 of [RFC7432]), ingress replication (Section 12.1 of [RFC7432]), Address Resolution Protocol (ARP) and Neighbor Discovery (ND) proxy (Section 10 of [RFC7432]), processing of Broadcast, unknown unicast, or multicast (BUM) (Section 12 of [RFC7432]), etc.

#### 7.5.2.2. LDP

The model supports the configuration of the parameters that are discussed in Section 6 of [RFC4762]. Such parameters include an Attachment Group Identifier (AGI) (a.k.a., VPLS-id), a Source Attachment Individual Identifier (SAII), a list of peers that are associated with a Target Attachment Individual Identifier (TAII), a pseudowire type, and a pseudowire description (Figure 12). Unlike BGP, only Ethernet and Ethernet tagged mode are supported. The AGI, SAII, and TAI are encoded following the types defined in Section 3.4

of [RFC4446].

```

...
+--rw (signaling-option)?
...
+--:(bgp)
|
...
+--:(ldp-or-l2tp)
+--rw ldp-or-l2tp
+--rw agi?
|
rt-types:route-distinguisher
+--rw saii?                               uint32
+--rw remote-targets* [taii]
|   +--rw tائي                               uint32
|   +--rw peer-addr          inet:ip-address
+--rw (ldp-or-l2tp)?
+--:(ldp)
|
+--rw t-ldp-pw-type?
|   identityref
+--rw pw-type?          identityref
+--rw pw-description?    string
+--rw mac-addr-withdraw? boolean
+--rw pw-peer-list*
|   [peer-addr vc-id]
|   +--rw peer-addr
|   |   inet:ip-address
|   +--rw vc-id    string
|   +--rw pw-priority?  uint32
+--rw qinq
|   +--rw s-tag    dot1q-types:vlanid
|   +--rw c-tag    dot1q-types:vlanid
+--:(l2tp)
...
...

```

Figure 12: Signaling Option Subtree (LDP)

#### 7.5.2.3. L2TP

The model supports the configuration of the parameters that are discussed in Section 4 of [RFC4667]. Such parameters include a Router ID that is used to uniquely identify a PE, a pseudowire type, an AGI, an SAI, and a list of peers that are associated with a TAI (Figure 13). The pseudowire type ('pseudowire-type') value is taken from the IANA-maintained "iana-pseudowire-types" module (Section 8.2).

```

...
+--rw (signaling-option)?
  ...
  +--:(bgp)
  |   ...
  +--:(ldp-or-l2tp)
  |   +--rw ldp-or-l2tp
  |   |   +--rw agi?
  |   |   |   rt-types:route-distinguisher
  |   |   +--rw saii?                               uint32
  |   |   +--rw remote-targets* [taii]
  |   |   |   +--rw taii                               uint32
  |   |   |   +--rw peer-addr       inet:ip-address
  |   +--rw (ldp-or-l2tp)?
  |   |   +--:(ldp)
  |   |   |   ...
  |   |   +--:(l2tp)
  |   |   |   +--rw router-id?
  |   |   |   |   rt-types:router-id
  |   |   |   +--rw pseudowire-type?
  |   |   |   |   identityref
...

```

Figure 13: Signaling Option Subtree (L2TP)

#### 7.6. VPN Network Accesses

A 'vpn-network-access' (Figure 14) represents an entry point to a VPN service. In other words, this container encloses the parameters that describe the access information for the traffic that belongs to a particular L2VPN.

A 'vpn-network-access' includes information such as the connection on which the access is defined, the specific Layer 2 service requirements, etc.

```

...
+--rw vpn-nodes
  +--rw vpn-node* [vpn-node-id]
    ...
    +--rw vpn-network-accesses
      +--rw vpn-network-access* [id]
        +--rw id                               vpn-common:vpn-id
        +--rw description?                     string
        +--rw interface-id?                    string
        +--rw active-vpn-node-profile?        leafref
        +--rw status
        |   ...
        +--rw connection
        |   ...
        +--rw (signaling-option)?
          +--:(bgp)
            +--rw (bgp-type)?
              +--:(l2vpn-bgp)
                +--rw ce-id?                    uint16
                +--rw remote-ce-id?            uint16
                +--rw vpls-instance
                |   +--rw vpls-edge-id?        uint16
                +--:(evpn-bgp)
                  +--rw df-preference?        uint16
                  +--rw vpws-service-instance
                  |   ...
            +--rw group* [group-id]
              +--rw group-id                    string
              +--rw precedence?
              |   identityref
              +--rw ethernet-segment-identifier? string
            +--rw ethernet-service-oam
            |   ...
            +--rw service
            |   ...

```

Figure 14: VPN Network Access Subtree

The VPN network access comprises:

'id': Includes an identifier of the VPN network access.

'description': Includes a textual description of the VPN network access.

'interface-id': Indicates the interface on which the VPN network access is bound.

'active-vpn-node-profile': Provides a pointer to an active 'global-parameters-profile' at the VPN node level. Referencing an active 'global-parameters-profile' implies that all associated data nodes will be inherited by the VPN network access. However, some of the inherited data nodes (e.g., ACL policies) can be overridden at the VPN network access level. In such case, adjusted values take precedence over inherited values.

'status': Indicates the administrative and operational status of the VPN network access.

'connection': Represents and groups the set of Layer 2 connectivity from where the traffic of the L2VPN in a particular VPN Network access is coming. See Section 7.6.1.

'signaling-option': Indicates a set of signaling options that are specific to a given VPN network access, e.g., a CE ID ('ce-id' identifying the CE within the VPN) and a remote CE ID as discussed in Section 2.2.2 of [RFC6624].

It can also include a set of data nodes that are required for the configuration of a VPWS-EVPN [RFC8214]. See Section 7.6.2.

'group': Is used for grouping VPN network accesses by assigning the same identifier to these accesses. The precedence attribute is used to differentiate the primary and secondary accesses for a service with multiple accesses. An example to illustrate the use of this container for redundancy purposes is provided in Appendix A.6. This container is also used to identify the link of an ES by allocating the same ESI. An example to illustrate this functionality is provided in Appendices A.4 and A.5.

'ethernet-service-oam': Carries information about the service OAM. See Section 7.6.3.

'service': Specifies the service parameters (e.g., QoS, multicast) to apply for a given VPN network access. See Section 7.6.4.

#### 7.6.1. Connection

The 'connection' container (Figure 15) is used to configure the relevant properties of the interface to which the L2VPN instance is attached to (e.g., encapsulation type, Link Aggregation Group (LAG) interfaces, split-horizon). The L2NM supports tag manipulation operations (e.g., tag rewrite).

Note that the 'connection' container does not include the physical-specific configuration as this is assumed to be directly handled using device modules (e.g., interfaces module). Moreover, this design is also meant to avoid manipulated global parameters at the service level and lower the risk of impacting other services sharing the same physical interface.

A reference to the bearer is maintained to allow keeping the link between the L2SM and the L2NM when both data models are used in a given deployment.

Some consistency checks should be ensured by implementations (typically, network controllers) for LAG interface as the same information (e.g., LACP system-id) should be provided to the involved nodes.

The L2NM inherits the 'member-link-list' structure from the L2SM (including indication of OAM 802.3ah support [IEEE-802-3ah]).

```

...
+---rw vpn-nodes
  +---rw vpn-node* [vpn-node-id]
    ...
    +---rw vpn-network-accesses
      +---rw vpn-network-access* [id]
        ...
        +---rw connection
          +---rw l2-termination-point?
          |   string
          +---rw local-bridge-reference?
          |   string
          +---rw bearer-reference?          string
          |   {vpn-common:bearer-reference}?
          +---rw encapsulation
            +---rw encap-type?              identityref
            +---rw dot1q
              +---rw tag-type?              identityref
              +---rw cvlan-id?
              |   dot1q-types:vlanid
              +---rw tag-operations
                +---rw (op-choice)?
                |   +---:(pop)
                |   |   +---rw pop?          empty
                |   +---:(push)
                |   |   +---rw push?         empty
                |   +---:(translate)
                |   |   +---rw translate?    empty
                +---rw tag-1?

```

```

|         dot1q-types:vlanid
+---rw tag-1-type?
|         dot1q-types:dot1q-tag-type
+---rw tag-2?
|         dot1q-types:vlanid
+---rw tag-2-type?
|         dot1q-types:dot1q-tag-type
+---rw priority-tagged
| +---rw tag-type?  identityref
+---rw qinq
+---rw tag-type?      identityref
+---rw svlan-id
|         dot1q-types:vlanid
+---rw cvlan-id
|         dot1q-types:vlanid
+---rw tag-operations
+---rw (op-choice)?
| +---:(pop)
| | +---rw pop?      uint8
+---:(push)
| | +---rw push?     empty
+---:(translate)
| | +---rw translate? empty
+---rw tag-1?
|         dot1q-types:vlanid
+---rw tag-1-type?
|         dot1q-types:dot1q-tag-type
+---rw tag-2?
|         dot1q-types:vlanid
+---rw tag-2-type?
|         dot1q-types:dot1q-tag-type
+---rw lag-interface
| {vpn-common:lag-interface}?
+---rw lag-interface-id?  string
+---rw lacp
+---rw lacp-state?        boolean
+---rw mode?              identityref
+---rw speed?             uint32
+---rw mini-link-num?     uint32
+---rw system-id?
|         yang:mac-address
+---rw admin-key?         uint16
+---rw system-priority?   uint16
+---rw member-link-list
| +---rw member-link* [name]
| | +---rw name           string
| | +---rw speed?         uint32
| | +---rw mode?          identityref

```



```

|
|
|      +--rw link-mtu?      uint32
|      +--rw oam-802.3ah-link
|          |
|          +--rw enable?    boolean
|      +--rw flow-control?  boolean
|      +--rw lldp?          boolean
+--rw split-horizon
+--rw group-name?          string
...

```

Figure 15: Connection Subtree

#### 7.6.2. EVPN-VPWS Service Instance

The 'vpws-service-instance' provides the local and remote VPWS Service Instance (VSI) [RFC8214]. This container is only present when the 'vpn-type' is VPWS-EVPN. As shown in Figure 16, the VSIs can be configured by a VPN service provider or auto-generated.

An example to illustrate the use of the L2NM to configure VPWS-EVPN instances is provided in Appendix A.4.

```

...
+--rw vpn-nodes
  +--rw vpn-node* [vpn-node-id]
    ...
    +--rw vpn-network-accesses
      +--rw vpn-network-access* [id]
        ...
        +--rw (signaling-option)?
          +--:(bgp)
            +--rw (bgp-type)?
              +--:(l2vpn-bgp)
                |
                ...
                +--:(evpn-bgp)
                  +--rw vpws-service-instance
                    +--rw (local-vsi-choice)?
                      +--:(directly-assigned)
                        +--rw local-vpws-service-instance?
                          uint32
                      +--:(auto-assigned)
                        +--rw local-vsi-auto
                          +--rw (auto-mode)?
                            +--:(from-pool)
                              +--rw vsi-pool-name?
                                string
                            +--:(full-auto)
                              +--rw auto?          empty
                              +--ro auto-local-vsi? uint32
                        +--rw (remote-vsi-choice)?
                          +--:(directly-assigned)
                            +--rw remote-vpws-service-instance?
                              uint32
                          +--:(auto-assigned)
                            +--rw remote-vsi-auto
                              +--rw (auto-mode)?
                                +--:(from-pool)
                                  +--rw vsi-pool-name?
                                    string
                                +--:(full-auto)
                                  +--rw auto?          empty
                                  +--ro auto-remote-vsi? uint32
          ...

```

Figure 16: EVPN-VPWS Service Instance Subtree

### 7.6.3. Ethernet OAM

Ethernet OAM refers to both [IEEE-802-lag] and [ITU-T-Y-1731].

As shown in Figure 17, the L2NM inherits the same structure as in Section 5.3.2.2.6 of [RFC8466] for OAM matters.

```

+--rw l2vpn-ntw
+--rw vpn-profiles
|   ...
+--rw vpn-services
+--rw vpn-service* [vpn-id]
    ...
+--rw vpn-nodes
+--rw vpn-node* [vpn-node-id]
    ...
+--rw vpn-network-accesses
+--rw vpn-network-access* [id]
    ...
+--rw ethernet-service-oam
+--rw md-name?      string
+--rw md-level?     uint8
+--rw cfm-802.1-ag
+--rw n2-uni-c* [maid]
+--rw maid          string
+--rw mep-id?       uint32
+--rw mep-level?    uint32
+--rw mep-up-down?  enumeration
+--rw remote-mep-id? uint32
+--rw cos-for-cfm-pdus? uint32
+--rw ccm-interval? uint32
+--rw ccm-holdtime? uint32
+--rw ccm-p-bits-pri? ccm-priority-type
+--rw n2-uni-n* [maid]
+--rw maid          string
+--rw mep-id?       uint32
+--rw mep-level?    uint32
+--rw mep-up-down?  enumeration
+--rw remote-mep-id? uint32
+--rw cos-for-cfm-pdus? uint32
+--rw ccm-interval? uint32
+--rw ccm-holdtime? uint32
+--rw ccm-p-bits-pri? ccm-priority-type
+--rw y-1731* [maid]
+--rw maid          string
+--rw mep-id?       uint32
+--rw pm-type?      identityref
+--rw remote-mep-id? uint32

```

```

|      +---rw message-period?      uint32
|      +---rw measurement-interval? uint32
|      +---rw cos?                  uint32
|      +---rw loss-measurement?     boolean
|      +---rw synthethic-loss-measurement?
|              boolean
|      +---rw delay-measurement
|              +---rw enable-dm?    boolean
|              +---rw two-way?      boolean
|      +---rw frame-size?           uint32
|      +---rw session-type?         enumeration
...

```

Figure 17: OAM Subtree

#### 7.6.4. Services

The 'service' container (Figure 18) provides a set of service-specific configuration such as Quality of Service (QoS).

```

+---rw l2vpn-ntw
|   +---rw vpn-profiles
|       ...
|   +---rw vpn-services
|       +---rw vpn-service* [vpn-id]
|           ...
|       +---rw vpn-nodes
|           +---rw vpn-node* [vpn-node-id]
|               ...
|       +---rw vpn-network-accesses
|           +---rw vpn-network-access* [id]
|               ...
|       +---rw service
|           +---rw mtu?                uint32
|           +---rw svc-pe-to-ce-bandwidth
|               {vpn-common:inbound-bw}?
|               ...
|           +---rw svc-ce-to-pe-bandwidth
|               {vpn-common:outbound-bw}?
|               ...
|           +---rw qos {vpn-common:qos}?
|               ...
|           +---rw mac-policies
|               ...
|           +---rw broadcast-unknown-unicast-multicast
|               ...

```

Figure 18: Service Overall Subtree

The description of the service data nodes is as follows:

'mtu': Specifies the Layer 2 MTU for the VPN network access.

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth': Specify the service bandwidth for the L2VPN service.

'svc-pe-to-ce-bandwidth' indicates the inbound bandwidth of the connection (i.e., download bandwidth from the service provider to the site).

'svc-ce-to-pe-bandwidth' indicates the outbound bandwidth of the connection (i.e., upload bandwidth from the site to the service provider).

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth' can be represented using the Committed Information Rate (CIR), the Excess Information Rate (EIR), or the Peak Information Rate (PIR).

As shown in Figure 19, the structure of service bandwidth data nodes is inherited from the L2SM [RFC8466]. The following types, defined in [RFC9181], can be used to indicate the bandwidth type:

'bw-per-cos': The bandwidth is per Class of Service (CoS).

'bw-per-port': The bandwidth is per VPN network access.

'bw-per-site': The bandwidth is to all VPN network accesses that belong to the same site.

'bw-per-service': The bandwidth is per L2VPN service.

```

+---rw service
...
+---rw svc-pe-to-ce-bandwidth
    {vpn-common:inbound-bw}?
    +---rw pe-to-ce-bandwidth* [bw-type]
    +---rw bw-type          identityref
    +---rw (type)?
    +---:(per-cos)
    |   +---rw cos* [cos-id]
    |   |   +---rw cos-id      uint8
    |   |   +---rw cir?        uint64
    |   |   +---rw cbs?        uint64
    |   |   +---rw eir?        uint64
    |   |   +---rw ebs?        uint64
    |   |   +---rw pir?        uint64
    |   |   +---rw pbs?        uint64
    |   +---:(other)
    |   |   +---rw cir?        uint64
    |   |   +---rw cbs?        uint64
    |   |   +---rw eir?        uint64
    |   |   +---rw ebs?        uint64
    |   |   +---rw pir?        uint64
    |   |   +---rw pbs?        uint64
+---rw svc-ce-to-pe-bandwidth
    {vpn-common:outbound-bw}?
    +---rw ce-to-pe-bandwidth* [bw-type]
    +---rw bw-type          identityref
    +---rw (type)?
    +---:(per-cos)
    |   +---rw cos* [cos-id]
    |   |   +---rw cos-id      uint8
    |   |   +---rw cir?        uint64
    |   |   +---rw cbs?        uint64
    |   |   +---rw eir?        uint64
    |   |   +---rw ebs?        uint64
    |   |   +---rw pir?        uint64
    |   |   +---rw pbs?        uint64
    |   +---:(other)
    |   |   +---rw cir?        uint64
    |   |   +---rw cbs?        uint64
    |   |   +---rw eir?        uint64
    |   |   +---rw ebs?        uint64
    |   |   +---rw pir?        uint64
    |   |   +---rw pbs?        uint64
...

```

Figure 19: Service Bandwidth Subtree

'qos': Is used to define a set of QoS policies to apply on a given VPN network access (Figure 20). The QoS classification can be based on many criteria such as source MAC address, destination MAC address, etc. See also Section 5.10.2.1 of [RFC8466] for more discussion of QoS classification including the use of color types.

```

+--rw service
...
+--rw qos {vpn-common:qos}?
|
|   +--rw qos-classification-policy
|   |
|   |   +--rw rule* [id]
|   |   |
|   |   |   +--rw id string
|   |   |   +--rw (match-type)?
|   |   |   |
|   |   |   |   +--:(match-flow)
|   |   |   |   |
|   |   |   |   |   +--rw match-flow
|   |   |   |   |   |
|   |   |   |   |   |   +--rw dscp? inet:dscp
|   |   |   |   |   |   +--rw dot1q? uint16
|   |   |   |   |   |   +--rw pcp? uint8
|   |   |   |   |   |   +--rw src-mac-address?
|   |   |   |   |   |   |   yang:mac-address
|   |   |   |   |   |   +--rw dst-mac-address?
|   |   |   |   |   |   |   yang:mac-address
|   |   |   |   |   |   +--rw color-type?
|   |   |   |   |   |   |   identityref
|   |   |   |   |   |   +--rw any? empty
|   |   |   |   |   +--:(match-application)
|   |   |   |   |   |
|   |   |   |   |   |   +--rw match-application?
|   |   |   |   |   |   |   identityref
|   |   |   |   +--rw target-class-id? string
|   |   +--rw qos-profile
|   |   |
|   |   |   +--rw qos-profile* [profile]
|   |   |   |
|   |   |   |   +--rw profile leafref
|   |   |   |   +--rw direction? identityref
|   |
|   |   ...
|
|   ...

```

Figure 20: QoS Subtree

'mac-policies': Lists a set of MAC-related policies such as MAC ACLs. Similar to [RFC8519], an ACL match can be based upon source MAC address, source MAC address mask, destination MAC address, destination MAC address mask, or a combination thereof.

A data frame that matches an ACL can be dropped, flooded, or trigger an alarm. A rate-limit policy can be defined for handling frames that match an ACL entry with 'flood' action.

When 'mac-loop-prevention' or 'mac-addr-limit' data nodes are provided, they take precedence over the ones included in the 'global-parameters-profile' at the VPN service or VPN node levels.

```

+--rw service
...
+--rw mac-policies
|
|   +--rw access-control-list* [name]
|   |
|   |   +--rw name                               string
|   |   +--rw src-mac-address*
|   |   |   yang:mac-address
|   |   +--rw src-mac-address-mask*
|   |   |   yang:mac-address
|   |   +--rw dst-mac-address*
|   |   |   yang:mac-address
|   |   +--rw dst-mac-address-mask*
|   |   |   yang:mac-address
|   |   +--rw action?                            identityref
|   |   +--rw rate-limit?                        decimal64
|   +--rw mac-loop-prevention
|   |
|   |   +--rw window?                            uint32
|   |   +--rw frequency?                         uint32
|   |   +--rw retry-timer?                       uint32
|   |   +--rw protection-type?                   identityref
|   +--rw mac-addr-limit
|   |
|   |   +--rw limit-number?                       uint16
|   |   +--rw time-interval?                     uint32
|   |   +--rw action?                            identityref
|   ...
...

```

Figure 21: MAC Policies Subtree

'broadcast-unknown-unicast-multicast': Defines the type of site in the customer multicast service topology: source, receiver, or both. It is also used to define multicast group-to-port mappings.

```

+--rw service
...
+--rw broadcast-unknown-unicast-multicast
|
|   +--rw multicast-site-type?
|   |   enumeration
|   +--rw multicast-gp-address-mapping* [id]
|   |
|   |   +--rw id                                uint16
|   |   +--rw vlan-id                          uint32
|   |   +--rw mac-gp-address
|   |   |   yang:mac-address
|   |   +--rw port-lag-number?                 uint32
|   +--rw bum-overall-rate?                    uint64

```



Figure 22: BUM Subtree

## 8. YANG Modules

### 8.1. IANA-Maintained Module for BGP Layer 2 Encapsulation Types

The "iana-bgp-l2-encaps" YANG module echoes the registry available at [IANA-BGP-L2].

This module references [RFC3032], [RFC4446], [RFC4448], [RFC4553], [RFC4618], [RFC4619], [RFC4717], [RFC4761], [RFC4816], [RFC4842], and [RFC5086].

<CODE BEGINS>

```
file "iana-bgp-l2-encaps@2021-07-05.yang"
```

```
module iana-bgp-l2-encaps {  
  yang-version 1.1;  
  namespace "urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps";  
  prefix iana-bgp-l2-encaps;
```

```
  organization
```

```
    "IANA";
```

```
  contact
```

```
    "Internet Assigned Numbers Authority
```

```
    Postal: ICANN
```

```
      12025 Waterfront Drive, Suite 300
```

```
      Los Angeles, CA 90094-2536
```

```
      United States of America
```

```
    Tel: +1 310 301 5800
```

```
    <mailto:iana@iana.org>;
```

```
  description
```

```
    "This module contains a collection of IANA-maintained YANG  
    data types that are used for referring to BGP Layer 2  
    encapsulation types."
```

```
  Copyright (c) 2022 IETF Trust and the persons identified as  
  authors of the code. All rights reserved.
```

```
  Redistribution and use in source and binary forms, with or  
  without modification, is permitted pursuant to, and subject  
  to the license terms contained in, the Revised BSD License  
  set forth in Section 4.c of the IETF Trust's Legal Provisions  
  Relating to IETF Documents  
  (https://trustee.ietf.org/license-info).
```

```
  This version of this YANG module is part of RFC XXXX; see
```

```
    the RFC itself for full legal notices.";

revision 2021-07-05 {
  description
    "First revision.";
  reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
}

identity bgp-l2-encaps-type {
  description
    "Base BGP Layer 2 encapsulation type.";
  reference
    "RFC 6624: Layer 2 Virtual Private Networks Using BGP for
      Auto-Discovery and Signaling";
}

identity frame-relay {
  base bgp-l2-encaps-type;
  description
    "Frame Relay.";
  reference
    "RFC 4446: IANA Allocations for Pseudowire Edge
      to Edge Emulation (PWE3)";
}

identity atm-aal5 {
  base bgp-l2-encaps-type;
  description
    "ATM AAL5 SDU VCC transport.";
  reference
    "RFC 4446: IANA Allocations for Pseudowire Edge
      to Edge Emulation (PWE3)";
}

identity atm-cell {
  base bgp-l2-encaps-type;
  description
    "ATM transparent cell transport.";
  reference
    "RFC 4816: Pseudowire Emulation Edge-to-Edge (PWE3)
      Asynchronous Transfer Mode (ATM) Transparent
      Cell Transport Service";
}

identity ethernet-tagged-mode {
  base bgp-l2-encaps-type;
  description
```

```
        "Ethernet (VLAN) Tagged Mode.";
    reference
        "RFC 4448: Encapsulation Methods for Transport of Ethernet
        over MPLS Networks";
}

identity ethernet-raw-mode {
    base bgp-l2-encaps-type;
    description
        "Ethernet Raw Mode.";
    reference
        "RFC 4448: Encapsulation Methods for Transport of Ethernet
        over MPLS Networks";
}

identity hdlc {
    base bgp-l2-encaps-type;
    description
        "Cisco HDLC.";
    reference
        "RFC 4618: Encapsulation Methods for Transport of
        PPP/High-Level Data Link Control (HDLC)
        over MPLS Networks";
}

identity ppp {
    base bgp-l2-encaps-type;
    description
        "PPP.";
    reference
        "RFC 4618: Encapsulation Methods for Transport of
        PPP/High-Level Data Link Control (HDLC)
        over MPLS Networks";
}

identity circuit-emulation {
    base bgp-l2-encaps-type;
    description
        "SONET/SDH Circuit Emulation Service.";
    reference
        "RFC 4842: Synchronous Optical Network/Synchronous Digital
        Hierarchy (SONET/SDH) Circuit Emulation over Packet
        (CEP)";
}

identity atm-to-vcc {
    base bgp-l2-encaps-type;
    description
```

```
    "ATM n-to-one VCC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}

identity atm-to-vpc {
  base bgp-l2-encaps-type;
  description
    "ATM n-to-one VPC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}

identity layer-2-transport {
  base bgp-l2-encaps-type;
  description
    "IP Layer 2 Transport.";
  reference
    "RFC 3032: MPLS Label Stack Encoding";
}

identity fr-port-mode {
  base bgp-l2-encaps-type;
  description
    "Frame Relay Port mode.";
  reference
    "RFC 4619: Encapsulation Methods for Transport of Frame Relay
      over Multiprotocol Label Switching (MPLS)
      Networks";
}

identity e1 {
  base bgp-l2-encaps-type;
  description
    "Structure-agnostic E1 over packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
      over Packet (SAToP)";
}

identity t1 {
  base bgp-l2-encaps-type;
  description
    "Structure-agnostic T1 (DS1) over packet.";
```

```
    reference
      "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
        over Packet (SAToP)";
  }

  identity vpls {
    base bgp-l2-encaps-type;
    description
      "VPLS.";
    reference
      "RFC 4761: Virtual Private LAN Service (VPLS)
        Using BGP for Auto-Discovery and Signaling";
  }

  identity t3 {
    base bgp-l2-encaps-type;
    description
      "Structure-agnostic T3 (DS3) over packet.";
    reference
      "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
        over Packet (SAToP)";
  }

  identity structure-aware {
    base bgp-l2-encaps-type;
    description
      "Nx64kbit/s Basic Service using Structure-aware.";
    reference
      "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
        Circuit Emulation Service over Packet Switched
        Network (CESoPSN)";
  }

  identity dlci {
    base bgp-l2-encaps-type;
    description
      "Frame Relay DLCI.";
    reference
      "RFC 4619: Encapsulation Methods for Transport of Frame Relay
        over Multiprotocol Label Switching (MPLS)
        Networks";
  }

  identity e3 {
    base bgp-l2-encaps-type;
    description
      "Structure-agnostic E3 over packet.";
    reference
```

```
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
          over Packet (SAToP)";
    }

    identity ds1 {
        base bgp-l2-encaps-type;
        description
            "Octet-aligned payload for Structure-agnostic DS1 circuits.";
        reference
            "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
              over Packet (SAToP)";
    }

    identity cas {
        base bgp-l2-encaps-type;
        description
            "E1 Nx64kbit/s with CAS using Structure-aware.";
        reference
            "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
              Circuit Emulation Service over Packet Switched
              Network (CESoPSN)";
    }

    identity esf {
        base bgp-l2-encaps-type;
        description
            "DS1 (ESF) Nx64kbit/s with CAS using Structure-aware.";
        reference
            "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
              Circuit Emulation Service over Packet Switched
              Network (CESoPSN)";
    }

    identity sf {
        base bgp-l2-encaps-type;
        description
            "DS1 (SF) Nx64kbit/s with CAS using Structure-aware.";
        reference
            "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
              Circuit Emulation Service over Packet Switched
              Network (CESoPSN)";
    }
}
<CODE ENDS>
```

## 8.2. IANA-Maintained Module for Pseudowire Types

The initial version of the "iana-pseudowire-types" YANG module echoes the registry available at [IANA-PW-Types].

This module references [MFA], [RFC2507], [RFC2508], [RFC3032], [RFC3545], [RFC4448], [RFC4618], [RFC4619], [RFC4717], [RFC4842], [RFC4863], [RFC4901], [RFC5086], [RFC5087], [RFC5143], [RFC5795], and [RFC6307].

<CODE BEGINS>

```
file "iana-pseudowire-types@2021-07-05.yang"
```

```
module iana-pseudowire-types {
```

```
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:iana-pseudowire-types";
```

```
  prefix iana-pw-types;
```

```
  organization
```

```
    "IANA";
```

```
  contact
```

```
    "Internet Assigned Numbers Authority
```

```
    Postal: ICANN
```

```
      12025 Waterfront Drive, Suite 300
```

```
      Los Angeles, CA 90094-2536
```

```
      United States of America
```

```
    Tel: +1 310 301 5800
```

```
    <mailto:iana@iana.org>;
```

```
  description
```

```
    "This module contains a collection of IANA-maintained YANG  
    data types that are used for referring to Pseudowire Types.
```

```
    Copyright (c) 2022 IETF Trust and the persons identified as  
    authors of the code. All rights reserved.
```

```
    Redistribution and use in source and binary forms, with or  
    without modification, is permitted pursuant to, and subject  
    to the license terms contained in, the Revised BSD License  
    set forth in Section 4.c of the IETF Trust's Legal Provisions  
    Relating to IETF Documents  
    (https://trustee.ietf.org/license-info).
```

```
    This version of this YANG module is part of RFC XXXX; see  
    the RFC itself for full legal notices."
```

```
  revision 2021-07-05 {
```

```
    description
```

```
      "First revision.";
```

```
    reference
      "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
  }

  identity iana-pw-types {
    description
      "Base Pseudowire Layer 2 encapsulation type.";
  }

  identity frame-relay {
    base iana-pw-types;
    description
      "Frame Relay DLCI (Martini Mode).";
    reference
      "RFC 4619: Encapsulation Methods for Transport of Frame Relay
        over Multiprotocol Label Switching (MPLS)
        Networks";
  }

  identity atm-aal5 {
    base iana-pw-types;
    description
      "ATM AAL5 SDU VCC transport.";
    reference
      "RFC 4717: Encapsulation Methods for Transport of
        Asynchronous Transfer Mode (ATM) over MPLS
        Networks";
  }

  identity atm-cell {
    base iana-pw-types;
    description
      "ATM transparent cell transport.";
    reference
      "RFC 4717: Encapsulation Methods for Transport of
        Asynchronous Transfer Mode (ATM) over MPLS
        Networks";
  }

  identity ethernet-tagged-mode {
    base iana-pw-types;
    description
      "Ethernet (VLAN) Tagged Mode.";
    reference
      "RFC 4448: Encapsulation Methods for Transport of Ethernet
        over MPLS Networks";
  }
```



```
identity ethernet {
  base iana-pw-types;
  description
    "Ethernet.";
  reference
    "RFC 4448: Encapsulation Methods for Transport of Ethernet
      over MPLS Networks";
}

identity hdlc {
  base iana-pw-types;
  description
    "HDLC.";
  reference
    "RFC 4618: Encapsulation Methods for Transport of
      PPP/High-Level Data Link Control (HDLC)
      over MPLS Networks";
}

identity ppp {
  base iana-pw-types;
  description
    "PPP.";
  reference
    "RFC 4618: Encapsulation Methods for Transport of
      PPP/High-Level Data Link Control (HDLC)
      over MPLS Networks";
}

identity circuit-emulation-mpls {
  base iana-pw-types;
  description
    "SONET/SDH Circuit Emulation Service Over MPLS Encapsulation.";
  reference
    "RFC 5143: Synchronous Optical Network/Synchronous Digital
      Hierarchy (SONET/SDH) Circuit Emulation Service over
      MPLS (CEM) Encapsulation";
}

identity atm-to-vcc {
  base iana-pw-types;
  description
    "ATM n-to-one VCC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}
```

```
identity atm-to-vpc {
  base iana-pw-types;
  description
    "ATM n-to-one VPC cell transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}

identity layer-2-transport {
  base iana-pw-types;
  description
    "IP Layer2 Transport.";
  reference
    "RFC 3032: MPLS Label Stack Encoding";
}

identity atm-one-to-one-vcc {
  base iana-pw-types;
  description
    "ATM one-to-one VCC Cell Mode.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}

identity atm-one-to-one-vpc {
  base iana-pw-types;
  description
    "ATM one-to-one VPC Cell Mode.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}

identity atm-aal5-vcc {
  base iana-pw-types;
  description
    "ATM AAL5 PDU VCC transport.";
  reference
    "RFC 4717: Encapsulation Methods for Transport of
      Asynchronous Transfer Mode (ATM) over MPLS
      Networks";
}
```

```
identity fr-port-mode {
  base iana-pw-types;
  description
    "Frame-Relay Port mode.";
  reference
    "RFC 4619: Encapsulation Methods for Transport of Frame Relay
      over Multiprotocol Label Switching (MPLS)
      Networks";
}

identity circuit-emulation-packet {
  base iana-pw-types;
  description
    "SONET/SDH Circuit Emulation over Packet.";
  reference
    "RFC 4842: Synchronous Optical Network/Synchronous Digital
      Hierarchy (SONET/SDH) Circuit Emulation over Packet
      (CEP)";
}

identity e1 {
  base iana-pw-types;
  description
    "Structure-agnostic E1 over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
      over Packet (SAToP)";
}

identity t1 {
  base iana-pw-types;
  description
    "Structure-agnostic T1 (DS1) over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
      over Packet (SAToP)";
}

identity e3 {
  base iana-pw-types;
  description
    "Structure-agnostic E3 over Packet.";
  reference
    "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
      over Packet (SAToP)";
}

identity t3 {
```

```
    base iana-pw-types;
    description
        "Structure-agnostic T3 (DS3) over Packet.";
    reference
        "RFC 4553: Structure-Agnostic Time Division Multiplexing (TDM)
        over Packet (SAToP)";
}

identity ces-over-psn {
    base iana-pw-types;
    description
        "CESoPSN basic mode.";
    reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
        Circuit Emulation Service over Packet Switched
        Network (CESoPSN)";
}

identity tdm-over-ip-aal1 {
    base iana-pw-types;
    description
        "TDMoIP AAL1 Mode.";
    reference
        "RFC 5087: Time Division Multiplexing over IP (TDMoIP)";
}

identity ces-over-psn-cas {
    base iana-pw-types;
    description
        "CESoPSN TDM with CAS.";
    reference
        "RFC 5086: Structure-Aware Time Division Multiplexed (TDM)
        Circuit Emulation Service over Packet Switched
        Network (CESoPSN)";
}

identity tdm-over-ip-aal2 {
    base iana-pw-types;
    description
        "TDMoIP AAL2 Mode.";
    reference
        "RFC 5087: Time Division Multiplexing over IP (TDMoIP)";
}

identity dlci {
    base iana-pw-types;
    description
        "Frame Relay DLCI.";
```

```
reference
  "RFC 4619: Encapsulation Methods for Transport of Frame Relay
    over Multiprotocol Label Switching (MPLS)
    Networks";
}

identity rohc {
  base iana-pw-types;
  description
    "ROHC Transport Header-compressed Packets.";
  reference
    "RFC 5795: The RObust Header Compression (ROHC) Framework
    RFC 4901: Protocol Extensions for Header Compression over
    MPLS";
}

identity ecrtmp {
  base iana-pw-types;
  description
    "ECRTP Transport Header-compressed Packets.";
  reference
    "RFC 3545: Enhanced Compressed RTP (CRTP) for Links with High
    Delay, Packet Loss and Reordering
    RFC 4901: Protocol Extensions for Header Compression over
    MPLS";
}

identity iphc {
  base iana-pw-types;
  description
    "IPHC Transport Header-compressed Packets.";
  reference
    "RFC 2507: IP Header Compression
    RFC 4901: Protocol Extensions for Header Compression over
    MPLS";
}

identity crtp {
  base iana-pw-types;
  description
    "cRTP Transport Header-compressed Packets.";
  reference
    "RFC 2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial
    Links
    RFC 4901: Protocol Extensions for Header Compression over
    MPLS";
}
```

```
identity atm-vp-virtual-trunk {
  base iana-pw-types;
  description
    "ATM VP Virtual Trunk.";
  reference
    "MFA Forum: The Use of Virtual Trunks for ATM/MPLS
      Control Plane Interworking Specification";
}

identity fc-port-mode {
  base iana-pw-types;
  description
    "FC Port Mode.";
  reference
    "RFC 6307: Encapsulation Methods for Transport of
      Fibre Channel Traffic over MPLS Networks";
}

identity wildcard {
  base iana-pw-types;
  description
    "Wildcard.";
  reference
    "RFC 4863: Wildcard Pseudowire Type";
}
}
<CODE ENDS>
```

### 8.3. Ethernet Segments

The "ietf-ethernet-segment" YANG module uses types defined in [RFC6991].

```
<CODE BEGINS>
file "ietf-ethernet-segment@2021-10-01.yang"
module iETF-ethernet-segment {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ethernet-segment";
  prefix l2vpn-es;

  import iETF-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types, Section 3";
  }

  organization
    "IETF OPSA (Operations and Management Area) Working Group";
```

## contact

"WG Web: <<https://datatracker.ietf.org/wg/opsawg/>>  
WG List: <<mailto:opsawg@ietf.org>>  
  
Editor: Mohamed Boucadair  
<<mailto:mohamed.boucadair@orange.com>>  
Editor: Samier Barguil  
<<mailto:samier.barguilgiraldo.ext@telefonica.com>>  
Author: Oscar Gonzalez de Dios  
<<mailto:oscar.gonzalezdedios@telefonica.com>>";

## description

"This YANG module defines a model for Ethernet Segments.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

## revision 2021-10-01 {

## description

"Initial version.";

## reference

"RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";

}

/\* Identities \*/

## identity esi-type {

## description

"T-(Ethernet Segment Identifier (ESI) Type) is a 1-octet field (most significant octet) that specifies the format of the remaining 9 octets (ESI Value).";

## reference

"RFC 7432: BGP MPLS-Based Ethernet VPN, Section 5";

}

## identity esi-type-0-operator {

## base esi-type;

## description

"This type indicates an arbitrary 9-octet ESI value,

```
        which is managed and configured by the operator.";
    }

    identity esi-type-1-lacp {
        base esi-type;
        description
            "When IEEE 802.1AX Link Aggregation Control Protocol (LACP)
            is used between the Provider Edge (PE) and Customer Edge (CE)
            devices, this ESI type indicates an auto-generated ESI value
            determined from LACP.";
        reference
            "IEEE Std. 802.1AX: Link Aggregation";
    }

    identity esi-type-2-bridge {
        base esi-type;
        description
            "The ESI value is auto-generated and determined based
            on the Layer 2 bridge protocol.";
    }

    identity esi-type-3-mac {
        base esi-type;
        description
            "This type indicates a MAC-based ESI value that can be
            auto-generated or configured by the operator.";
    }

    identity esi-type-4-router-id {
        base esi-type;
        description
            "This type indicates a Router ID ESI value that can be
            auto-generated or configured by the operator.";
    }

    identity esi-type-5-asn {
        base esi-type;
        description
            "This type indicates an Autonomous System (AS)-based ESI value
            that can be auto-generated or configured by the operator.";
    }

    identity df-election-methods {
        description
            "Base Identity Designated Forwarder (DF) election method.";
    }

    identity default-7432 {
```



```
base df-election-methods;
description
    "The default DF election method.

    The default procedure for DF election at the granularity of
    <ES,VLAN> for VLAN-based service or <ES, VLAN bundle> for
    VLAN-(aware) bundle service is referred to as
    'service carving'.";
reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.5";
}

identity highest-random-weight {
    base df-election-methods;
    description
        "The highest random weight (HRW) method.";
    reference
        "RFC 8584: Framework for Ethernet VPN Designated
        Forwarder Election Extensibility, Section 3";
}

identity preference {
    base df-election-methods;
    description
        "The preference based method. PEs are assigned with
        preferences to become the DF in the Ethernet Segment (ES).
        The exact preference-based algorithm (e.g., lowest-preference
        algorithm, highest-preference algorithm) to use is
        signaled at the control plane.";
}

identity es-redundancy-mode {
    description
        "Base identity for ES redundancy modes.";
}

identity single-active {
    base es-redundancy-mode;
    description
        "Indicates Single-Active redundancy mode for a given ES.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1.1";
}

identity all-active {
    base es-redundancy-mode;
    description
        "Indicates All-Active redundancy mode for a given ES.";
```

```
reference
  "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1.2";
}

/* Main Ethernet Segment Container */

container ethernet-segments {
  description
    "Top container for the Ethernet Segment Identifier (ESI).";
  list ethernet-segment {
    key "name";
    description
      "Top list for ESIs.";
    leaf name {
      type string;
      description
        "Includes the name of the Ethernet Segment (ES).";
    }
    leaf esi-type {
      type identityref {
        base esi-type;
      }
      default "esi-type-0-operator";
      description
        "T-(ESI Type) is a 1-octet field (most significant
         octet) that specifies the format of the remaining
         9 octets (ESI Value).";
      reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 5";
    }
    choice esi-choice {
      description
        "Ethernet segment choice between several types.
        For ESI Type 0: The esi is directly configured by the
        operator.
        For ESI Type 1: The auto-mode must be used.
        For ESI Type 2: The auto-mode must be used.
        For ESI Type 3: The directly-assigned or auto-mode must
        be used.
        For ESI Type 4: The directly-assigned or auto-mode must
        be used.
        For ESI Type 5: The directly-assigned or auto-mode must
        be used.";
      case directly-assigned {
        description
          "Explicitly assign an ESI value.";
        leaf ethernet-segment-identifier {
          type yang:hex-string {
```

```
        length "29";
    }
    description
        "10-octet ESI.";
}
}
case auto-assigned {
    description
        "The ESI is auto-assigned.";
    container esi-auto {
        description
            "The ESI is auto-assigned.";
        choice auto-mode {
            description
                "Indicates the auto-assignment mode. ESI can be
                automatically assigned either with or without
                indicating a pool from which the ESI should be
                taken.

                For both cases, the server will auto-assign an
                ESI value 'auto-assigned-ESI' and use that value
                operationally.";
            case from-pool {
                leaf esi-pool-name {
                    type string;
                    description
                        "The auto-assignment will be made from the
                        pool identified by the ESI-pool-name.";
                }
            }
            case full-auto {
                leaf auto {
                    type empty;
                    description
                        "Indicates an ESI is fully auto-assigned.";
                }
            }
        }
    }
}
leaf auto-ethernet-segment-identifier {
    type yang:hex-string {
        length "29";
    }
    config false;
    description
        "The value of the auto-assigned ESI.";
}
}
```

```
}
leaf esi-redundancy-mode {
  type identityref {
    base es-redundancy-mode;
  }
  description
    "Indicates the ES redundancy mode.";
  reference
    "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 14.1";
}
container df-election {
  description
    "Top container for the DF election method properties.";
  leaf df-election-method {
    type identityref {
      base df-election-methods;
    }
    default "default-7432";
    description
      "Specifies the DF election method.";
    reference
      "RFC 8584: Framework for Ethernet VPN Designated
        Forwarder Election Extensibility";
  }
  leaf revertive {
    when "derived-from-or-self(..df-election-method, "
      + "'preference') " {
      description
        "The revertive value is only applicable
          to the preference method.";
    }
    type boolean;
    default "true";
    description
      "The 'preempt' or 'revertive' behavior. This
        option allows a non-revertive behavior in the
        DF election.";
    reference
      "RFC 8584: Framework for Ethernet VPN Designated
        Forwarder Election Extensibility";
  }
  leaf election-wait-time {
    type uint32;
    units "seconds";
    default "3";
    description
      "Election wait timer.";
    reference
```

```
        "RFC 8584: Framework for Ethernet VPN Designated
          Forwarder Election Extensibility";
    }
}
leaf split-horizon-filtering {
    type boolean;
    description
        "Controls split-horizon filtering. It is enabled
         when set to 'true'."

        In order to achieve split-horizon filtering, every
        Broadcast, unknown unicast, or multicast (BUM)
        packet originating from a non-DF PE is encapsulated
        with an MPLS label that identifies the origin ES.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 8.3";
}
container pbb {
    description
        "Provider Backbone Bridging (PBB) parameters .";
    reference
        "IEEE 802.1ah: Provider Backbone Bridge";
    leaf backbone-src-mac {
        type yang:mac-address;
        description
            "The PEs connected to the same CE must share the
             same Provider Backbone (B-MAC) address in
             All-Active mode.";
        reference
            "RFC 7623: Provider Backbone Bridging Combined with
             Ethernet VPN (PBB-EVPN), Section 6.2.1.1";
    }
}
list member {
    key "ne-id interface-id";
    description
        "Includes a list of ES members.";
    leaf ne-id {
        type string;
        description
            "An identifier of the network element where the ES
             is configured within a service provider network.";
    }
    leaf interface-id {
        type string;
        description
            "Identifier of a node interface.";
    }
}
```

```
    }  
  }  
}  
}  
<CODE ENDS>
```

#### 8.4. L2NM

The "ietf-l2vpn-ntw" YANG module uses types defined in [RFC6991], [RFC9181], [RFC8294], and [IEEE802.1Qcp-2018].

```
<CODE BEGINS>  
file "ietf-l2vpn-ntw@2022-04-29.yang"  
module ietf-l2vpn-ntw {  
  yang-version 1.1;  
  namespace "urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw";  
  prefix l2vpn-ntw;  
  
  import ietf-inet-types {  
    prefix inet;  
    reference  
      "RFC 6991: Common YANG Data Types, Section 4";  
  }  
  import ietf-yang-types {  
    prefix yang;  
    reference  
      "RFC 6991: Common YANG Data Types, Section 3";  
  }  
  import ietf-vpn-common {  
    prefix vpn-common;  
    reference  
      "RFC 9181: A Common YANG for Data Model for Layer 2  
        and Layer 3 VPNs";  
  }  
  import iana-bgp-l2-encaps {  
    prefix iana-bgp-l2-encaps;  
    reference  
      "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";  
  }  
  import iana-pseudowire-types {  
    prefix iana-pw-types;  
    reference  
      "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";  
  }  
  import ietf-routing-types {  
    prefix rt-types;  
    reference  
      "RFC 8294: Common YANG Data Types for the Routing Area";  
  }  
}
```

```
}
import ieee802-dot1q-types {
  prefix dot1q-types;
  reference
    "IEEE Std 802.1Qcp-2018: Bridges and Bridged Networks -
      Amendment: YANG Data Model";
}

organization
  "IETF OPSA (Operations and Management Area) Working Group";
contact
  "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
  WG List:  <mailto:opsawg@ietf.org>

  Editor:    Mohamed Boucadair
             <mailto:mohamed.boucadair@orange.com>
  Editor:    Samier Barguil
             <mailto:samier.barguilgiraldo.ext@telefonica.com>
  Author:    Oscar Gonzalez de Dios
             <mailto:oscar.gonzalezdedios@telefonica.com>";
description
  "This YANG module defines a network model for Layer 2 VPN
  services.

  Copyright (c) 2022 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Revised BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision 2022-04-29 {
  description
    "Initial version.";
  reference
    "RFC XXXX: A YANG Network Data Model for Layer 2 VPNs.";
}

/* Features */

feature oam-3ah {
  description
```

```
        "Indicates the support of OAM 802.3ah.";
    reference
        "IEEE Std 802.3ah: Media Access Control Parameters, Physical
          Layers, and Management Parameters for
          Subscriber Access Networks";
}

/* Identities */

identity evpn-service-interface-type {
    description
        "Base identity for EVPN service interface type.";
}

identity vlan-based-service-interface {
    base evpn-service-interface-type;
    description
        "VLAN-Based Service Interface.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.1";
}

identity vlan-bundle-service-interface {
    base evpn-service-interface-type;
    description
        "VLAN Bundle Service Interface.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.2";
}

identity vlan-aware-bundle-service-interface {
    base evpn-service-interface-type;
    description
        "VLAN-Aware Bundle Service Interface.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN, Section 6.3";
}

identity mapping-type {
    base vpn-common:multicast-gp-address-mapping;
    description
        "Identity for multicast group mapping type.";
}

identity loop-prevention-type {
    description
        "Identity of loop prevention.";
}
```



```
identity shut {
  base loop-prevention-type;
  description
    "Shut protection type.";
}

identity trap {
  base loop-prevention-type;
  description
    "Trap protection type.";
}

identity color-type {
  description
    "Identity of color types. A type is assigned to a service frame
    to identify its QoS profile conformance.";
}

identity green {
  base color-type;
  description
    "'green' color type. A service frame is 'green' if it is
    conformant with the committed rate of the bandwidth profile.";
}

identity yellow {
  base color-type;
  description
    "'yellow' color type. A service frame is 'yellow' if it exceeds
    the committed rate but is conformant with the excess rate
    of the bandwidth profile.";
}

identity red {
  base color-type;
  description
    "'red' color type. A service frame is 'red' if it is not
    conformant with both the committed and excess rates of the
    bandwidth profile.";
}

identity t-ldp-pw-type {
  description
    "Identity for t-ldp-pw-type.";
}

identity vpws-type {
  base t-ldp-pw-type;
```

```
description
  "Virtual Private Wire Service (VPWS) t-ldp-pw-type.";
reference
  "RFC 4664: Framework for Layer 2 Virtual Private Networks
    (L2VPNs), Section 3.3";
}

identity vpls-type {
  base t-ldp-pw-type;
  description
    "Virtual Private LAN Service (VPLS) t-ldp-pw-type.";
  reference
    "RFC 4762: Virtual Private LAN Service (VPLS) Using
      Label Distribution Protocol (LDP)
      Signaling, Section 6.1";
}

identity hvpls {
  base t-ldp-pw-type;
  description
    "Identity for Hierarchical Virtual Private LAN Service (H-VPLS)
      t-ldp-pw-type.";
  reference
    "RFC 4762: Virtual Private LAN Service (VPLS) Using
      Label Distribution Protocol (LDP)
      Signaling, Section 10";
}

identity lacp-mode {
  description
    "Identity of the LACP mode.";
}

identity lacp-active {
  base lacp-mode;
  description
    "LACP active mode.

    This mode refers to the mode where auto-speed negotiation
    is initiated followed by an establishment of an
    Ethernet channel with the other end.";
}

identity lacp-passive {
  base lacp-mode;
  description
    "LACP passive mode.
```

```
        This mode refers to the LACP mode where an endpoint does
        not initiate the negotiation, but only responds to LACP
        packets initiated by the other end (e.g., full duplex
        or half duplex)";
    }

    identity pm-type {
        description
            "Identity for performance monitoring type.";
    }

    identity loss {
        base pm-type;
        description
            "Loss measurement is the performance monitoring type.";
    }

    identity delay {
        base pm-type;
        description
            "Delay measurement is the performance monitoring type.";
    }

    identity mac-learning-mode {
        description
            "Media Access Control (MAC) learning mode.";
    }

    identity data-plane {
        base mac-learning-mode;
        description
            "User MAC addresses are learned through ARP broadcast.";
    }

    identity control-plane {
        base mac-learning-mode;
        description
            "User MAC addresses are advertised through EVPN-BGP.";
    }

    identity mac-action {
        description
            "Base identity for a MAC action.";
    }

    identity drop {
        base mac-action;
        description
```

```
    "Dropping a packet as the MAC action.";
}

identity flood {
    base mac-action;
    description
        "Packet flooding as the MAC action.";
}

identity warning {
    base mac-action;
    description
        "Log a warning message as the MAC action.";
}

identity precedence-type {
    description
        "Redundancy type. The service can be created
        with primary and secondary signalization.";
}

identity primary {
    base precedence-type;
    description
        "Identifies the main VPN network access.";
}

identity secondary {
    base precedence-type;
    description
        "Identifies the secondary VPN network access.";
}

identity ldp-pw-type {
    description
        "Identity for allowed LDP-based pseudowire (PW) type.";
    reference
        "RFC 4762: Virtual Private LAN Service (VPLS) Using
        Label Distribution Protocol (LDP)
        Signaling, Section 6.1.1";
}

identity ethernet {
    base ldp-pw-type;
    description
        "PW Ethernet type.";
}
```

```
identity ethernet-tagged {
    base ldp-pw-type;
    description
        "PW Ethernet tagged mode type.";
}

/* Typedefs */

typedef ccm-priority-type {
    type uint8 {
        range "0..7";
    }
    description
        "A 3-bit priority value to be used in the VLAN tag,
        if present in the transmitted frame. A larger value
        indicates a higher priority.";
}

/* Groupings */

grouping cfm-802 {
    description
        "Grouping for 802.1ag Connectivity Fault Management (CFM)
        attributes.";
    reference
        "IEEE Std 802-1ag: Virtual Bridged Local Area Networks
        Amendment 5: Connectivity Fault Management";
    leaf maid {
        type string;
        description
            "Maintenance Association Identifier (MAID).";
    }
    leaf mep-id {
        type uint32;
        description
            "Local Maintenance Entity Group End Point (MEP) ID.";
    }
    leaf mep-level {
        type uint32;
        description
            "MEP level.";
    }
    leaf mep-up-down {
        type enumeration {
            enum up {
                description
                    "MEP is up.";
            }
        }
    }
}
```

```
        enum down {
            description
                "MEP is down.";
        }
    }
    default "up";
    description
        "MEP up/down.";
}
leaf remote-mep-id {
    type uint32;
    description
        "Remote MEP ID.";
}
leaf cos-for-cfm-pdus {
    type uint32;
    description
        "Class of service for CFM PDUs.";
}
leaf ccm-interval {
    type uint32;
    units "milliseconds";
    default "10000";
    description
        "Continuity Check Message (CCM) interval.";
}
leaf ccm-holdtime {
    type uint32;
    units "milliseconds";
    default "35000";
    description
        "CCM hold time.";
}
leaf ccm-p-bits-pri {
    type ccm-priority-type;
    description
        "The priority parameter for Continuity Check Messages (CCMs)
        transmitted by the MEP.";
}
}

grouping y-1731 {
    description
        "Grouping for Y-1731";
    reference
        "ITU-T Y-1731: Operations, administration and maintenance
        (OAM) functions and mechanisms for
        Ethernet-based networks";
}
```

```
list y-1731 {
  key "maid";
  description
    "List of configured Y-1731 instances.";
  leaf maid {
    type string;
    description
      "MAID.";
  }
  leaf mep-id {
    type uint32;
    description
      "Local MEP ID.";
  }
  leaf pm-type {
    type identityref {
      base pm-type;
    }
    default "delay";
    description
      "Performance monitor types.";
  }
  leaf remote-mep-id {
    type uint32;
    description
      "Remote MEP ID.";
  }
  leaf message-period {
    type uint32;
    units "milliseconds";
    default "10000";
    description
      "Defines the interval between OAM messages.";
  }
  leaf measurement-interval {
    type uint32;
    units "seconds";
    description
      "Specifies the measurement interval for statistics.";
  }
  leaf cos {
    type uint32;
    description
      "Identifies the Class of Service.";
  }
  leaf loss-measurement {
    type boolean;
    default "false";
  }
}
```

```
    description
      "Controls whether loss measurement is enabled/disabled.";
  }
  leaf synthetic-loss-measurement {
    type boolean;
    default "false";
    description
      "Indicates whether synthetic loss measurement is enabled.";
  }
  container delay-measurement {
    description
      "Container for delay measurement";
    leaf enable-dm {
      type boolean;
      default "false";
      description
        "Controls whether delay measurement is enabled ('true')
        or disabled ('false').";
    }
    leaf two-way {
      type boolean;
      default "false";
      description
        "Whether delay measurement is two-way ('true') of one-
        way ('false').";
    }
  }
  leaf frame-size {
    type uint32;
    units "bytes";
    description
      "Indicates the frame size.";
  }
  leaf session-type {
    type enumeration {
      enum proactive {
        description
          "Proactive mode.";
      }
      enum on-demand {
        description
          "On-demand mode.";
      }
    }
    default "on-demand";
    description
      "Specifies the session type.";
  }
}
```



```
    }  
  }  
  
  grouping parameters-profile {  
    description  
      "Container for per-service parameters.";  
    leaf local-autonomous-system {  
      type inet:as-number;  
      description  
        "Indicates a local AS Number (ASN).";  
    }  
    leaf svc-mtu {  
      type uint32;  
      description  
        "Layer 2 service MTU.  
        It is also known as the maximum transmission  
        unit or maximum frame size.";  
    }  
    leaf ce-vlan-preservation {  
      type boolean;  
      description  
        "Preserve the CE-VLAN ID from ingress to egress, i.e.,  
        CE-VLAN tag of the egress frame is identical to  
        that of the ingress frame that yielded this egress  
        service frame. If all-to-one bundling within a site  
        is enabled, then preservation applies to all ingress  
        service frames. If all-to-one bundling is disabled,  
        then preservation applies to tagged ingress service  
        frames having CE-VLAN ID 1 through 4094.";  
    }  
    leaf ce-vlan-cos-preservation {  
      type boolean;  
      description  
        "CE VLAN CoS preservation. Priority Code Point (PCP) bits  
        in the CE-VLAN tag of the egress frame are identical to  
        those of the ingress frame that yielded this egress  
        service frame.";  
    }  
    leaf control-word-negotiation {  
      type boolean;  
      description  
        "Controls whether Control-word negotiation is enabled  
        (if set to true) or not (if set to false).";  
      reference  
        "RFC 8077: Pseudowire Setup and Maintenance  
        Using the Label Distribution Protocol (LDP),  
        Section 7";  
    }  
  }
```

```
container mac-policies {
  description
    "Container of MAC policies.";
  container mac-addr-limit {
    description
      "Container of MAC address limit configuration.";
    leaf limit-number {
      type uint16;
      description
        "Maximum number of MAC addresses learned from
         the customer for a single service instance.
         The default value is '2' when this grouping
         is used at the service level";
    }
    leaf time-interval {
      type uint32;
      units "milliseconds";
      description
        "The aging time of the mac address.
         The default value is '300' when this grouping
         is used at the service level";
    }
    leaf action {
      type identityref {
        base mac-action;
      }
      description
        "Specifies the action when the upper limit is
         exceeded: drop the packet, flood the packet,
         or log a warning message (without dropping
         the packet).
         The default value is 'warning' when this
         grouping is used at the service level";
    }
  }
}
container mac-loop-prevention {
  description
    "Container for MAC loop prevention.";
  leaf window {
    type uint32;
    units "seconds";
    description
      "The time interval over which a MAC mobility event
       is detected and checked.
       The default value is '180' when this grouping
       is used at the service level";
  }
  leaf frequency {
```

```
    type uint32;
    description
        "The number of times to detect MAC duplication, where
        a 'duplicate MAC address' situation has occurred
        within the 'window' time interval and the duplicate
        MAC address has been added to a list of duplicate
        MAC addresses.
        The default value is '5' when this grouping is
        called at the service level";
}
leaf retry-timer {
    type uint32;
    units "seconds";
    description
        "The retry timer. When the retry timer expires,
        the duplicate MAC address will be flushed from
        the MAC-VRF.";
}
leaf protection-type {
    type identityref {
        base loop-prevention-type;
    }
    description
        "Protection type.
        The default value is 'trap' when this grouping
        is used at the service level";
}
}
}
container multicast {
    if-feature "vpn-common:multicast";
    description
        "Multicast container.";
    leaf enabled {
        type boolean;
        default "false";
        description
            "Enables multicast.";
    }
    container customer-tree-flavors {
        description
            "Type of trees used by the customer.";
        leaf-list tree-flavor {
            type identityref {
                base vpn-common:multicast-tree-type;
            }
            description
                "Type of multicast tree to be used.";
        }
    }
}
```

```
    }  
  }  
}  
  
grouping bandwidth-parameters {  
  description  
    "A grouping for bandwidth parameters.";  
  leaf cir {  
    type uint64;  
    units "bps";  
    description  
      "Committed Information Rate. The maximum  
       number of bits that a port can receive or  
       send during one-second over an  
       interface.";  
  }  
  leaf cbs {  
    type uint64;  
    units "bytes";  
    description  
      "Committed Burst Size. CBS controls the  
       bursty nature of the traffic. Traffic  
       that does not use the configured CIR  
       accumulates credits until the credits  
       reach the configured CBS.";  
  }  
  leaf eir {  
    type uint64;  
    units "bps";  
    description  
      "Excess Information Rate, i.e., excess  
       frame delivery allowed not subject to  
       SLA. The traffic rate can be limited  
       by EIR.";  
  }  
  leaf ebs {  
    type uint64;  
    units "bytes";  
    description  
      "Excess Burst Size. The bandwidth  
       available for burst traffic from the  
       EBS is subject to the amount of  
       bandwidth that is accumulated during  
       periods when traffic allocated by the  
       EIR policy is not used.";  
  }  
  leaf pir {
```

```
    type uint64;
    units "bps";
    description
        "Peak Information Rate, i.e., maximum
        frame delivery allowed. It is equal
        to or less than sum of CIR and EIR.";
}
leaf pbs {
    type uint64;
    units "bytes";
    description
        "Peak Burst Size.";
}
}

/* Main L2NM Container */

container l2vpn-ntw {
    description
        "Container for the L2NM.";
    container vpn-profiles {
        description
            "Container for VPN profiles.";
        uses vpn-common:vpn-profile-cfg;
    }
    container vpn-services {
        description
            "Container for L2VPN services.";
        list vpn-service {
            key "vpn-id";
            description
                "Container of a VPN service.";
            uses vpn-common:vpn-description;
            leaf parent-service-id {
                type vpn-common:vpn-id;
                description
                    "Pointer to the parent service that
                    triggered the L2NM.";
            }
            leaf vpn-type {
                type identityref {
                    base vpn-common:service-type;
                }
                must "not (derived-from-or-self(current(), "
                    + "'vpn-common:l3vpn'))" {
                    error-message "L3VPN is only applicable in L3NM.";
                }
            }
            description

```

```
        "Service type.";
    }
    leaf vpn-service-topology {
        type identityref {
            base vpn-common:vpn-topology;
        }
        description
            "Defining service topology, such as
             any-to-any, hub-spoke, etc.";
    }
    leaf bgp-ad-enabled {
        type boolean;
        description
            "Indicates whether BGP auto-discovery is enabled
             or disabled.";
    }
    leaf signaling-type {
        type identityref {
            base vpn-common:vpn-signaling-type;
        }
        description
            "VPN signaling type.";
    }
    container global-parameters-profiles {
        description
            "Container for a list of global parameters
             profiles.";
        list global-parameters-profile {
            key "profile-id";
            description
                "List of global parameters profiles.";
            leaf profile-id {
                type string;
                description
                    "The identifier of the global parameters profile.";
            }
            uses vpn-common:route-distinguisher;
            uses vpn-common:vpn-route-targets;
            uses parameters-profile;
        }
    }
    container underlay-transport {
        description
            "Container for the underlay transport.";
        uses vpn-common:underlay-transport;
    }
    uses vpn-common:service-status;
    container vpn-nodes {
```

```
description
  "Set of VPN nodes that are involved in the L2NM.";
list vpn-node {
  key "vpn-node-id";
  description
    "Container of the VPN nodes.";
  leaf vpn-node-id {
    type vpn-common:vpn-id;
    description
      "Sets the identifier of the VPN node.";
  }
  leaf description {
    type string;
    description
      "Textual description of a VPN node.";
  }
  leaf ne-id {
    type string;
    description
      "An identifier of the network element where
       the VPN node is deployed. This identifier
       uniquely identifies the network element within
       an administrative domain.";
  }
  leaf role {
    type identityref {
      base vpn-common:role;
    }
    default "vpn-common:any-to-any-role";
    description
      "Role of the VPN node in the VPN.";
  }
  leaf router-id {
    type rt-types:router-id;
    description
      "A 32-bit number in the dotted-quad format that is
       used to uniquely identify a node within an
       autonomous system (AS).";
  }
  container active-global-parameters-profiles {
    description
      "Container for a list of global parameters
       profiles.";
    list global-parameters-profile {
      key "profile-id";
      description
        "List of active global parameters profiles.";
      leaf profile-id {
```

```
    type leafref {
      path "../../../global-parameters-profiles"
        + "/global-parameters-profile/profile-id";
    }
    description
      "Points to a global profile defined at the
       service level.";
  }
  uses parameters-profile;
}
}
uses vpn-common:service-status;
container bgp-auto-discovery {
  when "../../../bgp-ad-enabled = 'true'" {
    description
      "Only applies when BGP auto-discovery is enabled.";
  }
  description
    "BGP is used for auto-discovery.";
  choice bgp-type {
    description
      "Choice for the BGP type.";
    case l2vpn-bgp {
      description
        "Container for BGP L2VPN.";
      leaf vpn-id {
        type vpn-common:vpn-id;
        description
          "VPN Identifier. This identifier serves to
           unify components of a given VPN for the
           sake of auto-discovery.";
        reference
          "RFC 6624: Layer 2 Virtual Private Networks
           Using BGP for Auto-Discovery and
           Signaling";
      }
    }
  }
  case evpn-bgp {
    description
      "EVPN case.";
    leaf evpn-type {
      type leafref {
        path "../../../vpn-type";
      }
      description
        "EVPN type.";
    }
    leaf auto-rt-enable {
```



```
        type boolean;
        default "false";
        description
            "Enables/disabled RT auto-derivation based on
            the ASN and Ethernet Tag ID.";
        reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN,
            Section 7.10.1";
    }
    leaf auto-route-target {
        when "../auto-rt-enable = 'true'" {
            description
                "Can only be used when auto-RD is enabled.";
        }
        type rt-types:route-target;
        config false;
        description
            "The value of the auto-assigned RT.";
    }
}
uses vpn-common:route-distinguisher;
uses vpn-common:vpn-route-targets;
}
container signaling-option {
    description
        "Container for the L2VPN signaling.";
    leaf advertise-mtu {
        type boolean;
        description
            "Controls whether MTU is advertised.";
        reference
            "RFC 4667: Layer 2 Virtual Private Network (L2VPN)
            Extensions for Layer 2 Tunneling
            Protocol (L2TP), Section 4.3";
    }
    leaf mtu-allow-mismatch {
        type boolean;
        description
            "When set to true, it allows MTU mismatch.";
        reference
            "RFC 4667: Layer 2 Virtual Private Network (L2VPN)
            Extensions for Layer 2 Tunneling
            Protocol (L2TP), Section 4.3";
    }
    leaf signaling-type {
        type leafref {
            path "../.../.../signaling-type";
        }
    }
}
```

```
    }
    description
      "VPN signaling type.";
  }
  choice signaling-option {
    description
      "Choice for the signaling-option.";
    case bgp {
      description
        "BGP is used as the signaling protocol.";
      choice bgp-type {
        description
          "Choice for the BGP type.";
        case l2vpn-bgp {
          description
            "Container for BGP L2VPN.";
          leaf ce-range {
            type uint16;
            description
              "Determines the number of remote CEs with
               which a given CE can communicate in the
               context of a VPN.";
            reference
              "RFC 6624: Layer 2 Virtual Private Networks
               Using BGP for Auto-Discovery and
               Signaling";
          }
          leaf pw-encapsulation-type {
            type identityref {
              base iana-bgp-l2-encaps:bgp-l2-encaps-type;
            }
            description
              "PW encapsulation type.";
          }
        }
        container vpls-instance {
          when "derived-from-or-self(..../..../..../"
            + "vpn-type, 'vpn-common:vpls')" {
            description
              "Only applies for VPLS.";
          }
          description
            "VPLS instance.";
          leaf vpls-edge-id {
            type uint16;
            description
              "VPLS Edge Identifier (VE ID). This is
               used when the same VE ID is configured
               for the PE.";
          }
        }
      }
    }
  }
}
```

```
        reference
        "RFC 4761: Virtual Private LAN Service
        (VPLS) Using BGP for Auto-
        Discovery and Signaling,
        Section 3.5";
    }
    leaf vpls-edge-id-range {
        type uint16;
        description
            "Specifies the size of the range of
            VE ID in a VPLS service. The range
            controls the size of the label
            block advertised in the context of
            a VPLS instance.";
        reference
        "RFC 4761: Virtual Private LAN Service
        (VPLS) Using BGP for Auto-
        Discovery and Signaling";
    }
}
case evpn-bgp {
    description
        "Used for EVPN.";
    leaf evpn-type {
        type leafref {
            path "../..//bgp-auto-discovery/evpn-type";
        }
        description
            "EVPN type.";
    }
    leaf service-interface-type {
        type identityref {
            base evpn-service-interface-type;
        }
        description
            "EVPN service interface type.";
    }
}
container evpn-policies {
    description
        "Includes a set of EVPN policies such
        as those related to handling MAC
        addresses.";
    leaf mac-learning-mode {
        type identityref {
            base mac-learning-mode;
        }
        description
```

```
        "Indicates through which plane MAC
        addresses are advertised.";
    }
    leaf ingress-replication {
        type boolean;
        description
            "Controls whether ingress replication is
            enabled/disabled.";
        reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN,
            Section 8.3.1.1";
    }
    leaf p2mp-replication {
        type boolean;
        description
            "Controls whether P2MP replication is
            enabled/disabled.";
        reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN,
            Section 8.3.1.2";
    }
    container arp-proxy {
        if-feature "vpn-common:ipv4";
        description
            "Top container for the ARP proxy.";
        leaf enable {
            type boolean;
            default "false";
            description
                "Enables (when set to 'true') or
                disables (when set to 'false')
                ARP proxy.";
            reference
                "RFC 7432: BGP MPLS-Based Ethernet VPN,
                Section 10";
        }
        leaf arp-suppression {
            type boolean;
            default "false";
            description
                "Enables (when set to 'true') or
                disables (when set to 'false') ARP
                suppression.";
            reference
                "RFC 7432: BGP MPLS-Based Ethernet
                VPN";
        }
        leaf ip-mobility-threshold {
```

```
type uint16;
description
    "It is possible for a given host (as
    defined by its IP address) to move
    from one ES to another.
    IP mobility threshold specifies the
    number of IP mobility events
    that are detected for a given IP
    address within the
    detection-threshold before it
    is identified as a duplicate IP
    address.
    Once the detection threshold is
    reached, updates for the IP address
    are suppressed.";
}
leaf duplicate-ip-detection-interval {
    type uint16;
    units "seconds";
    description
        "The time interval used in detecting a
        duplicate IP address. Duplicate IP
        address detection number of host moves
        are allowed within this interval
        period.";
}
}
container nd-proxy {
    if-feature "vpn-common:ipv6";
    description
        "Top container for the ND proxy.";
    leaf enable {
        type boolean;
        default "false";
        description
            "Enables (when set to 'true') or
            disables (when set to 'false') ND
            proxy.";
        reference
            "RFC 7432: BGP MPLS-Based Ethernet VPN,
            Section 10";
    }
    leaf nd-suppression {
        type boolean;
        default "false";
        description
            "Enables (when set to 'true') or
            disables (when set to 'false')";
    }
}
```

```
Neighbor Discovery (ND) message
suppression.
ND suppression is a technique that
is used to reduce the amount of ND
packets flooding within individual
segments, that is between hosts
connected to the same logical
switch.";
}
leaf ip-mobility-threshold {
  type uint16;
  description
    "It is possible for a given host (as
    defined by its IP address) to move
    from one ES to another.
    IP mobility threshold specifies the
    number of IP mobility events
    that are detected for a given IP
    address within the
    detection-threshold before it
    is identified as a duplicate IP
    address.
    Once the detection threshold is
    reached, updates for the IP address
    are suppressed.";
}
leaf duplicate-ip-detection-interval {
  type uint16;
  units "seconds";
  description
    "The time interval used in detecting a
    duplicate IP address. Duplicate IP
    address detection number of host moves
    are allowed within this interval
    period.";
}
}
leaf underlay-multicast {
  type boolean;
  default "false";
  description
    "Enables (when set to 'true') or disables
    (when set to 'false') underlay
    multicast.";
}
leaf flood-unknown-unicast-supression {
  type boolean;
  default "false";
```

```
description
  "Enables (when set to 'true') or disables
   (when set to 'false') unknown flood
   unicast suppression.";
}
leaf vpws-vlan-aware {
  type boolean;
  default "false";
  description
    "Enables (when set to 'true') or disables
     (when set to 'false') VPWS VLAN-aware.";
}
container bum-management {
  description
    "Broadcast-unknown-unicast-multicast
     management.";
  leaf discard-broadcast {
    type boolean;
    default "false";
    description
      "Discards broadcast, when enabled.";
  }
  leaf discard-unknown-multicast {
    type boolean;
    default "false";
    description
      "Discards unknown multicast, when
       enabled.";
  }
  leaf discard-unknown-unicast {
    type boolean;
    default "false";
    description
      "Discards unknown unicast, when
       enabled.";
  }
}
container pbb {
  when "derived-from-or-self("
    + "../..//evpn-type, 'pbb-evpn') " {
    description
      "Only applies for PBB EVPN.";
  }
  description
    "PBB parameters container.";
  reference
    "IEEE 802.1ah: Provider Backbone Bridge";
  leaf backbone-src-mac {
```

```
        type yang:mac-address;
        description
            "Includes provider backbone MAC (B-MAC)
            address.";
        reference
            "RFC 7623: Provider Backbone Bridging
            Combined with Ethernet VPN
            (PBB-EVPN), Section 8.1";
    }
}
}
}
}
}
}
container ldp-or-l2tp {
    description
        "Container for LDP or L2TP-signaled PWs.";
    leaf agi {
        type rt-types:route-distinguisher;
        description
            "Attachment Group Identifier. Also, called
            VPLS-Id.";
        reference
            "RFC 4667: Layer 2 Virtual Private Network
            (L2VPN) Extensions for Layer 2
            Tunneling Protocol (L2TP),
            Section 4.3
            RFC 4762: Virtual Private LAN Service (VPLS)
            Using Label Distribution Protocol
            (LDP) Signaling, Section 6.1.1";
    }
    leaf saii {
        type uint32;
        description
            "Source Attachment Individual Identifier
            (SAII).";
        reference
            "RFC 4667: Layer 2 Virtual Private Network
            (L2VPN) Extensions for Layer 2
            Tunneling Protocol (L2TP),
            Section 3";
    }
    list remote-targets {
        key "taii";
        description
            "List of allowed target Attachment Individual
            Identifier (AII) and peers.";
        reference
```



```
        "RFC 4667: Layer 2 Virtual Private Network
          (L2VPN) Extensions for Layer 2
          Tunneling Protocol (L2TP),
          Section 5";
    leaf taii {
        type uint32;
        description
            "Target Attachment Individual Identifier.";
        reference
            "RFC 4667: Layer 2 Virtual Private Network
              (L2VPN) Extensions for Layer 2
              Tunneling Protocol (L2TP),
              Section 3";
    }
    leaf peer-addr {
        type inet:ip-address;
        description
            "Indicates the peer forwarder's IP address.";
    }
}
choice ldp-or-l2tp {
    description
        "Choice of LDP or L2TP-signaled PWs.";
    case ldp {
        description
            "Container for T-LDP PW configurations.";
        leaf t-ldp-pw-type {
            type identityref {
                base t-ldp-pw-type;
            }
            description
                "T-LDP PW type.";
        }
        leaf pw-type {
            type identityref {
                base ldp-pw-type;
            }
            description
                "PW encapsulation type.";
            reference
                "RFC 4762: Virtual Private LAN Service
                  (VPLS) Using Label Distribution
                  Protocol (LDP) Signaling,
                  Section 6.1.1";
        }
        leaf pw-description {
            type string;
            description
```

```
        "Includes a human-readable description
        of the interface. This may be used when
        communicating with a remote peer.";
    reference
        "RFC 4762: Virtual Private LAN Service
        (VPLS) Using Label Distribution
        Protocol (LDP) Signaling,
        Section 6.1.1";
}
leaf mac-addr-withdraw {
    type boolean;
    description
        "If set to 'true', then MAC address
        withdrawal is enabled. If 'false',
        then MAC address withdrawal is
        disabled.";
    reference
        "RFC 4762: Virtual Private LAN Service
        (VPLS) Using Label Distribution
        Protocol (LDP) Signaling,
        Section 6.2";
}
list pw-peer-list {
    key "peer-addr vc-id";
    description
        "List of AC and PW bindings.";
    leaf peer-addr {
        type inet:ip-address;
        description
            "Indicates the peer's IP address.";
    }
    leaf vc-id {
        type string;
        description
            "VC label used to identify a PW.";
    }
    leaf pw-priority {
        type uint32;
        description
            "Defines the priority for the PW.
            The higher the pw-priority value, the
            higher the preference of the PW will
            be.";
    }
}
}
container qinq {
    when "derived-from-or-self("
        + "../t-ldp-pw-type, 'hvpls')\" {
```

```
        description
        "Only applies when t-ldp pw type
        is h-vpls.";
    }
    description
    "Container for QinQ.";
    leaf s-tag {
        type dot1q-types:vlanid;
        mandatory true;
        description
        "S-TAG.";
    }
    leaf c-tag {
        type dot1q-types:vlanid;
        mandatory true;
        description
        "C-TAG.";
    }
}
}
case l2tp {
    description
    "Container for L2TP PWs.";
    leaf router-id {
        type rt-types:router-id;
        description
        "A 32-bit number in the dotted-quad format
        that is used to uniquely identify a node
        within a service provider network.";
        reference
        "RFC 4667: Layer 2 Virtual Private Network
        (L2VPN) Extensions for Layer 2
        Tunneling Protocol (L2TP),
        Section 4.2";
    }
    leaf pseudowire-type {
        type identityref {
            base iana-pw-types:iana-pw-types;
        }
        description
        "Encapsulation type.";
        reference
        "RFC 4667: Layer 2 Virtual Private Network
        (L2VPN) Extensions for Layer 2
        Tunneling Protocol (L2TP),
        Section 4.2";
    }
}
}
```

```
    }
  }
}
container vpn-network-accesses {
  description
    "Main container for VPN network accesses.";
  list vpn-network-access {
    key "id";
    description
      "List of VPN network accesses.";
    leaf id {
      type vpn-common:vpn-id;
      description
        "Identifier of the network access";
    }
    leaf description {
      type string;
      description
        "A textual description of the VPN network
        access.";
    }
    leaf interface-id {
      type string;
      description
        "Refers to a physical or logical interface.";
    }
    leaf active-vpn-node-profile {
      type leafref {
        path "../../"
          + "/active-global-parameters-profiles"
          + "/global-parameters-profile/profile-id";
      }
      description
        "An identifier of an active VPN instance
        profile.";
    }
  }
  uses vpn-common:service-status;
  container connection {
    description
      "Container for the bearer and AC.";
    leaf l2-termination-point {
      type string;
      description
        "Specifies a reference to a local Layer 2
        termination point such as a Layer 2
        sub-interface.";
    }
  }
}
```

```
leaf local-bridge-reference {
  type string;
  description
    "Specifies a local bridge reference to
    accommodate, for example, implementations
    that require internal bridging.
    A reference may be a local bridge domain.";
}
leaf bearer-reference {
  if-feature "vpn-common:bearer-reference";
  type string;
  description
    "This is an internal reference for the service
    provider to identify the bearer associated
    with this VPN.";
}
container encapsulation {
  description
    "Container for Layer 2 encapsulation.";
  leaf encap-type {
    type identityref {
      base vpn-common:encapsulation-type;
    }
    default "vpn-common:priority-tagged";
    description
      "Tagged interface type. By default, the
      type of the tagged interface is
      'priority-tagged'.";
  }
  container dot1q {
    when "derived-from-or-self(../encap-type, "
      + "'vpn-common:dot1q') " {
      description
        "Only applies when the type of the
        tagged interface is 'dot1q'.";
    }
    description
      "Tagged interface.";
    leaf tag-type {
      type identityref {
        base vpn-common:tag-type;
      }
      default "vpn-common:c-vlan";
      description
        "Tag type. By default, the tag type is
        'c-vlan'.";
    }
  }
  leaf cvlan-id {
```

```
type dot1q-types:vlanid;
description
  "VLAN identifier.";
}
container tag-operations {
  description
    "Sets the tag manipulation policy for this
    VPN network access. It defines a set of
    tag manipulations that allow for the
    insertion, removal, or rewriting
    of 802.1Q VLAN tags. These operations are
    indicated for the CE-PE direction.
    By default, tag operations are symmetric.
    As such, the reverse tag operation is
    assumed on the PE-CE direction.";
  choice op-choice {
    description
      "Selects the tag rewriting policy for a
      VPN network access.";
    leaf pop {
      type empty;
      description
        "Pop the outer tag.";
    }
    leaf push {
      type empty;
      description
        "Push one or two tags defined by the
        tag-1 and tag-2 leaves. It is
        assumed that, absent any policy, the
        default value of 0 will be used for
        PCP setting.";
    }
    leaf translate {
      type empty;
      description
        "Translate the outer tag to one or two
        tags. PCP bits are preserved.";
    }
  }
}
leaf tag-1 {
  when 'not(..pop)';
  type dot1q-types:vlanid;
  description
    "A first tag to be used for push or
    translate operations. This tag will be
    used as the outermost tag as a result
    of the tag operation.";
```

```
    }
    leaf tag-1-type {
      type dot1q-types:dot1q-tag-type;
      default "dot1q-types:s-vlan";
      description
        "Specifies a specific 802.1Q tag type
         of tag-1.";
    }
    leaf tag-2 {
      when '../translate';
      type dot1q-types:vlanid;
      description
        "A second tag to be used for
         translation.";
    }
    leaf tag-2-type {
      type dot1q-types:dot1q-tag-type;
      default "dot1q-types:c-vlan";
      description
        "Specifies a specific 802.1Q tag type
         of tag-2.";
    }
  }
}
container priority-tagged {
  when "derived-from-or-self(../encap-type, "
    + "'vpn-common:priority-tagged') " {
    description
      "Only applies when the type of the
       tagged interface is 'priority-tagged'.";
  }
  description
    "Priority tagged container.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:c-vlan";
    description
      "Tag type. By default, the tag type is
       'c-vlan'.";
  }
}
container qinq {
  when "derived-from-or-self(../encap-type, "
    + "'vpn-common:qinq') " {
    description
      "Only applies when the type of the tagged
```

```
        interface is QinQ.";
    }
    description
        "Includes QinQ parameters.";
    leaf tag-type {
        type identityref {
            base vpn-common:tag-type;
        }
        default "vpn-common:s-c-vlan";
        description
            "Tag type. By default, the tag type is
             's-c-vlan'.";
    }
    leaf svlan-id {
        type dot1q-types:vlanid;
        mandatory true;
        description
            "S-VLAN identifier.";
    }
    leaf cvlan-id {
        type dot1q-types:vlanid;
        mandatory true;
        description
            "C-VLAN identifier.";
    }
    container tag-operations {
        description
            "Sets the tag manipulation policy for this
             VPN network access. It defines a set of
             tag manipulations that allow for the
             insertion, removal, or rewriting
             of 802.1Q VLAN tags. These operations are
             indicated for the CE-PE direction.
             By default, tag operations are symmetric.
             As such, the reverse tag operation is
             assumed on the PE-CE direction.";
        choice op-choice {
            description
                "Selects the tag rewriting policy for a
                 VPN network access.";
            leaf pop {
                type uint8 {
                    range "1|2";
                }
                description
                    "Pop one or two tags as a function
                     of the indicated pop value.";
            }
        }
    }
}
```



```
leaf push {
  type empty;
  description
    "Push one or two tags defined by the
    tag-1 and tag-2 leaves. It is
    assumed that, absent any policy, the
    default value of 0 will be used for
    PCP setting.";
}
leaf translate {
  type uint8 {
    range "1|2";
  }
  description
    "Translate one or two outer tags. PCP
    bits are preserved.

    The following operations are
    supported:

    - translate 1 with tag-1 leaf is
      provided: only the outermost tag is
      translated to the value in tag-1.

    - translate 2 with both tag-1 and
      tag-2 leaves are provided: both
      outer and inner tags are translated
      to the values in tag-1 and tag-2,
      respectively.

    - translate 2 with tag-1 leaf is
      provided: the outer tag is popped
      while the inner tag is translated
      to the value in tag-1.";
}
}
leaf tag-1 {
  when 'not(..../pop)';
  type dot1q-types:vlanid;
  description
    "A first tag to be used for push or
    translate operations. This tag will be
    used as the outermost tag as a result
    of the tag operation.";
}
leaf tag-1-type {
  type dot1q-types:dot1q-tag-type;
  default "dot1q-types:s-vlan";
}
```

```
        description
        "Specifies a specific 802.1Q tag type
        of tag-1.";
    }
    leaf tag-2 {
        when 'not(..pop)';
        type dot1q-types:vlanid;
        description
        "A second tag to be used for push or
        translate operations.";
    }
    leaf tag-2-type {
        type dot1q-types:dot1q-tag-type;
        default "dot1q-types:c-vlan";
        description
        "Specifies a specific 802.1Q tag type
        of tag-2.";
    }
}
}
}
container lag-interface {
    if-feature "vpn-common:lag-interface";
    description
    "Container of LAG interface attributes
    configuration.";
    leaf lag-interface-id {
        type string;
        description
        "LAG interface identifier.";
    }
}
container lacp {
    description
    "Container for LACP.";
    leaf lacp-state {
        type boolean;
        default "false";
        description
        "Controls whether LACP is enabled.";
    }
    leaf mode {
        type identityref {
            base lacp-mode;
        }
        description
        "Indicates the LACP mode.";
    }
    leaf speed {
```

```
    type uint32;
    units "mbps";
    default "10";
    description
        "LACP speed. This low default value
         is inherited from the L2SM.";
}
leaf mini-link-num {
    type uint32;
    description
        "Defines the minimum number of links that
         must be active before the aggregating
         link is put into service.";
}
leaf system-id {
    type yang:mac-address;
    description
        "Indicates the System ID used by LACP.";
}
leaf admin-key {
    type uint16;
    description
        "Indicates the value of the key used for
         the aggregate interface.";
}
leaf system-priority {
    type uint16 {
        range "0..65535";
    }
    default "32768";
    description
        "Indicates the LACP priority for the
         system.";
}
container member-link-list {
    description
        "Container of Member link list.";
    list member-link {
        key "name";
        description
            "Member link.";
        leaf name {
            type string;
            description
                "Member link name.";
        }
        leaf speed {
            type uint32;
        }
    }
}
```

```
        units "mbps";
        default "10";
        description
            "Port speed.";
    }
    leaf mode {
        type identityref {
            base vpn-common:neg-mode;
        }
        description
            "Negotiation mode.";
    }
    leaf link-mtu {
        type uint32;
        description
            "Link MTU size.";
    }
    container oam-802.3ah-link {
        if-feature "oam-3ah";
        description
            "Container for oam 802.3ah link.";
        leaf enable {
            type boolean;
            default "false";
            description
                "Indicates support of OAM 802.3ah
                 link.";
        }
    }
}

leaf flow-control {
    type boolean;
    default "false";
    description
        "Indicates whether flow control is
         supported.";
}

leaf lldp {
    type boolean;
    default "false";
    description
        "Indicates whether Link Layer Discovery
         Protocol (LLDP) is supported.";
}

container split-horizon {
    description
```

```
        "Configuration with split horizon enabled.";
    leaf group-name {
        type string;
        description
            "Group name of the Split Horizon.";
    }
}
}
choice signaling-option {
    description
        "Choice for the signaling-option.";
    case bgp {
        description
            "BGP is used as the signaling protocol.";
        choice bgp-type {
            description
                "Choice for the BGP type.";
            case l2vpn-bgp {
                description
                    "Container for BGP L2VPN.";
                leaf ce-id {
                    type uint16;
                    description
                        "Identifies the CE within the VPN.";
                    reference
                        "RFC 6624: Layer 2 Virtual Private
                        Networks Using BGP for
                        Auto-Discovery and
                        Signaling";
                }
                leaf remote-ce-id {
                    type uint16;
                    description
                        "Indicates the identifier of the remote
                        CE.";
                }
            }
            case vpls {
                description
                    "VPLS instance."
                    when "derived-from-or-self ../../../../.."
                        + "vpn-type, 'vpn-common:vpls'" {
                        description
                            "Only applies for VPLS.";
                    }
                leaf vpls-edge-id {
                    type uint16;
                    description

```

```
        "VPLS Edge Identifier (VE ID).";
    reference
        "RFC 4761: Virtual Private LAN Service
        (VPLS) Using BGP for Auto-
        Discovery and Signaling,
        Section 3.2.1";
    }
}
}
case evpn-bgp {
    description
        "Used for EVPN.";
    leaf df-preference {
        type uint16;
        default "32767";
        description
            "Defines a 2-octet value that indicates
            the PE preference to become the DF in
            the ES.

            The preference value is only applicable
            to the preference based method.";
        reference
            "RFC 8584: Framework for Ethernet VPN
            Designated Forwarder Election
            Extensibility";
    }
    container vpws-service-instance {
        when "derived-from-or-self ../../../../"
            + "vpn-type, 'vpn-common:vpws-evpn'" {
            description
                "Only applies for EVPN-VPWS.";
        }
        description
            "Local and remote VPWS Service Instance
            (VSI)";
        reference
            "RFC 8214: Virtual Private Wire Service
            Support in Ethernet VPN";
        choice local-vsi-choice {
            description
                "Choices for assigning local VSI.";
            case directly-assigned {
                description
                    "Explicitly assign a local VSI.";
                leaf local-vpws-service-instance {
                    type uint32 {
                        range "1..16777215";
                    }
                }
            }
        }
    }
}
```

```
    }
    description
      "Indicates the assigned local
       VSI.";
  }
}
case auto-assigned {
  description
    "The local VSI is auto-assigned.";
  container local-vsi-auto {
    description
      "The local VSI is auto-assigned.";
    choice auto-mode {
      description
        "Indicates the auto-assignment
         mode of local VSI. VSI can be
         automatically assigned either
         with or without indicating a
         pool from which the VSI
         should be taken.

         For both cases, the server
         will auto-assign a local VSI
         value and use that value.";
      case from-pool {
        leaf vsi-pool-name {
          type string;
          description
            "The auto-assignment will be
             made from this pool.";
        }
      }
    }
    case full-auto {
      leaf auto {
        type empty;
        description
          "Indicates that a local VSI
           is fully auto-assigned.";
      }
    }
  }
}
leaf auto-local-vsi {
  type uint32 {
    range "1..16777215";
  }
  config false;
  description
    "The value of the auto-assigned
```

```
        local VSI.";
    }
}
}
choice remote-vsi-choice {
    description
        "Choice for assigning the remote VSI.";
    case directly-assigned {
        description
            "Explicitly assign a remote VSI.";
        leaf remote-vpws-service-instance {
            type uint32 {
                range "1..16777215";
            }
            description
                "Indicates the value of the remote
                VSI.";
        }
    }
    case auto-assigned {
        description
            "The remote VSI is auto-assigned.";
        container remote-vsi-auto {
            description
                "The remote VSI is auto-assigned.";
            choice auto-mode {
                description
                    "Indicates the auto-assignment
                    mode of remote VSI. VSI can be
                    automatically assigned either
                    with or without indicating a
                    pool from which the VSI
                    should be taken.

                    For both cases, the server
                    will auto-assign a remote VSI
                    value and use that value.";
                case from-pool {
                    leaf vsi-pool-name {
                        type string;
                        description
                            "The auto-assignment will be
                            made from this pool.";
                    }
                }
                case full-auto {
                    leaf auto {

```



```
        type empty;
        description
            "Indicates that a remote VSI
             is fully auto-assigned.";
    }
}
}
leaf auto-remote-vsi {
    type uint32 {
        range "1..16777215";
    }
    config false;
    description
        "The value of the auto-assigned
         remote VSI.";
}
}
}
}
}
}
}
}
}
list group {
    key "group-id";
    description
        "List of group-ids.";
    leaf group-id {
        type string;
        description
            "Indicates the group-id to which the network
             access belongs to.";
    }
    leaf precedence {
        type identityref {
            base precedence-type;
        }
        description
            "Defining service redundancy in transport
             network.";
    }
    leaf ethernet-segment-identifier {
        type string;
        description
            "Reference to the ESI associated with the VPN
             network access.";
    }
}
```

```
}
container ethernet-service-oam {
  description
    "Container for Ethernet service OAM.";
  leaf md-name {
    type string;
    description
      "Maintenance domain name.";
  }
  leaf md-level {
    type uint8;
    description
      "Maintenance domain level.";
  }
  container cfm-802.1-ag {
    description
      "Container of 802.1ag CFM configurations.";
    list n2-uni-c {
      key "maid";
      description
        "List of UNI-N to UNI-C.";
      uses cfm-802;
    }
    list n2-uni-n {
      key "maid";
      description
        "List of UNI-N to UNI-N.";
      uses cfm-802;
    }
  }
  uses y-1731;
}
container service {
  description
    "Container for service";
  leaf mtu {
    type uint32;
    description
      "Layer 2 MTU, it is also known as the maximum
        transmission unit or maximum frame size.";
  }
  container svc-pe-to-ce-bandwidth {
    if-feature "vpn-common:inbound-bw";
    description
      "From the customer site's perspective, the
        service inbound bandwidth of the connection
        or download bandwidth from the service
        provider the site. Note that the L2SM uses
```

```
        'input-bandwidth' to refer to the same
        concept.";
list pe-to-ce-bandwidth {
  key "bw-type";
  description
    "List for PE-to-CE bandwidth data nodes.";
  leaf bw-type {
    type identityref {
      base vpn-common:bw-type;
    }
    description
      "Indicates the bandwidth type.";
  }
  choice type {
    description
      "Choice based upon bandwidth type.";
    case per-cos {
      description
        "Bandwidth per CoS.";
      list cos {
        key "cos-id";
        description
          "List of class of services.";
        leaf cos-id {
          type uint8;
          description
            "Identifier of the CoS, indicated by
            DSCP or a CE-CLAN CoS (802.1p) value
            in the service frame.";
          reference
            "IEEE Std 802.1Q: Bridges and Bridged
            Networks";
        }
        uses bandwidth-parameters;
      }
    }
    case other {
      description
        "Other bandwidth types.";
      uses bandwidth-parameters;
    }
  }
}
}
container svc-ce-to-pe-bandwidth {
  if-feature "vpn-common:outbound-bw";
  description
    "From the customer site's perspective,
```

```
the service outbound bandwidth of the
connection or upload bandwidth from
the CE to the PE. Note that the L2SM uses
'output-bandwidth' to refer to the same
concept.";
list ce-to-pe-bandwidth {
  key "bw-type";
  description
    "List for CE-to-PE bandwidth.";
  leaf bw-type {
    type identityref {
      base vpn-common:bw-type;
    }
    description
      "Indicates the bandwidth type.";
  }
  choice type {
    description
      "Choice based upon bandwidth type.";
    case per-cos {
      description
        "Bandwidth per CoS.";
      list cos {
        key "cos-id";
        description
          "List of class of services.";
        leaf cos-id {
          type uint8;
          description
            "Identifier of the CoS, indicated by
            DSCP or a CE-CLAN CoS (802.1p) value
            in the service frame.";
          reference
            "IEEE Std 802.1Q: Bridges and Bridged
            Networks";
        }
        uses bandwidth-parameters;
      }
    }
    case other {
      description
        "Other non CoS-aware bandwidth types.";
      uses bandwidth-parameters;
    }
  }
}
}
container qos {
```

```
if-feature "vpn-common:qos";
description
  "QoS configuration.";
container qos-classification-policy {
  description
    "Configuration of the traffic classification
    policy.";
  list rule {
    key "id";
    ordered-by user;
    description
      "List of classification rules.";
    leaf id {
      type string;
      description
        "A description identifying the QoS
        classification policy rule.";
    }
    choice match-type {
      default "match-flow";
      description
        "Choice for classification.";
      case match-flow {
        container match-flow {
          description
            "Describes flow-matching criteria.";
          leaf dscp {
            type inet:dscp;
            description
              "DSCP value.";
          }
          leaf dot1q {
            type uint16;
            description
              "802.1Q matching. It is a VLAN tag
              added into a frame.";
            reference
              "IEEE Std 802.1Q: Bridges and
              Bridged
              Networks";
          }
          leaf pcps {
            type uint8 {
              range "0..7";
            }
            description
              "Priority Code Point (PCP) value.";
          }
        }
      }
    }
  }
}
```

```
    leaf src-mac-address {
      type yang:mac-address;
      description
        "Source MAC address.";
    }
    leaf dst-mac-address {
      type yang:mac-address;
      description
        "Destination MAC address.";
    }
    leaf color-type {
      type identityref {
        base color-type;
      }
      description
        "Color type.";
    }
    leaf any {
      type empty;
      description
        "Allows all.";
    }
  }
}
case match-application {
  leaf match-application {
    type identityref {
      base vpn-common:customer-application;
    }
    description
      "Defines the application to match.";
  }
}
}
leaf target-class-id {
  type string;
  description
    "Identification of the CoS.
    This identifier is internal to the
    administration.";
}
}
}
container qos-profile {
  description
    "QoS profile configuration.";
  list qos-profile {
    key "profile";
```

```
    description
      "QoS profile.
      Can be standard profile or customized
      profile.";
    leaf profile {
      type leafref {
        path "/l2vpn-ntw/vpn-profiles"
          + "/valid-provider-identifiers"
          + "/qos-profile-identifier/id";
      }
      description
        "QoS profile to be used.";
    }
    leaf direction {
      type identityref {
        base vpn-common:qos-profile-direction;
      }
      default "vpn-common:both";
      description
        "The direction to which the QoS profile
        is applied.";
    }
  }
}
}
container mac-policies {
  description
    "Container for MAC-related policies.";
  list access-control-list {
    key "name";
    description
      "Container for access control List.";
    leaf name {
      type string;
      description
        "Specifies the name of the ACL.";
    }
    leaf-list src-mac-address {
      type yang:mac-address;
      description
        "Specifies the source MAC address.";
    }
    leaf-list src-mac-address-mask {
      type yang:mac-address;
      description
        "Specifies the source MAC address mask.";
    }
    leaf-list dst-mac-address {
```

```
    type yang:mac-address;
    description
      "Specifies the destination MAC address.";
  }
  leaf-list dst-mac-address-mask {
    type yang:mac-address;
    description
      "Specifies the destination MAC address
      mask.";
  }
  leaf action {
    type identityref {
      base mac-action;
    }
    default "drop";
    description
      "Specifies the filtering action.";
  }
  leaf rate-limit {
    when "derived-from-or-self(.. / action, "
      + "'flood') " {
      description
        "Rate-limit is valid only when the action
        is to accept the matching frame.";
    }
    type decimal64 {
      fraction-digits 2;
    }
    units "bytes per second";
    description
      "Specifies how to rate-limit the traffic.";
  }
}
container mac-loop-prevention {
  description
    "Container of MAC loop prevention.";
  leaf window {
    type uint32;
    units "seconds";
    default "180";
    description
      "The timer when a MAC mobility event is
      detected.";
  }
  leaf frequency {
    type uint32;
    default "5";
    description
```



```
        "The number of times to detect MAC
        duplication, where a 'duplicate MAC
        address' situation has occurred and
        the duplicate MAC address has been
        added to a list of duplicate MAC
        addresses.";
    }
    leaf retry-timer {
        type uint32;
        units "seconds";
        description
            "The retry timer. When the retry timer
            expires, the duplicate MAC address will
            be flushed from the MAC-VRF.";
    }
    leaf protection-type {
        type identityref {
            base loop-prevention-type;
        }
        default "trap";
        description
            "Protection type";
    }
}
container mac-addr-limit {
    description
        "Container of MAC-Addr limit configurations";
    leaf limit-number {
        type uint16;
        default "2";
        description
            "Maximum number of MAC addresses learned
            from the subscriber for a single service
            instance.";
    }
    leaf time-interval {
        type uint32;
        units "milliseconds";
        default "300";
        description
            "The aging time of the mac address.";
    }
    leaf action {
        type identityref {
            base mac-action;
        }
        default "warning";
        description
```

```
        "Specifies the action when the upper limit
        is exceeded: drop the packet, flood the
        packet, or log a warning message (without
        dropping the packet).";
    }
}
}
container broadcast-unknown-unicast-multicast {
  description
    "Container of broadcast, unknown unicast, and
    multicast configurations";
  leaf multicast-site-type {
    type enumeration {
      enum receiver-only {
        description
          "The site only has receivers.";
      }
      enum source-only {
        description
          "The site only has sources.";
      }
      enum source-receiver {
        description
          "The site has both sources and
          receivers.";
      }
    }
    default "source-receiver";
    description
      "Type of the multicast site.";
  }
  list multicast-gp-address-mapping {
    key "id";
    description
      "List of Port to group mappings.";
    leaf id {
      type uint16;
      description
        "Unique identifier for the mapping.";
    }
    leaf vlan-id {
      type uint32;
      mandatory true;
      description
        "The VLAN ID of the multicast group.";
    }
    leaf mac-gp-address {
      type yang:mac-address;
    }
  }
}
```

```

        mandatory true;
        description
            "The MAC address of the multicast group.";
    }
    leaf port-lag-number {
        type uint32;
        description
            "The port/LAG belonging to the multicast
             group.";
    }
}
leaf bum-overall-rate {
    type uint64;
    units "bps";
    description
        "Overall rate for BUM.";
}
}
}
}
}
}
}
}
}
}
<CODE ENDS>

```

## 9. Security Considerations

The YANG modules specified in this document defines schemas for data that are designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in "ietf-l2vpn-ntw" and "ietf-ethernet-segment" YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network

environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability in the "ietf-l2vpn-ntw" and "ietf-ethernet-segment" modules:

- \* 'vpn-profiles': This container includes a set of sensitive data that influence how the L3VPN service is delivered. For example, an attacker who has access to these data nodes may be able to manipulate routing policies, QoS policies, or encryption properties. These data nodes are defined with "nacm:default-deny-write" tagging [RFC9181].
- \* 'ethernet-segments' and 'vpn-services': An attacker who is able to access network nodes can undertake various attacks, such as deleting a running L2VPN service, interrupting all the traffic of a client. In addition, an attacker may modify the attributes of a running service (e.g., QoS, bandwidth) or an ES, leading to malfunctioning of the service and therefore to SLA violations. In addition, an attacker could attempt to create an L2VPN service, add a new network access, or intercept/redirect the traffic to a non-authorized node. In addition to using NACM to prevent unauthorized access, such activity can be detected by adequately monitoring and tracking network configuration changes.

Some of the readable data nodes in the "ietf-l2vpn-ntw" YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- \* 'customer-name' and 'ip-connection': An attacker can retrieve privacy-related information which can be used to track a customer. Disclosing such information may be considered as a violation of the customer-provider trust relationship.

Both "iana-bgp-l2-encaps" and "iana-pseudowire-types" modules define YANG identities for encapsulation/pseudowires types. These identities are intended to be referenced by other YANG modules, and by themselves do not expose any nodes which are writable, contain read-only state, or RPCs.

## 10. IANA Considerations

### 10.1. Registering YANG Modules

This document requests IANA to register the following URIs in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-pseudowire-types  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ethernet-segment  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry:

name: iana-bgp-l2-encaps  
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-l2-encaps  
maintained by IANA: Y  
prefix: iana-bgp-l2-encaps  
reference: RFC XXXX

name: iana-pseudowire-types  
namespace: urn:ietf:params:xml:ns:yang:iana-pseudowire-types  
maintained by IANA: Y  
prefix: iana-pw-types  
reference: RFC XXXX

name: ietf-ethernet-segment  
namespace: urn:ietf:params:xml:ns:yang:ietf-ethernet-segment  
maintained by IANA: N  
prefix: l2vpn-es  
reference: RFC XXXX

name: ietf-l2vpn-ntw  
namespace: urn:ietf:params:xml:ns:yang:ietf-l2vpn-ntw  
maintained by IANA: N  
prefix: l2vpn-ntw  
reference: RFC XXXX

## 10.2. BGP Layer 2 Encapsulation Types

This document defines the initial version of the IANA-maintained "iana-bgp-l2-encaps" YANG module (Section 8.1). IANA is requested to add this note to the registry:

BGP Layer 2 encapsulation types must not be directly added to the "iana-bgp-l2-encaps" YANG module. They must instead be added to the "BGP Layer 2 Encapsulation Types" registry [IANA-BGP-L2].

When a Layer 2 encapsulation type is added to the "BGP Layer 2 Encapsulation Types" registry, a new "identity" statement must be added to the "iana-bgp-l2-encaps" YANG module. The name of the "identity" is a lower-case version of the encapsulation name provided in the description. The "identity" statement should have the following sub-statements defined:

"base": Contains 'bgp-l2-encaps-type'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry with the title of the document added.

Unassigned or reserved values are not present in the module.

When the "iana-bgp-l2-encaps" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to [IANA-BGP-L2]:

When this registry is modified, the YANG module "iana-bgp-l2-encaps" must be updated as defined in RFCXXXX.

## 10.3. Pseudowire Types

This document defines the initial version of the IANA-maintained "iana-pseudowire-types" YANG module (Section 8.2). IANA is requested to add this note to the registry:

MPLS pseudowire types must not be directly added to the "iana-bgp-l2-encaps" YANG module. They must instead be added to the "MPLS Pseudowire Types" registry [IANA-PW-Types].

When a pseudowire type is added to the "iana-pseudowire-types" registry, a new "identity" statement must be added to the "iana-pseudowire-types" YANG module. The name of the "identity" is a

lower-case version of the encapsulation name provided in the description. The "identity" statement should have the following sub-statements defined:

"base": Contains 'iana-pw-types'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry with the title of the document added

Unassigned or reserved values are not present in the module.

When the "iana-pseudowire-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to [IANA-PW-Types]:

When this registry is modified, the YANG module "iana-pseudowire-types" must be updated as defined in RFCXXXX.

## 11. References

### 11.1. Normative References

[IANA-BGP-L2]

IANA, "BGP Layer 2 Encapsulation Types",  
<<https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-l2-encapsulation-types-registry>>.

[IANA-PW-Types]

IANA, "MPLS Pseudowire Types Registry",  
<<http://www.iana.org/assignments/pwe3-parameters/pwe3-parameters.xhtml#pwe3-parameters-2>>.

[IEEE-802-1ag]

IEEE, "802.1ag - 2007 - IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management", 2007, <DOI 10.1109/IEEESTD.2007.4431836>.

[IEEE802.1Qcp-2018]

IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model", September 2018,  
<<https://ieeexplore.ieee.org/document/8467507>>.

- [ITU-T-Y-1731] Union, I. T., "Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet-based networks", August 2015, <<https://www.itu.int/rec/T-REC-Y.1731/en>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4667] Luo, W., "Layer 2 Virtual Private Network (L2VPN) Extensions for Layer 2 Tunneling Protocol (L2TP)", RFC 4667, DOI 10.17487/RFC4667, September 2006, <<https://www.rfc-editor.org/info/rfc4667>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6074] Rosen, E., Davie, B., Radoaca, V., and W. Luo, "Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)", RFC 6074, DOI 10.17487/RFC6074, January 2011, <<https://www.rfc-editor.org/info/rfc6074>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.



- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <<https://www.rfc-editor.org/info/rfc7623>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <<https://www.rfc-editor.org/info/rfc8077>>.
- [RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <<https://www.rfc-editor.org/info/rfc8214>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.

- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.
- [RFC9181] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., and Q. Wu, "A Common YANG Data Model for Layer 2 and Layer 3 VPNs", RFC 9181, DOI 10.17487/RFC9181, February 2022, <<https://www.rfc-editor.org/info/rfc9181>>.

## 11.2. Informative References

- [I-D.ietf-bess-evpn-pref-df]  
Rabadan, J., Sathappan, S., Przygienda, T., Lin, W., Drake, J., Sajassi, A., and S. Mohanty, "Preference-based EVPN DF Election", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-pref-df-08, 23 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-bess-evpn-pref-df-08.txt>>.
- [I-D.ietf-bess-evpn-yang]  
Brissette, P., Shah, H., Hussain, I., Tiruveedhula, K., and J. Rabadan, "Yang Data Model for EVPN", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-yang-07, 11 March 2019, <<https://www.ietf.org/archive/id/draft-ietf-bess-evpn-yang-07.txt>>.
- [I-D.ietf-idr-bgp-model]  
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-13, 6 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgp-model-13.txt>>.
- [I-D.ietf-opsawg-sap]  
Boucadair, M., Dios, O. G. D., Barguil, S., Wu, Q., and V. Lopez, "A Network YANG Model for Service Attachment Points (SAPs)", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-04, 11 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-sap-04.txt>>.

[I-D.ietf-teas-enhanced-vpn]

Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-10, 6 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-10.txt>>.

[I-D.ietf-teas-ietf-network-slices]

Farrel, A., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slices-10, 27 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slices-10.txt>>.

[I-D.ietf-teas-te-service-mapping-yang]

Lee, Y., Dhody, D., Fioccola, G., Wu, Q., Ceccarelli, D., and J. Tantsura, "Traffic Engineering (TE) and Service Mapping YANG Model", Work in Progress, Internet-Draft, draft-ietf-teas-te-service-mapping-yang-10, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-teas-te-service-mapping-yang-10.txt>>.

[IEEE-802-1ah]

IEEE, "IEEE Standard for Local and metropolitan area networks -- Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", IEEE Std 801.3AH-2008, 2008, <[https://standards.ieee.org/standard/802\\_1ah-2008.html](https://standards.ieee.org/standard/802_1ah-2008.html)>.

[IEEE-802-3ah]

IEEE, "802.3ah - 2004 - IEEE Standard for Information technology-- Local and metropolitan area networks-- Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment: Media Access Control Parameters, Physical Layers, and Management Parameters for Subscriber Access Networks", IEEE Std 802.3AH-2004, 2004, <DOI 10.1109/IEEESTD.2004.94617>.

[IEEE802.1AX]

"Link Aggregation", IEEE Std 802.1AX-2020, 2020.

[IEEE802.1Q]

"Bridges and Bridged Networks", IEEE Std 802.1Q-2018, 6 July 2018, <<https://ieeexplore.ieee.org/document/8403927>>.

- [MFA] "The Use of Virtual Trunks for ATM/MPLS Control Plane Interworking Specification", MFA Forum 9.0.0 , February 2006.
- [PYANG] "pyang", November 2020,  
<<https://github.com/mbj4668/pyang>>.
- [RFC2507] Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression", RFC 2507, DOI 10.17487/RFC2507, February 1999, <<https://www.rfc-editor.org/info/rfc2507>>.
- [RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, DOI 10.17487/RFC2508, February 1999, <<https://www.rfc-editor.org/info/rfc2508>>.
- [RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <<https://www.rfc-editor.org/info/rfc3032>>.
- [RFC3545] Koren, T., Casner, S., Geevarghese, J., Thompson, B., and P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering", RFC 3545, DOI 10.17487/RFC3545, July 2003, <<https://www.rfc-editor.org/info/rfc3545>>.
- [RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <<https://www.rfc-editor.org/info/rfc3644>>.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <<https://www.rfc-editor.org/info/rfc4026>>.
- [RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <<https://www.rfc-editor.org/info/rfc4446>>.
- [RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <<https://www.rfc-editor.org/info/rfc4448>>.

- [RFC4553] Vainshtein, A., Ed. and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, DOI 10.17487/RFC4553, June 2006, <<https://www.rfc-editor.org/info/rfc4553>>.
- [RFC4618] Martini, L., Rosen, E., Heron, G., and A. Malis, "Encapsulation Methods for Transport of PPP/High-Level Data Link Control (HDLC) over MPLS Networks", RFC 4618, DOI 10.17487/RFC4618, September 2006, <<https://www.rfc-editor.org/info/rfc4618>>.
- [RFC4619] Martini, L., Ed., Kawa, C., Ed., and A. Malis, Ed., "Encapsulation Methods for Transport of Frame Relay over Multiprotocol Label Switching (MPLS) Networks", RFC 4619, DOI 10.17487/RFC4619, September 2006, <<https://www.rfc-editor.org/info/rfc4619>>.
- [RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <<https://www.rfc-editor.org/info/rfc4664>>.
- [RFC4717] Martini, L., Jayakumar, J., Bocci, M., El-Aawar, N., Brayley, J., and G. Koleyni, "Encapsulation Methods for Transport of Asynchronous Transfer Mode (ATM) over MPLS Networks", RFC 4717, DOI 10.17487/RFC4717, December 2006, <<https://www.rfc-editor.org/info/rfc4717>>.
- [RFC4816] Malis, A., Martini, L., Brayley, J., and T. Walsh, "Pseudowire Emulation Edge-to-Edge (PWE3) Asynchronous Transfer Mode (ATM) Transparent Cell Transport Service", RFC 4816, DOI 10.17487/RFC4816, February 2007, <<https://www.rfc-editor.org/info/rfc4816>>.
- [RFC4842] Malis, A., Pate, P., Cohen, R., Ed., and D. Zelig, "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation over Packet (CEP)", RFC 4842, DOI 10.17487/RFC4842, April 2007, <<https://www.rfc-editor.org/info/rfc4842>>.
- [RFC4863] Martini, L. and G. Swallow, "Wildcard Pseudowire Type", RFC 4863, DOI 10.17487/RFC4863, May 2007, <<https://www.rfc-editor.org/info/rfc4863>>.
- [RFC4901] Ash, J., Ed., Hand, J., Ed., and A. Malis, Ed., "Protocol Extensions for Header Compression over MPLS", RFC 4901, DOI 10.17487/RFC4901, June 2007, <<https://www.rfc-editor.org/info/rfc4901>>.

- [RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, DOI 10.17487/RFC5086, December 2007, <<https://www.rfc-editor.org/info/rfc5086>>.
- [RFC5087] Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, DOI 10.17487/RFC5087, December 2007, <<https://www.rfc-editor.org/info/rfc5087>>.
- [RFC5143] Malis, A., Brayley, J., Shirron, J., Martini, L., and S. Vogelsang, "Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation Service over MPLS (CEM) Encapsulation", RFC 5143, DOI 10.17487/RFC5143, February 2008, <<https://www.rfc-editor.org/info/rfc5143>>.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC6307] Black, D., Ed., Dunbar, L., Ed., Roth, M., and R. Solomon, "Encapsulation Methods for Transport of Fibre Channel Traffic over MPLS Networks", RFC 6307, DOI 10.17487/RFC6307, April 2012, <<https://www.rfc-editor.org/info/rfc6307>>.
- [RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <<https://www.rfc-editor.org/info/rfc6624>>.
- [RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <<https://www.rfc-editor.org/info/rfc7209>>.
- [RFC7267] Martini, L., Ed., Bocci, M., Ed., and F. Balus, Ed., "Dynamic Placement of Multi-Segment Pseudowires", RFC 7267, DOI 10.17487/RFC7267, June 2014, <<https://www.rfc-editor.org/info/rfc7267>>.

- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <<https://www.rfc-editor.org/info/rfc7297>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <<https://www.rfc-editor.org/info/rfc8453>>.
- [RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <<https://www.rfc-editor.org/info/rfc8519>>.
- [RFC8584] Rabadan, J., Ed., Mohanty, S., Ed., Sajassi, A., Drake, J., Nagaraj, K., and S. Sathappan, "Framework for Ethernet VPN Designated Forwarder Election Extensibility", RFC 8584, DOI 10.17487/RFC8584, April 2019, <<https://www.rfc-editor.org/info/rfc8584>>.

- [RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <<https://www.rfc-editor.org/info/rfc8792>>.
- [RFC8960] Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A YANG Data Model for MPLS Base", RFC 8960, DOI 10.17487/RFC8960, December 2020, <<https://www.rfc-editor.org/info/rfc8960>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.

## Appendix A. Examples

This section includes a non-exhaustive list of examples to illustrate the use of the L2NM.

In the following subsections, only the content of the message bodies is shown using JSON notations [RFC7951].

The examples use the folding defined in [RFC8792] for long lines.

### A.1. BGP-based VPLS

This section provides an example to illustrate how the L2NM can be used to manage BGP-based VPLS. We consider the sample VPLS service delivered using the architecture depicted in Figure 23. In accordance with [RFC4761], we assume that a full mesh is established between all PEs. The details about such full mesh are not detailed here.

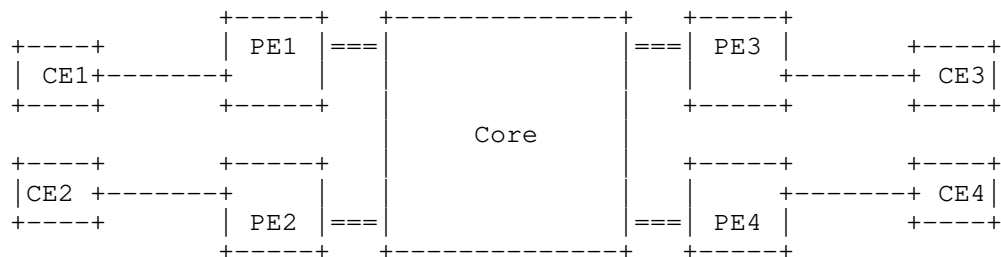


Figure 23: An Example of VPLS



Figure 24 show an example of a message body used to configure a VPLS instance using the L2NM. In this example, BGP is used for both auto-discovery and signaling. The 'signaling-type' data node is set to 'vpn-common:bgp-signaling'.

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpls7714825356",
          "vpn-description": "Sample BGP-based VPLS",
          "customer-name": "customer-7714825356",
          "vpn-type": "ietf-vpn-common:vpls",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:bgp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65535,
                "svc-mtu": 1518,
                "rd-suffix": 1,
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
              }
            ]
          }
        },
        {
          "vpn-node-id": "pe1",
          "ne-id": "198.51.100.1",
          "active-global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile"
              }
            ]
          }
        }
      ]
    }
  }
}
```

```

    }
  ]
},
"bgp-auto-discovery": {
  "vpn-id": "1"
},
"signaling-option": {
  "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
  "vpls-instance": {
    "vpls-edge-id": 1,
    "vpls-edge-id-range": 100
  }
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE1",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 1
          }
        }
      }
    }
  ]
}
},
{
  "vpn-node-id": "pe2",
  "ne-id": "198.51.100.2",
  "active-global-parameters-profiles": {
    "global-parameters-profile": [
      {
        "profile-id": "simple-profile"
      }
    ]
  }
},

```

```

    "bgp-auto-discovery": {
      "vpn-id": "1"
    },
    "signaling-option": {
      "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
      "vpls-instance": {
        "vpls-edge-id": 2,
        "vpls-edge-id-range": 100
      }
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "1/1/1.1",
          "interface-id": "1/1/1",
          "description": "Interface to CE2",
          "active-vpn-node-profile": "simple-profile",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          },
          "connection": {
            "encapsulation": {
              "encap-type": "ietf-vpn-common:dot1q",
              "dot1q": {
                "cvlan-id": 1
              }
            }
          }
        }
      ]
    }
  },
  {
    "vpn-node-id": "pe3",
    "ne-id": "198.51.100.3",
    "active-global-parameters-profiles": {
      "global-parameters-profile": [
        {
          "profile-id": "simple-profile"
        }
      ]
    }
  },
  "bgp-auto-discovery": {
    "vpn-id": "1"
  },
},

```

```

        "signaling-option": {
            "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",
            "vpls-instance": {
                "vpls-edge-id": 3,
                "vpls-edge-id-range": 100
            }
        },
        "vpn-network-accesses": {
            "vpn-network-access": [
                {
                    "id": "1/1/1.1",
                    "interface-id": "1/1/1",
                    "description": "Interface to CE3",
                    "active-vpn-node-profile": "simple-profile",
                    "status": {
                        "admin-status": {
                            "status": "ietf-vpn-common:admin-up"
                        }
                    },
                    "connection": {
                        "encapsulation": {
                            "encap-type": "ietf-vpn-common:dot1q",
                            "dot1q": {
                                "cvlan-id": 1
                            }
                        }
                    }
                }
            ]
        },
        {
            "vpn-node-id": "pe4",
            "ne-id": "198.51.100.4",
            "active-global-parameters-profiles": {
                "global-parameters-profile": [
                    {
                        "profile-id": "simple-profile"
                    }
                ]
            }
        },
        "bgp-auto-discovery": {
            "vpn-id": "1"
        },
        "signaling-option": {
            "pw-encapsulation-type": "iana-bgp-l2-encaps:ethernet\
-tagged-mode",

```

```

        "vpls-instance": {
            "vpls-edge-id": 4,
            "vpls-edge-id-range": 100
        }
    },
    "vpn-network-accesses": {
        "vpn-network-access": [
            {
                "id": "1/1/1.1",
                "interface-id": "1/1/1",
                "description": "Interface to CE4",
                "active-vpn-node-profile": "simple-profile",
                "status": {
                    "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                    }
                },
                "connection": {
                    "encapsulation": {
                        "encap-type": "ietf-vpn-common:dot1q",
                        "dot1q": {
                            "cvlan-id": 1
                        }
                    }
                }
            }
        ]
    }
}

```

Figure 24: Example of L2NM Message Body to Configure a BGP-based VPLS

#### A.2. BGP-based VPWS with LDP Signaling

Let's consider the simple architecture depicted in Figure 25 to offer a VPWS between CE1 and CE2. The service uses BGP for auto-discovery and LDP for signaling.

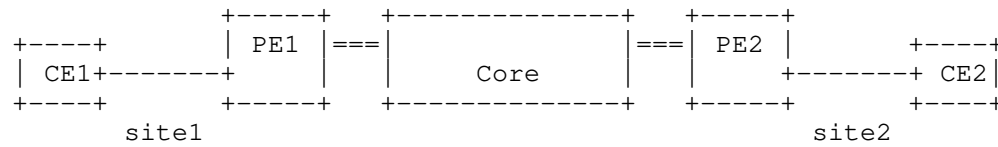


Figure 25: An Example of VPLS

```

{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpws12345",
          "vpn-description": "Sample VPWS",
          "customer-name": "customer-12345",
          "vpn-type": "ietf-vpn-common:vpws",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65550,
                "rd-auto": {
                  "auto": [
                    null
                  ]
                },
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
              }
            ]
          },
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "pe1",
                "ne-id": "2001:db8:100::1",

```

```
"active-global-parameters-profiles": {
  "global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ]
},
"bgp-auto-discovery": {
  "vpn-id": "587"
},
"signaling-option": {
  "advertise-mtu": true,
  "ldp-or-l2tp": {
    "saii": 1,
    "remote-targets": [
      {
        "taii": 2
      }
    ]
  },
  "ldp-pw-type": "ethernet"
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "1/1/1.1",
      "interface-id": "1/1/1",
      "description": "Interface to CE1",
      "active-vpn-node-profile": "simple-profile",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      }
    }
  ]
},
{
  "vpn-node-id": "pe2",
  "ne-id": "2001:db8:200::1",
  "active-global-parameters-profiles": {
    "global-parameters-profile": [
      {
        "profile-id": "simple-profile"
      }
    ]
  }
},
```

```

    "bgp-auto-discovery": {
      "vpn-id": "587"
    },
    "signaling-option": {
      "advertise-mtu": true,
      "ldp-or-l2tp": {
        "saii": 2,
        "remote-targets": [
          {
            "taii": 1
          }
        ],
        "ldp-pw-type": "ethernet"
      }
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "5/1/1.1",
          "interface-id": "5/1/1",
          "description": "Interface to CE2",
          "active-vpn-node-profile": "simple-profile",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          }
        }
      ]
    }
  ]
}

```

Figure 26: Example of L2NM Message Body to Configure a BGP-based VPWS with LDP Signaling



## A.3. LDP-based VPLS

This section provides an example to illustrate how the L2NM can be used to manage a VPLS with LDP signaling. The connectivity between the CE and the PE is direct using Dot1q encapsulation [IEEE802.1Q]. We consider the sample service delivered using the architecture depicted in Figure 27.

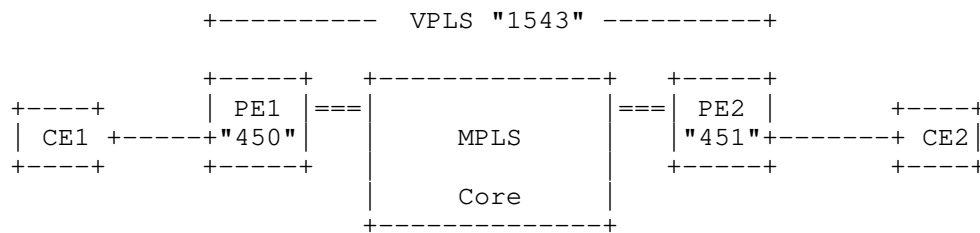


Figure 27: An Example of VPLS topology

Figure 28 shows how the L2NM is used to instruct both PE1 and PE2 to use the targeted LDP session between them to establish the VPLS "1543" between the ends. A single VPN service is created for this purpose. Additionally, two VPN Nodes and each with a corresponding VPN network access is also created.

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "450",
          "vpn-name": "CORPO-EXAMPLE",
          "vpn-description": "SEDE_CENTRO_450",
          "customer-name": "EXAMPLE",
          "vpn-type": "ietf-vpn-common:vpls",
          "vpn-service-topology": "ietf-vpn-common:hub-spoke",
          "bgp-ad-enabled": false,
          "signaling-type": "ietf-vpn-common:ldp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "ce-vlan-preservation": true,
                "ce-vlan-cos-preservation": true
              }
            ]
          }
        }
      ]
    }
  }
}
```

```
    ]
  },
  "vpn-nodes": {
    "vpn-node": [
      {
        "vpn-node-id": "450",
        "description": "SEDE_CENTRO_450",
        "ne-id": "2001:db8:5::1",
        "role": "ietf-vpn-common:hub-role",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        },
        "active-global-parameters-profiles": {
          "global-parameters-profile": [
            {
              "profile-id": "simple-profile"
            }
          ]
        },
        "signaling-option": {
          "ldp-or-l2tp": {
            "t-ldp-pw-type": "vpls-type",
            "pw-peer-list": [
              {
                "peer-addr": "2001:db8:50::1",
                "vc-id": "1543"
              }
            ]
          }
        }
      ]
    },
    "vpn-network-accesses": {
      "vpn-network-access": [
        {
          "id": "4508671287",
          "description": "VPN_450_SNA",
          "interface-id": "gigabithethernet0/0/1",
          "status": {
            "admin-status": {
              "status": "ietf-vpn-common:admin-up"
            }
          }
        },
        "connection": {
          "l2-termination-point": "550",
          "encapsulation": {
            "encap-type": "ietf-vpn-common:dot1q",
            "dot1q": {
```

```

        "tag-type": "ietf-vpn-common:c-vlan",
        "cvlan-id": 550
    }
}
},
"service": {
    "mtu": 1550,
    "svc-pe-to-ce-bandwidth": {
        "pe-to-ce-bandwidth": [
            {
                "bw-type": "ietf-vpn-common:bw-per-port",
                "cir": "20480000"
            }
        ]
    },
    "svc-ce-to-pe-bandwidth": {
        "ce-to-pe-bandwidth": [
            {
                "bw-type": "ietf-vpn-common:bw-per-port",
                "cir": "20480000"
            }
        ]
    },
    "qos": {
        "qos-profile": {
            "qos-profile": [
                {
                    "profile": "QoS_Profile_A",
                    "direction": "ietf-vpn-common:both"
                }
            ]
        }
    }
}
}
}
}
}
}
},
{
    "vpn-node-id": "451",
    "description": "SEDE_CHAPINERO_451",
    "ne-id": "2001:db8:50::1",
    "role": "ietf-vpn-common:spoke-role",
    "status": {
        "admin-status": {
            "status": "ietf-vpn-common:admin-up"
        }
    }
},

```

```
"active-global-parameters-profiles": {
  "global-parameters-profile": [
    {
      "profile-id": "simple-profile"
    }
  ]
},
"signaling-option": {
  "ldp-or-l2tp": {
    "t-ldp-pw-type": "vpls-type",
    "pw-peer-list": [
      {
        "peer-addr": "2001:db8:5::1",
        "vc-id": "1543"
      }
    ]
  }
},
"vpn-network-accesses": {
  "vpn-network-access": [
    {
      "id": "4508671288",
      "description": "VPN_450_SNA",
      "interface-id": "gigabithethernet0/0/1",
      "status": {
        "admin-status": {
          "status": "ietf-vpn-common:admin-up"
        }
      },
      "connection": {
        "l2-termination-point": "550",
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "tag-type": "ietf-vpn-common:c-vlan",
            "cvlan-id": 550
          }
        }
      },
      "service": {
        "mtu": 1550,
        "svc-pe-to-ce-bandwidth": {
          "pe-to-ce-bandwidth": [
            {
              "bw-type": "ietf-vpn-common:bw-per-port",
              "cir": "20480000"
            }
          ]
        }
      }
    }
  ]
}
```

```

    },
    "svc-ce-to-pe-bandwidth": {
      "ce-to-pe-bandwidth": [
        {
          "bw-type": "ietf-vpn-common:bw-per-port",
          "cir": "20480000"
        }
      ]
    },
    "qos": {
      "qos-profile": {
        "qos-profile": [
          {
            "profile": "QoS_Profile_A",
            "direction": "ietf-vpn-common:both"
          }
        ]
      }
    }
  }
}

```

Figure 28: Example of L2NM Message Body for LDP-based VPLS

#### A.4. VPWS-EVPN Service Instance

Figure 29 depicts a sample architecture to offer VPWS-EVPN service between CE1 and CE2. Both CEs are multi-homed. BGP sessions are maintained between these PEs as per [RFC8214]. In this EVPN instance, an All-Active redundancy mode is used.

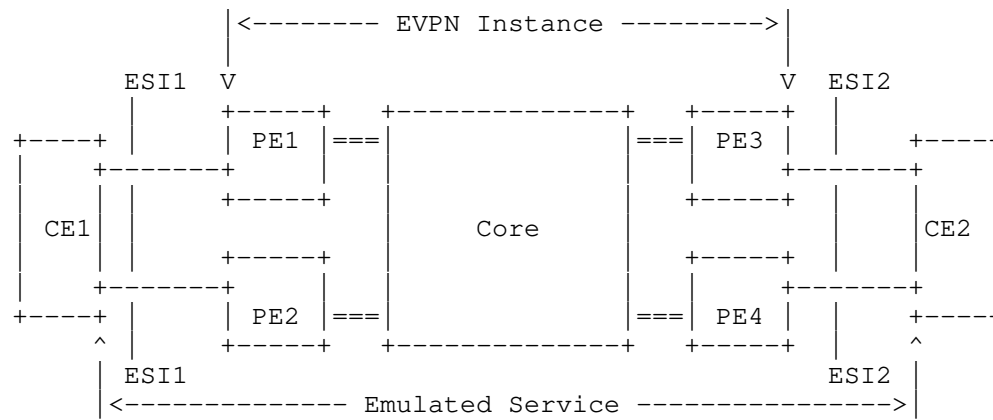


Figure 29: An Example of VPWS-EVPN

Let's first suppose that the following ES was created (Figure 30).

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "esi1",
        "ethernet-segment-identifier": "00:11:11:11:11:11:\
11:11:11",
        "esi-redundancy-mode": "all-active"
      },
      {
        "name": "esi2",
        "ethernet-segment-identifier": "00:22:22:22:22:22:\
22:22:22",
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```

Figure 30: Example of L2NM Message Body to Configure an Ethernet Segment

Figure 29 shows a simplified configuration to illustrate the use of the L2NM to configured VPWS-EVPN instance.

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "vpws15432855",
          "vpn-description": "Sample VPWS-EVPN",
          "customer-name": "customer_15432855",
          "vpn-type": "ietf-vpn-common:vpws-evpn",
          "bgp-ad-enabled": true,
          "signaling-type": "ietf-vpn-common:bgp-signaling",
          "global-parameters-profiles": {
            "global-parameters-profile": [
              {
                "profile-id": "simple-profile",
                "local-autonomous-system": 65535,
                "rd-suffix": 1,
                "vpn-target": [
                  {
                    "id": 1,
                    "route-targets": [
                      {
                        "route-target": "0:65535:1"
                      }
                    ],
                    "route-target-type": "both"
                  }
                ]
              }
            ]
          }
        }
      ],
      "vpn-nodes": {
        "vpn-node": [
          {
            "vpn-node-id": "pe1",
            "ne-id": "198.51.100.1",
            "active-global-parameters-profiles": {
              "global-parameters-profile": [
                {
                  "profile-id": "simple-profile"
                }
              ]
            }
          }
        ],
        "vpn-network-accesses": {
          "vpn-network-access": [
            {
              "id": "1/1/1.1",
              "interface-id": "1/1/1",
            }
          ]
        }
      }
    }
  }
}
```

```
    "description": "Interface to CE1",
    "active-vpn-node-profile": "simple-profile",
    "status": {
      "admin-status": {
        "status": "ietf-vpn-common:admin-up"
      }
    },
    "connection": {
      "encapsulation": {
        "encap-type": "ietf-vpn-common:dot1q",
        "dot1q": {
          "cvlan-id": 1
        }
      }
    },
    "vpws-service-instance": {
      "local-vpws-service-instance": 1111,
      "remote-vpws-service-instance": 1112
    },
    "group": [
      {
        "group-id": "gr1",
        "ethernet-segment-identifier": "es1"
      }
    ]
  }
},
{
  "vpn-node-id": "pe2",
  "ne-id": "198.51.100.2",
  "active-global-parameters-profiles": {
    "global-parameters-profile": [
      {
        "profile-id": "simple-profile"
      }
    ]
  },
  "vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1",
        "description": "Interface to CE1",
        "active-vpn-node-profile": "simple-profile",
        "status": {
          "admin-status": {
```



```
        "status": "ietf-vpn-common:admin-up"
      }
    },
    "connection": {
      "encapsulation": {
        "encap-type": "ietf-vpn-common:dot1q",
        "dot1q": {
          "cvlan-id": 1
        }
      }
    },
    "vpws-service-instance": {
      "local-vpws-service-instance": 1111,
      "remote-vpws-service-instance": 1112
    },
    "group": [
      {
        "group-id": "gr1",
        "ethernet-segment-identifier": "es11"
      }
    ]
  }
}
},
{
  "vpn-node-id": "pe3",
  "ne-id": "198.51.100.3",
  "active-global-parameters-profiles": {
    "global-parameters-profile": [
      {
        "profile-id": "simple-profile"
      }
    ]
  },
  "vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "1/1/1.1",
        "interface-id": "1/1/1",
        "description": "Interface to CE2",
        "active-vpn-node-profile": "simple-profile",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        }
      },
      "connection": {
```

```
        "encapsulation": {
          "encap-type": "ietf-vpn-common:dot1q",
          "dot1q": {
            "cvlan-id": 1
          }
        },
        "vpws-service-instance": {
          "local-vpws-service-instance": 1112,
          "remote-vpws-service-instance": 1111
        },
        "group": [
          {
            "group-id": "gr1",
            "ethernet-segment-identifier": "esi2"
          }
        ]
      }
    ],
    {
      "vpn-node-id": "pe4",
      "ne-id": "198.51.100.4",
      "active-global-parameters-profiles": {
        "global-parameters-profile": [
          {
            "profile-id": "simple-profile"
          }
        ]
      },
      "vpn-network-accesses": {
        "vpn-network-access": [
          {
            "id": "1/1/1.1",
            "interface-id": "1/1/1",
            "description": "Interface to CE2",
            "active-vpn-node-profile": "simple-profile",
            "status": {
              "admin-status": {
                "status": "ietf-vpn-common:admin-up"
              }
            }
          }
        ],
        "connection": {
          "encapsulation": {
            "encap-type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        }
      }
    }
  ]
}
```

```
    }  
  },  
  "vpws-service-instance": {  
    "local-vpws-service-instance": 1112,  
    "remote-vpws-service-instance": 1111  
  },  
  "group": [  
    {  
      "group-id": "gr1",  
      "ethernet-segment-identifier": "esi2"  
    }  
  ]  
}  
]  
}  
]  
}  
]  
}  
]  
}
```

Figure 31: Example of L2NM Message Body to Configure a VPWS-EVPN Instance

### A.5. Automatic ESI Assignment

This section provides an example to illustrate how the L2NM can be used to manage ESI auto-assignment. We consider the sample EVPN service delivered using the architecture depicted in Figure 32.

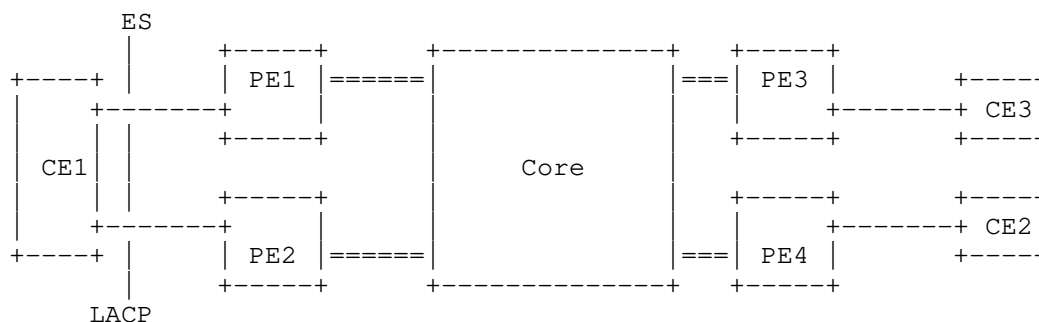


Figure 32: An Example of Automatic ESI Assignment

Figure 33 and Figure 34 show how the L2NM is used to instruct both PE1 and PE2 to auto-assign the ESI to identify the ES used with CE1. In this example, we suppose that LACP is enabled and that a Type 1 (T=0x01) is used as per Section 5 of [RFC7432]. Note that this example does not include all the details to configure the EVPN service, but focuses only on the ESI management part.

```
{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "esi1",
        "esi-type": "esi-type-1-lacp",
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```

Figure 33: Example of L2NM Message Body to Auto-Assign Ethernet Segment Identifiers

```
{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "ietf-l2vpn-ntw:vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "auto-esi-lacp",
          "vpn-description": "Sample to illustrate auto-ESI",
          "vpn-type": "ietf-vpn-common:vpws-evpn",
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "pe1",
                "ne-id": "198.51.100.1",
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "1/1/1.1",
                      "interface-id": "1/1/1",
                      "description": "Interface to CE1",
                      "status": {
                        "admin-status": {
                          "status": "ietf-vpn-common:admin-up"
                        }
                      }
                    }
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

```
    "connection": {
      "lag-interface": {
        "lag-interface-id": "1",
        "lcp": {
          "lcp-state": true,
          "system-id": "11:00:11:00:11:11",
          "admin-key": 154
        }
      }
    },
    "group": [
      {
        "group-id": "gr1",
        "ethernet-segment-identifier": "es1"
      }
    ]
  }
},
{
  "vpn-node-id": "pe2",
  "ne-id": "198.51.100.2",
  "vpn-network-accesses": {
    "vpn-network-access": [
      {
        "id": "2/2/2.5",
        "interface-id": "2/2/2",
        "description": "Interface to CE1",
        "status": {
          "admin-status": {
            "status": "ietf-vpn-common:admin-up"
          }
        }
      },
      {
        "connection": {
          "lag-interface": {
            "lag-interface-id": "1",
            "lcp": {
              "lcp-state": true,
              "system-id": "11:00:11:00:11:11",
              "admin-key": 154
            }
          }
        }
      }
    ]
  },
  "group": [
    {
      "group-id": "gr1",
      "ethernet-segment-identifier": "es1"
    }
  ]
}
```

Figure 34: An Example of L2NM Message Body for ESI Auto-Assignment

The auto-assigned ESI can be retrieved using, e.g., a GET RESTCONF method. The assigned value will be then returned as shown in the 'esi-auto' data node in Figure 35.

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-ethernet-segment:ethernet-segments": {
    "ethernet-segment": [
      {
        "name": "es1",
        "ethernet-segment-identifier": "esi-type-1-lacp",
        "esi-auto": {
          "auto-ethernet-segment-identifier": "01:11:00:11:00:11:\n11:9a:00:00"
        },
        "esi-redundancy-mode": "all-active"
      }
    ]
  }
}
```

Figure 35: An Example of L2NM Message Body to Retrieve the Assigned ESI

## A.6. VPN Network Access Precedence

In reference to the example depicted in Figure 36, an L2VPN service involves two VPN network accesses to sites that belong to the same customer.

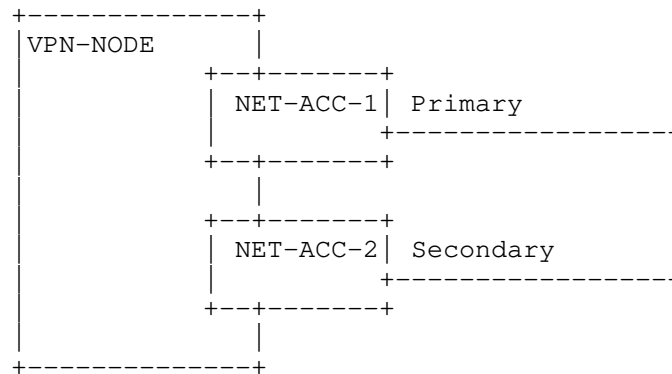


Figure 36: Example of Multiple VPN Network Accesses

In order to tag one of these VPN network accesses as "primary" and the other one as "secondary", Figure 37 shows an excerpt of the corresponding L2NM configuration. In such a configuration, both accesses are bound to the same "group-id" and the "precedence" data node set as function of the intended role of each access (primary or secondary).

```

{
  "ietf-l2vpn-ntw:l2vpn-ntw": {
    "vpn-services": {
      "vpn-service": [
        {
          "vpn-id": "Sample-Service",
          "vpn-nodes": {
            "vpn-node": [
              {
                "vpn-node-id": "VPN-NODE",
                "vpn-network-accesses": {
                  "vpn-network-access": [
                    {
                      "id": "NET-ACC-1",
                      "connection": {
                        "bearer-reference": "br1"
                      },
                      "group": [
                        {
                          "group-id": "1",
                          "precedence": "primary"
                        }
                      ]
                    },
                    {
                      "id": "NET-ACC-2",
                      "connection": {
                        "bearer-reference": "br2"
                      },
                      "group": [
                        {
                          "group-id": "1",
                          "precedence": "secondary"
                        }
                      ]
                    }
                  ]
                }
              }
            ]
          }
        }
      ]
    }
  }
}

```



Figure 37: Example of Message Body to Associate Priority Levels  
with VPN Network Accesses

#### Acknowledgements

During the discussions of this work, helpful comments, suggestions, and reviews were received from: Sergio Belotti, Italo Busi, Miguel Cros Cecilia, Joe Clarke, Dhruv Dhody, Adrian Farrel, Roque Gagliano, Christian Jacquenet, Kireeti Kompella, Julian Lucek, Moti Morgenstern, Erez Segev, and Tom Petch. Many thanks to them.

Luay Jalil, Jichun Ma, Daniel King, and Zhang Guiyu contributed to an early version of this document.

Thanks to Yingzhen Qu and Himanshu Shah for the rtgdir reviews, Ladislav Lhotka for the yangdoctors review, Chris Lonvick for the secdir review, and Dale Worley for the gen-art review. Special thanks to Adrian Farrel for the careful Shepherd review. Thanks to Robert Wilton for the careful AD review and various suggestions to enhance the model.

A YANG module for Ethernet segments was first defined in the context of the EVPN device module [I-D.ietf-bess-evpn-yang].

This work is partially supported by the European Commission under Horizon 2020 grant agreement number 101015857 Secured autonomic traffic management for a Tera of SDN flows (Teraflow).

#### Contributors

Victor Lopez  
Nokia  
Email: victor.lopez@nokia.com

Qin Wu  
Huawei  
Email: bill.wu@huawei.com

Raul Arco  
Nokia  
Email: raul.arco@nokia.com

#### Authors' Addresses

Mohamed Boucadair (editor)  
Orange  
Rennes  
France  
Email: mohamed.boucadair@orange.com

Oscar Gonzalez de Dios (editor)  
Telefonica  
Madrid  
Spain  
Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil  
Telefonica  
Madrid  
Spain  
Email: samier.barguilgiraldo.ext@telefonica.com

Luis Angel Munoz  
Vodafone  
Spain  
Email: luis-angel.munoz@vodafone.com

OPSAWG Working Group  
Internet-Draft  
Updates: 8520 (if approved)  
Intended status: Best Current Practice  
Expires: 9 April 2022

M. Richardson  
Sandelman Software Works  
W. Pan  
Huawei Technologies  
E. Lear  
Cisco Systems  
6 October 2021

Authorized update to MUD URLs  
draft-ietf-opsawg-mud-acceptable-urls-04

## Abstract

This document provides a way for an RFC8520 Manufacturer Usage Description (MUD) definitions to declare what are acceptable replacement MUD URLs for a device.

RFCEditor-please-remove: this document is being worked on at:  
<https://github.com/mcr/iot-mud-acceptable-urls>

## Status of This Memo

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

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

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

This Internet-Draft will expire on 9 April 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Updating the MUD files in place . . . . .	3
2.1. Adding capabilities . . . . .	3
2.2. Removing capabilities . . . . .	4
2.3. Significant changes to protocols . . . . .	5
2.4. Motivation for updating MUD URLs . . . . .	5
3. Updating the MUD URLs . . . . .	5
3.1. Leveraging the manufacturer signature . . . . .	6
3.2. Concerns about same-signer mechanism . . . . .	6
4. Proposed mechanism . . . . .	7
4.1. Merger, Acquisitions and Key Changes . . . . .	8
4.1.1. Changing file structure . . . . .	9
4.1.2. Changing hosting URLs . . . . .	9
4.1.3. Changing Signing Authority . . . . .	9
5. Polling for changes in MUD files . . . . .	10
6. Privacy Considerations . . . . .	10
7. Security Considerations . . . . .	10
7.1. Updating files vs Updating MUD URLs . . . . .	11
8. References . . . . .	12
8.1. Normative References . . . . .	12
8.2. Informative References . . . . .	12
Appendix A. Appendices . . . . .	13
Contributors . . . . .	13
Authors' Addresses . . . . .	13

## 1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources and associated suggested network behavior. The behaviors are described in a MUD file hosted in its manufacturer's server. The device provides a MUD URL to the network manager, which can locate this MUD file and determine the required network authorization of the device.

In some cases, e.g., the firmware update, the network behaviors of the device may change, and the description in the original MUD file will no longer apply. To solve this problem, there are two common ways which the manufacturer can use.

One is to change what is in the MUD file, i.e., update the MUD file in place, whenever the behavior of the firmware changes. Section 2 discusses three scenarios for updating the MUD file and the corresponding potential issues.

The other is to change which MUD file is processed by changing the MUD URL. Section 3 describes the common sources of MUD URLs and the problems and threats faced by each type of source when updating the MUD URL. This document proposes an enhanced mechanism of how to securely update the MUD URL in Section 4.

There are also some assumptions and prerequisites in this document.

While MUD files may include signatures, [RFC8520] does not mandate checking them, and there is not a clear way to connect the entity which signed the MUD file to the device itself. This document limits itself to situations in which the MUD file is signed, and that the MUD controller has been configured to always check the signatures, rejecting files whose signatures do not match.

[RFC8520] does not specify how MUD controllers establish their trust in the manufacturers' signing key: there are many possible solutions from manual configuration of trust anchors, some kind of automatic configuration during onboarding, but also including to Trust on First Use (TOFU). How this initial trust is established is not important for this document, it is sufficient that some satisfactory initial trust is established.

## 2. Updating the MUD files in place

Three scenarios for updating the MUD file and the corresponding potential issues are discussed below.

### 2.1. Adding capabilities

For situations where new capabilities are added to the firmware, the MUD file will detail the new access that the new firmware requires. This may involve new incoming or outgoing connections that should be authorized. Devices that have been upgraded to the new firmware will make use of the new features. Devices that have not been upgraded to the new firmware may have new connections that are authorized, but which the device does not use (outgoing connections), or which the device is not prepared to respond to (new incoming connections).

It is possible that older versions of the firmware have vulnerabilities that were not easily exploitable due to the MUD file preventing particular kinds of access. For example, an older firmware could have no credentials required (or default credentials)

access via telnet on port 23 or HTTP on port 80. The MUD file protected the device such that it could either not be accessed at all, or access was restricted to connections from a controller only.

Useful and needed upgrades to the firmware could add credentials to that service, allowing it to be opened up for more general access. The new MUD file would provide for such access, but when combined with the weak security of the old firmware, it results in a compromised device.

While there is an argument that old firmware was insecure and should be replaced, it is often the case that the upgrade process involves downtime, or can introduce risks due to needed evaluations not having been completed yet. As an example: moving vehicles (cars, airplanes, etc.) should not perform upgrades while in motion! It is probably undesirable to perform any upgrade to an airplane outside of its service facility. A vehicle owner may desire only to perform software upgrades when they are at their residence. Should there be a problem, they could make alternate arrangements for transportation. This is contrasted with the situation when the vehicle is parked at, for instance, a remote cabin. The situation for upgrades of medical devices has even more considerations involving regulatory compliance.

## 2.2. Removing capabilities

For situations where existing capabilities prove to be a problem and are to be turned off or removed in subsequent versions of the firmware, the MUD file will be updated to disallow connections that previously were allowed.

In this case, the new MUD file will forbid some connections, which the old firmware still expects to do. As explained in the previous section, upgrades may not always occur immediately upon releasing the new firmware.

In this case, the old device will be performing unwanted connections, and the MUD controller will be alerting the network owner that the device is misbehaving rather than not upgraded. This causes a false-positive situation (see [boycrieswolf]), leading to real security issues being ignored. This is a serious issue as documented also in [boywolfinfosec], and [falsemalware].

### 2.3. Significant changes to protocols

[I-D.ietf-opsawg-mud-tls] suggests MUD definitions to allow examination of TLS protocol details. Such a profile may be very specific to the TLS library which is shipped in a device. Changes to the library (including bug fixes) may cause significant changes to the profile, requiring changes to the profile described in the MUD file. Such changes are likely neither forward nor backward compatible with other versions, and in place updates to MUD files are therefore not advised.

### 2.4. Motivation for updating MUD URLs

While many small tweaks to a MUD file can be done in place, the situation described above, particularly when it comes to removing capabilities will suggest that changes to the MUD URL are in order. A strategy for doing this securely is needed, and the rest of this document provides a mechanism to do this securely.

## 3. Updating the MUD URLs

MUD URLs can come from a number of sources:

- \* IDevID Extensions
- \* DHCP option
- \* LLDP TLV
- \* [I-D.richardson-mud-qrcode] proposes to scan them from QRcodes.

The IDevID mechanism provides a URL that is asserted cryptographically by a manufacturer. However, it is difficult for manufacturers to update the IDevID of a device which is already in a box.

The DHCP and LLDP mechanisms are not signed, but are asserted by the device. A firmware update may update what MUD URL is emitted. Sufficiently well targeted malware would also be able to change the MUD URL that is emitted.

The QRcode mechanism is usually done via paper/stickers, and is typically not under the control of the device itself at all. However, being applied by a human and not easily changed, a MUD URL obtained in this fashion is likely trustworthy. (It may not, due to mixups in labeling represent the correct device, but this is a human coordination issue, and is out of scope for this document.)

The manufacturer can use all the four mechanisms above when manufacturing the device. But when considering updating the firmware, it seems like only the DHCP and LLDP mechanisms are sufficiently easy to send the new MUD URL. Because of that sensitivity, they may also be easily changed by malware!

There are mitigating mechanisms which may be enough to deal with this problem when MUD files are signed by the manufacturer.

While [RFC8520] has established a mechanism for signing of MUD files, the document does not define a way for a MUD controller to determine who should sign the MUD file for a particular device.

[RFC8520] leaves this for a local policy. There are a number of processes that could be used, but they require coordination of many players. It is expected that each industrial vertical will work out supply chain arrangements or other heuristics.

### 3.1. Leveraging the manufacturer signature

When the first time a signature of the MUD file related to a particular device-type is verified by the MUD controller, the identity of the signing authority is recorded. That is, the signing authority is pinned. This policy means that subsequent MUD files must be signed by the same entity in order to be accepted.

The trust and acceptance of the first signer may come from many sources, for example, it could be manually configured to trust which signer, or using the IDevID mechanism for the first MUD URL and the signer of the corresponding MUD file is more trustworthy, or the MUD controller can use a Trust on First Use (TOFU) mechanism and trusts the first signer by default.

Based upon this process, an update to the MUD URL would be valid if the new MUD file was signed by the same entity that signed the previous entry. This mechanism permits a replacement URL to be any URL that the same manufacturer can provide.

### 3.2. Concerns about same-signer mechanism

There is still a potential threat: a manufacturer which has many products may have a MUD definition for another product that has the privileges that the malware desires.

The malware could simply change the expressed MUD URL to that of the other product, and it will be accepted by the MUD controller as valid.



This works as long as manufacturers use a single key to sign all products. Some manufacturers could sign each product with a different key. Going logically down this path, if all these product keys are collected into a single PKI, signed by a common certification authority.

In this case, the question then becomes whether the MUD controller should pin the End-Entity (EE) certificate, or the CA certificate.

Pinning the End-Entity (EE) certificate defends against malware that changes the product type, but prevents the manufacturer from being able to cycle the validity of the End-Entity certificate for cryptographic hygiene reasons.

Pinning the CA certificate allows the EE certificate to change, but may not defend against product type changes.

It is possible to invent policy mechanisms that would link the EE certificate to a value that is in the MUD file. This could be a policy OID, or could involve some content in a subjectAltName. Future work could go in this direction. This document proposes a simpler solution.

#### 4. Proposed mechanism

The document proposes to limit what MUD URLs are considered valid from the device, limiting new MUD URLs to be variations of the initial (presumed to be secure) URL.

The first MUD file which is defined for a device can come from an IDevID (which is considered more secure), or via Trust on First Use with DHCP or LLDP or other mechanisms. This first, initially trusted, MUD file will be called the "root" MUD file.

A MUD file contains a self-referential MUD-URL attribute that points to the MUD file itself located on the vendor's website. While the IDevID, DHCP and LLDP mechanisms only transmit a URL, there are some newer, not yet standardized proposals that would fetch an entire MUD file from the device, such as [I-D.jimenez-t2trg-mud-coap].

The MUD-URL MUST always be an Absolute URI: see [RFC3986] section 4.3.

The URL found in the MUD-URL attribute is to be called the canonical MUD URL for the device.

The MUD-SIGNATURE attribute in the MUD file SHOULD be a relative URI (see [RFC3986] section 4.2) with the (hierarchical) base URI for this reference being the MUD-URL attribute.

When pinning the signature, the MUD controller SHOULD pin the lowest Certification Authority (CA) that was used in the validation of the CMS structure, along with the chain of Subject Names leading to the signature. The MUD controller may need additional trust anchors (including previously pinned ones) in order to verify that CA certificate.

Subsequent MUD files are considered valid if:

- \* they have the same initial Base-URI as the MUD-URL, but may have a different final part
- \* they are signed by an equivalent End Entity (same trusted CA and same Subject Name) as the "root" MUD file.

Section 5.2 of [RFC3986] details many cases for calculating the Base-URI. The test is simplified to: remove everything to the right of the last (rightmost) "/" in the URL of "root" MUD file URL, and the proposed new URL. The resulting two strings MUST be identical.

For a simple example, if the "root" MUD-URL is `http://example.com/hello/there/file.json` then any URL that starts with `http://example.com/hello/there/` would be acceptable, such as `http://example.com/hello/there/revision2.json`.

Once the new MUD file is accepted, then it becomes the new "root" MUD file, and any subsequent updates MUST be relative to the MUD-URL in the new file.

#### 4.1. Merger, Acquisitions and Key Changes

The above process allows for a manufacturer to rework its file structure. They can change web server host names, so long as they retain the old structure long enough for all devices to upgrade at least once.

The process also allows a manufacturer to change the EE certificate and Certification Authority used for signing.

#### 4.1.1. Changing file structure

A manufacturer has been hosting a MUD file at `https://example.com/household/products/mudfiles/toaster.json` and wishes to move it to `https://example.com/mudfiles/toasters/model1945/mud.json`

The manufacturer simply changes the MUD-URL contained with the files at the old location to have a value of `https://example.com/mudfiles/toasters/model1945/mud.json`. The manufacturer must continue to serve the files from the old location for some time, or to return an HTTP 301 (Moved Permanently) redirecting to the new location.

#### 4.1.2. Changing hosting URLs

A manufacturer has been hosting a MUD file at `https://example.com/household/products/mudfiles/toaster.json` and wishes to move it to `https://mud.example/example.com/toasters/model1945/mud.json`

The manufacturer simply changes the MUD-URL contained with the files at the old location to have a value of `https://example.com/mudfiles/toasters/model1945/mud.json`. The manufacturer has to continue to host at the old location until such time as it is sure that all MUD controllers have loaded the new data, and that all devices in the field have upgraded their URL. A 301 Redirect that changed the hostname SHOULD NOT be accepted by MUD controllers.

#### 4.1.3. Changing Signing Authority

A manufacturer has been signing MUD files using an EE Certificate with subjectAltName `foo.example`, issued by an internal Certification Authority BAZ.

The manufacturer wishes to begin signing with an EE Certificate with subjectAltname `foo.example`, but now signed by a public CA (call it: Fluffy).

The manufacturer first creates a new MUD file with a new detached signature file. Within this signature file, the manufacturer places a certificate chain: Internal-CA BAZ->Fluffy, and then the Fluffy Certificate, and then the `foo.example` certificate issued from Fluffy.

This supports changing certification authorities, but it does not support changing the Subject Name of the signing entity.

## 5. Polling for changes in MUD files

The MUD file update mechanisms described in Section 2 requires that the MUD controller poll for updates. The MUD controller will receive no signal about a change from the device because the URL will not have changed.

The manufacturer SHOULD serve mud files from a source for which ETag Section 2.3 of [RFC7232] may be generated. Static files on disk satisfy this requirement. MUD files generated from a database process might not. The use of ETag allows a MUD controller to more efficiently poll for changes in the file.

A manufacturer should also serve MUD files with an HTTP Max-Age header as per Section 5.2.2.8 of [RFC7234].

The MUD controller should take the Max-Age as an indication of when to next poll for updates to the MUD file. Values of less than 1 hour, or more than 1 month should be considered out of range, and clamped into the range (1 hour, 1 month).

MUD controllers SHOULD add some random jitter to the timing of their requests. MUD controllers MAY use a single HTTP(S)/1.1 connection to retrieve all resources at the same destination.

## 6. Privacy Considerations

The MUD URL contains sensitive model and even firmware revision numbers. Thus the MUD URL identifies the make, model and revision of a device.

[RFC8520] already identifies this privacy concern, and suggests use of TLS so that the HTTP requests that retrieve the MUD file do not divulge that level of detail.

The requirement for the MUD controller to poll for changes results in multiple interactions between the MUD controller and the manufacturer. Even if HTTPS used, an observer of the traffic to that manufacturer will be revealing, and [RFC8520] goes on to suggest use of a proxy as well.

## 7. Security Considerations

Prior to the standardization of the process in this document, if a device was infiltrated by malware, and said malware wished to make accesses beyond what the current MUD file allowed, the the malware would have to:

1. arrange for an equivalent MUD file to be visible somewhere on the Internet
2. depend upon the MUD controller either not checking signatures, or
3. somehow get the manufacturer to sign the alternate MUD
4. announce this new URL via DHCP or LLDP, updating the MUD controller with the new permissions.

One way to accomplish (3) is to leverage the existence of MUD files created by the manufacturer for different classes of devices. Such files would already be signed by the same manufacturer, eliminating the need to spoof a signature.

With the standardization of the process in this document, then the attacker can no longer point to arbitrary MUD files in step 4, but can only make use of MUD files that the manufacturer has already provided for this device.

Manufacturers are advised to maintain an orderly layout of MUD files in their web servers, with each unique product having its own directory/pathname.

The process described updates only MUD controllers and the processes that manufacturers use to manage the location of their MUD files.

A manufacturer which has not managed their MUD files in the the way described here can deploy new directories of per-product MUD files, and then can update the existing MUD files in place to point to the new URLs using the MUD-URL attribute.

There is therefore no significant flag day: MUD controllers may implement the new policy without significant concern about backwards compatibility.

#### 7.1. Updating files vs Updating MUD URLs

Device developers need to consider whether to make a change by updating a MUD file, or updating the MUD URL.

MUD URLs can only be updated by shipping a new firmware. It is reasonable to update the MUD URL whenever a new firmware release causes new connectivity to be required. The updated mechanism defined in this document makes this a secure operation, and there is no practical limitation on the number of files that a web server can hold.

In place updates to a MUD file should be restricted to cases where it turns out that the description was inaccurate: a missing connection, an inadvertent one authorized, or just incorrect information.

Developers should be aware that many enterprise web sites use outsourced content distribution networks, and MUD controllers are likely to cache files for some time. Changes to MUD files will take some time to propagate through the various caches. An updated MUD URL will however, not experience any cache issues, but can not be deployed with a firmware update.

## 8. References

### 8.1. Normative References

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

### 8.2. Informative References

- [boycrieswolf] "The Boy Who Cried Wolf", 18 January 2020, <<https://fablesfaesop.com/the-boy-who-cried-wolf.html>>.
- [boywolfinfosec] "Security Alerts - A Case of the Boy Who Cried Wolf?", 18 January 2020, <<https://www.infosecurity-magazine.com/opinions/security-alerts-boy-cried-wolf/>>.
- [falsemalware] "False malware alerts cost organizations \$1.27M annually, report says", 18 January 2020, <<https://www.scmagazine.com/home/security-news/false-malware-alerts-cost-organizations-1-27m-annually-report-says/> and <http://go.cyphort.com/Ponemon-Report-Page.html>>.

[I-D.ietf-opsawg-mud-tls]

Reddy, T., Wing, D., and B. Anderson, "Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices", Work in Progress, Internet-Draft, draft-ietf-opsawg-mud-tls-05, 27 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-mud-tls-05.txt>>.

[I-D.jimenez-t2trg-mud-coap]

Jimenez, J., "Using MUD on CoAP environments", Work in Progress, Internet-Draft, draft-jimenez-t2trg-mud-coap-00, 9 March 2020, <<https://www.ietf.org/archive/id/draft-jimenez-t2trg-mud-coap-00.txt>>.

[I-D.richardson-mud-qrcode]

Richardson, M., Latour, J., and H. H. Gharakheili, "On loading MUD URLs from QR codes", Work in Progress, Internet-Draft, draft-richardson-mud-qrcode-01, 15 May 2021, <<https://www.ietf.org/archive/id/draft-richardson-mud-qrcode-01.txt>>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <<https://www.rfc-editor.org/info/rfc7232>>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.

## Appendix A. Appendices

### Contributors

Jie Yang

Email: [jay.yang@huawei.com](mailto:jay.yang@huawei.com)

Tianqing Tang

Email: [tangtianqing@huawei.com](mailto:tangtianqing@huawei.com)

### Authors' Addresses

Michael Richardson  
Sandelman Software Works

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)

Wei Pan  
Huawei Technologies

Email: [william.panwei@huawei.com](mailto:william.panwei@huawei.com)

Eliot Lear  
Cisco Systems

Email: [lear@cisco.com](mailto:lear@cisco.com)



OPSAWG Working Group  
Internet-Draft  
Intended status: Best Current Practice  
Expires: 28 September 2022

M. Richardson  
Sandelman Software Works  
W. Pan  
Huawei Technologies  
27 March 2022

Operational Considerations for use of DNS in IoT devices  
draft-ietf-opsawg-mud-iot-dns-considerations-04

Abstract

This document details concerns about how Internet of Things devices use IP addresses and DNS names. The issue becomes acute as network operators begin deploying RFC8520 Manufacturer Usage Description (MUD) definitions to control device access.

This document makes recommendations on when and how to use DNS names in MUD files.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-opsawg-mud-iot-dns-considerations/>.

Discussion of this document takes place on the opsawg Working Group mailing list (<mailto:opsawg@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/opsawg/>.

Source for this draft and an issue tracker can be found at <https://github.com/mcr/iot-mud-dns-considerations>.

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 September 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Strategies to map names . . . . .	4
4. DNS and IP Anti-Patterns for IoT device Manufacturers . . . . .	6
4.1. Use of IP address literals in-protocol . . . . .	7
4.2. Use of non-deterministic DNS names in-protocol . . . . .	8
4.3. Use of a too inclusive DNS name . . . . .	8
5. DNS privacy and outsourcing versus MUD controllers . . . . .	9
6. Recommendations to IoT device manufacturer on MUD and DNS usage . . . . .	9
6.1. Consistently use DNS . . . . .	10
6.2. Use primary DNS names controlled by the manufacturer . . . . .	10
6.3. Use Content-Distribution Network with stable names . . . . .	10
6.4. Do not use geofenced names . . . . .	10
6.5. Prefer DNS servers learnt from DHCP/Route Advertisements . . . . .	10
7. Privacy Considerations . . . . .	11
8. Security Considerations . . . . .	12
9. References . . . . .	12
9.1. Normative References . . . . .	13
9.2. Informative References . . . . .	14
Appendix A. Appendices . . . . .	15
Authors' Addresses . . . . .	15

## 1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources. Access Control Lists (ACLs) can be defined in an RFC8520 Manufacturer Usage Description (MUD) file that permit a device to access Internet resources by DNS name.

Use of a DNS name rather than IP address in the ACL has many advantages: not only does the layer of indirection permit the mapping of name to IP address to be changed over time, it also generalizes automatically to IPv4 and IPv6 addresses, as well as permitting load balancing of traffic by many different common ways, including multi-CDN deployments wherein load balancing can account for geography and load.

At the MUD policy enforcement point -- the firewall -- there is a problem. The firewall has only access to the layer-3 headers of the packet. This includes the source and destination IP address, and if not encrypted by IPsec, the destination UDP or TCP port number present in the transport header. The DNS name is not present!

It has been suggested that one answer to this problem is to provide a forced intermediate for the TLS connections. This could in theory be done for TLS 1.2 connections. The MUD policy enforcement point could observe the Server Name Identifier (SNI) [RFC6066]. Some Enterprises do this already. But, as this involves active termination of the TCP connection (a forced circuit proxy) in order to see enough of the traffic, it requires significant effort. But, TLS 1.3 provides options to encrypt the SNI as the ESNI, which renders the practice useless in the end.

So in order to implement these name based ACLs, there must be a mapping between the names in the ACLs and layer-3 IP addresses. The first section of this document details a few strategies that are used.

The second section of this document details how common manufacturer anti-patterns get in the way this mapping.

The third section of this document details how current trends in DNS resolution such as public DNS servers, DNS over TLS (DoT), and DNS over HTTPS (DoH) cause problems for the strategies employed. Poor interactions with content-distribution networks is a frequent pathology that can result.

The fourth section of this document makes a series of recommendations ("best current practices") for manufacturers on how to use DNS, and IP addresses with specific purpose IoT devices.

The Privacy Considerations section concerns itself with issues that DNS-over-TLS and DNS-over-HTTPS are frequently used to deal with. How these concerns apply to IoT devices located within a residence or enterprise is a key concern.

The Security Considerations section covers some of the negative outcomes should MUD/firewall managers and IoT manufacturers choose not to cooperate.

## 2. Terminology

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Strategies to map names

The most naive method is to try to map IP addresses to names using the in-addr.arpa (IPv4), and ipv6.arpa (IPv6) mappings. This fails for a number of reasons:

1. it can not be done fast enough,
2. it reveals usage patterns of the devices,
3. the mapping are often incomplete,
4. even if the mapping is present, due to virtual hosting, it may not map back to the name used in the ACL.

This is not a successful strategy, its use is NOT RECOMMENDED.

XXX --- explain in detail how this can fail.

XXX --- explain N:1 vs 1:1 for virtual hosting.

The simplest successful strategy for translating names is for a MUD controller to take is to do a DNS lookup on the name (a forward lookup), and then use the resulting IP addresses to populate the physical ACLs.

There are still a number of failures possible.

The most important one is in the mapping of the names to IP addresses may be non-deterministic. [RFC1794] describes the very common mechanism that returns DNS A (or reasonably AAAA) records in a permuted order. This is known as Round Robin DNS, and it has been used for many decades. The device is intended to use the first IP address that is returned, and each query returns addresses in a different ordering, splitting the load among many servers.

This situation does not result in failures as long as all possible A/AAAA records are returned. The MUD controller and the device get a matching set, and the ACLs that are setup cover all possibilities.

There are a number of circumstances in which the list is not exhaustive. The simplest is when the round robin does not return all addresses. This is routinely done by geographical DNS load balancing system. It can also happen if there are more addresses than will conveniently fit into a DNS reply. The reply will be marked as truncated. (If DNSSEC resolution will be done, then the entire RR must be retrieved over TCP (or using a larger EDNS(0) size) before being validated)

However, in a geographical DNS load balancing system, different answers are given based upon the locality of the system asking. There may also be further layers of round-robin indirection.

Aside from the list of records being incomplete, the list may have changed between the time that the MUD controller did the lookup and the time that the IoT device does the lookup, and this change can result in a failure for the ACL to match.

In order to compensate for this, the MUD controller SHOULD regularly do DNS lookups in order to get never have stale data. These lookups need to be rate limited in order to avoid load. It may be necessary to avoid local recursive DNS servers. The MUD controller SHOULD incorporate its own recursive caching DNS server. Properly designed recursive servers should cache data for many minutes to days, while the underlying DNS data can change at a higher frequency, providing different answers to different queries!

A MUD controller that is aware of which recursive DNS server that the IoT device will use can instead query that server on a periodic basis. Doing so provides three advantages:

1. any geographic load balancing will base the decision on the geolocation of the recursive DNS server, and the recursive name server will provide the same answer to the MUD controller as to the IoT device.
2. the resulting name to IP address mapping in the recursive name server will be cached, and will remain the same for the entire advertised Time-To-Live reported in the DNS query return. This also allows the MUD controller to avoid doing unnecessary queries.
3. if any addresses have been omitted in a round-robin DNS process, the cache will have the set of addresses that were returned.

The solution of using the same caching recursive resolver as the target device is very simple when the MUD controllers is located in a residential CPE device. The device is usually also the policy enforcement point for the ACLs, and a caching resolver is typically located on the same device. In addition the convenience, there is a shared fate advantage: as all three components are running on the same device, if the device is rebooted, clearing the cache, then all three components will get restarted when the device is restarted.

Where the solution is more complex is when the MUD controller is located elsewhere in an Enterprise, or remotely in a cloud such as when a Software Defines Network (SDN) is used to manage the ACLs. The DNS servers for a particular device may not be known to the MUD controller, nor the MUD controller be even permitted to make recursive queries that server if it is known. In this case, additional installation specific mechanisms are probably needed to get the right view of DNS.

#### 4. DNS and IP Anti-Patterns for IoT device Manufacturers

In many design fields, there are good patterns that should be emulated, and often there are patterns that should not be emulated. The latter are called anti-patterns, as per [antipatterns].

This section describes a number of things with IoT manufacturers have been observed to do in the field, each of which presents difficulties for MUD enforcement points.

#### 4.1. Use of IP address literals in-protocol

A common pattern for a number of devices is to look for firmware updates in a two step process. An initial query is made (often over HTTPS, sometimes with a POST, but the method is immaterial) to a vendor system that knows whether or not an update is required.

The current firmware model of the device is sometimes provided and then the authoritative server provides a determination if a new version is required, and if so, what version. In simpler cases, an HTTPS end point is queried which provides the name and URL of the most recent firmware.

The authoritative upgrade server then responds with a URL of a firmware blob that the device should download and install. Best practice is that firmware is either signed internally ([I-D.ietf-suit-architecture]) so that it can be verified, or a hash of the blob is provided.

An authoritative server might be tempted to provide an IP address literal inside the protocol: there are two arguments (anti-patterns) for doing this.

One is that it eliminates problems to firmware updates that might be caused by lack of DNS, or incompatibilities with DNS. For instance the bug that causes interoperability issues with some recursive servers would become unpatchable for devices that were forced to use that recursive resolver type.

A second reason to avoid a DNS in the URL is when an inhouse content-distribution system is involved that involves on-demand instances being added (or removed) from a cloud computing architecture.

But, there are many problems with use of IP address literals for the location of the firmware.

The first is that the update service server must decide whether to provide an IPv4 or an IPv6 literal. A DNS name can contain both kinds of addresses, and can also contain many different IP addresses of each kind.

The second problem is that it forces the MUD file definition to contain the exact same IP address literals. It must also contain an ACL for each address literal. DNS provides a useful indirection method that naturally aggregates the addresses.

A third problem involves the use of HTTPS. IP address literals do not provide enough context for TLS ServerNameIndicator to be useful [RFC6066]. This limits the firmware repository to be a single tenant on that IP address, and for IPv4 (at least), this is no longer a sustainable use of IP addresses.

And with any non-deterministic name or address that is returned, the MUD controller is not challenged to validate the transaction, as it can not see into the communication.

Third-party content-distribution networks (CDN) tend to use DNS names in order to isolate the content-owner from changes to the distribution network.

#### 4.2. Use of non-deterministic DNS names in-protocol

A second pattern is for a control protocol to connect to a known HTTP end point. This is easily described in MUD. Within that control protocol references are made to additional content at other URLs. The values of those URLs do not fit any easily described pattern and may point at arbitrary names.

Those names are often within some third-party Content-Distribution-Network (CDN) system, or may be arbitrary names in a cloud-provider storage system such as Amazon S3 (such [AmazonS3], or [Akamai]).

Such names may be unpredictably chosen by the content provider, and not the content owner, and so impossible to insert into a MUD file.

Even if the content provider chosen names are deterministic they may change at a rate much faster than MUD files can be updated.

This in particular may apply to the location where firmware updates may be retrieved.

#### 4.3. Use of a too inclusive DNS name

Some CDNs make all customer content at a single URL (such as s3.amazonaws.com). This seems to be ideal from a MUD point of view: a completely predictable URL.

The problem is that a compromised device could then connect to the contents of any bucket, potentially attacking the data from other customers.



Exactly what the risk is depends upon what the other customers are doing: it could be limited to simply causing a distributed denial of service attack resulting to high costs to those customers, or such an attack could potentially include writing content.

Amazon has recognized the problems associated with this practice, and aims to change it to a virtual hosting model, as per [awss3virtualhosting].

The MUD ACLs provide only for permitting end points (hostnames and ports), but do not filter URLs (nor could filtering be enforced within HTTPS).

#### 5. DNS privacy and outsourcing versus MUD controllers

[RFC7858] and [RFC8094] provide for DNS over TLS (DoT) and DNS over HTTPS (DoH). [I-D.ietf-dnsop-terminology-ter] details the terms. But, even with traditional DNS over Port-53 (Do53), it is possible to outsource DNS queries to other public services, such as those operated by Google, CloudFlare, Verisign, etc.

For some users and classes of device, revealing the DNS queries to those outside entities may constitute a privacy concern. For other users the use of an insecure local resolver may constitute a privacy concern.

As described above in Section 3 the MUD controller needs to have access to the same resolver(s) as the IoT device.

#### 6. Recommendations to IoT device manufacturer on MUD and DNS usage

Inclusion of a MUD file with IoT devices is operationally quite simple. It requires only a few small changes to the DHCP client code to express the MUD URL. It can even be done without code changes via the use of a QR code affixed to the packaging (see [I-D.richardson-mud-qrcode]).

The difficult part is determining what to put into the MUD file itself. There are currently tools that help with the definition and analysis of MUD files, see [mudmaker]. The remaining difficulty is now the semantic contents of what is in the MUD file. An IoT manufacturer must now spend some time reviewing what the network communications that their device does.

This document has discussed a number of challenges that occur relating to how DNS requests are made and resolved, and it is the goal of this section to make recommendations on how to modify IoT systems to work well with MUD.

### 6.1. Consistently use DNS

For the reasons explained in Section 4.1, the most important recommendation is to avoid using IP address literals in any protocol. Names should always be used.

### 6.2. Use primary DNS names controlled by the manufacturer

The second recommendation is to allocate and use names within zones controlled by the manufacturer. These names can be populated with an alias (see [RFC8499] section 2) that points to the production system. Ideally, a different name is used for each logical function, allowing for different rules in the MUD file to be enabled and disabled.

While it used to be costly to have a large number of aliases in a web server certificate, this is no longer the case. Wildcard certificates are also commonly available which allowed for an infinite number of possible names.

### 6.3. Use Content-Distribution Network with stable names

When aliases point to a Content-Distribution Network (CDN), prefer to use stable names that point to appropriately load balanced targets. CDNs that employ very low time-to-live (TTL) values for DNS make it harder for the MUD controller to get the same answer as the IoT Device. A CDN that always returns the same set of A and AAAA records, but permutes them to provide the best one first provides a more reliable answer.

### 6.4. Do not use geofenced names

Due the problems with different answers from different DNS servers, described above, a strong recommendation is to avoid using such things.

### 6.5. Prefer DNS servers learnt from DHCP/Route Advertisements

IoT Devices should prefer doing DNS to the network provided DNS servers. Whether this is restricted to Classic DNS (Do53) or also includes using DoT/DoH is a local decision, but a locally provided DoT server SHOULD be used, as recommended by [I-D.reddy-dprive-bootstrap-dns-server] and [I-D.peterson-doh-dhcp].

The ADD WG is currently only focusing on insecure discovery mechanisms like DHCP/RA [I-D.btw-add-home] and DNS based discovery mechanisms ({I-D.pauly-add-deer}). Secure discovery of network provided DoH/DoT resolver is possible using the mechanisms discussed in [I-D.reddy-add-enterprise] section-4.

Use of public QuadX resolver instead of the provided DNS resolver, whether Do53, DoT or DoH is discouraged. Should the network provide such a resolver for use, then there is no reason not to use it, as the network operator has clearly thought about this.

Some manufacturers would like to have a fallback to using a public resolver to mitigate against local misconfiguration. There are a number of reasons to avoid this, or at least do this very carefully.

It is recommended that use of non-local resolvers is only done when the locally provided resolvers provide no answers to any queries at all, and do so repeatedly. The use of the operator provided resolvers SHOULD be retried on a periodic basis, and once they answer, there should be no further attempts to contact public resolvers.

Finally, the list of public resolvers that might be contacted MUST be listed in the MUD file as destinations that are to be permitted! This should include the port numbers (53, 853 for DoT, 443 for DoH) that will be used as well.

## 7. Privacy Considerations

The use of non-local DNS servers exposes the list of names resolved to a third parties, including passive eavesdroppers.

The use of DoT and DoH eliminates the minimizes threat from passive eavesdropped, but still exposes the list to the operator of the DoT or DoH server. There are additional methods, such as described by [I-D.pauly-dprive-oblivious-doh].

The use of unencrypted (Do53) requests to a local DNS server exposes the list to any internal passive eavesdroppers, and for some situations that may be significant, particularly if unencrypted WiFi is used. Use of Encrypted DNS connection to a local DNS recursive resolver is a preferred choice, assuming that the trust anchor for the local DNS server can be obtained, such as via [I-D.reddy-dprive-bootstrap-dns-server].

IoT devices that reach out to the manufacturer at regular intervals to check for firmware updates are informing passive eavesdroppers of the existence of a specific manufacturer's device being present at the origin location.

Identifying the IoT device type empowers the attacker to launch targeted attacks to the IoT device (e.g., Attacker can advantage of the device vulnerability).

While possession of a Large (Kitchen) Appliance at a residence may be uninteresting to most, possession of intimate personal devices (e.g., "sex toys") may be a cause for embarrassment.

IoT device manufacturers are encouraged to find ways to anonymize their update queries. For instance, contracting out the update notification service to a third party that deals with a large variety of devices would provide a level of defense against passive eavesdropping. Other update mechanisms should be investigated, including use of DNSSEC signed TXT records with current version information. This would permit DoT or DoH to convey the update notification in a private fashion. This is particularly powerful if a local recursive DoT server is used, which then communicates using DoT over the Internet.

The more complex case of section Section 4.1 postulates that the version number needs to be provided to an intelligent agent that can decide the correct route to do upgrades. The current [I-D.ietf-suit-architecture] specification provides a wide variety of ways to accomplish the same thing without having to divulge the current version number.

The use of a publically specified firmware update protocol would also enhance privacy of IoT devices. In such a system the IoT device would never contact the manufacturer for version information or for firmware itself. Instead, details of how to query and where to get the firmware would be provided as a MUD extension, and an Enterprise-wide mechanism would retrieve firmware, and then distribute it internally. Aside from the bandwidth savings of downloading the firmware only once, this also makes the number of devices active confidential, and provides some evidence about which devices have been upgraded and which ones might still be vulnerable. (The unpatched devices might be lurking, powered off, lost in a closet)

## 8. Security Considerations

This document deals with conflicting Security requirements:

1. devices which an operator wants to manage using [RFC8520]
2. requirements for the devices to get access to network resources that may be critical to their continued safe operation.

This document takes the view that the two requirements do not need to be in conflict, but resolving the conflict requires some advance planning by all parties.

## 9. References

## 9.1. Normative References

- [Akamai] "Akamai", 2019,  
<[https://en.wikipedia.org/wiki/Akamai\\_Technologies](https://en.wikipedia.org/wiki/Akamai_Technologies)>.
- [AmazonS3] "Amazon S3", 2019,  
<[https://en.wikipedia.org/wiki/Amazon\\_S3](https://en.wikipedia.org/wiki/Amazon_S3)>.
- [I-D.ietf-dnsop-terminology-ter]  
Hoffman, P., "Terminology for DNS Transports and Location", Work in Progress, Internet-Draft, draft-ietf-dnsop-terminology-ter-02, 3 August 2020,  
<<https://www.ietf.org/archive/id/draft-ietf-dnsop-terminology-ter-02.txt>>.
- [I-D.ietf-suit-architecture]  
Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", Work in Progress, Internet-Draft, draft-ietf-suit-architecture-16, 27 January 2021, <<https://www.ietf.org/archive/id/draft-ietf-suit-architecture-16.txt>>.
- [I-D.peterson-doh-dhcp]  
Peterson, T., "DNS over HTTP resolver announcement Using DHCP or Router Advertisements", Work in Progress, Internet-Draft, draft-peterson-doh-dhcp-01, 21 October 2019, <<https://www.ietf.org/archive/id/draft-peterson-doh-dhcp-01.txt>>.
- [I-D.reddy-dprive-bootstrap-dns-server]  
Reddy, T., Wing, D., Richardson, M. C., and M. Boucadair, "A Bootstrapping Procedure to Discover and Authenticate DNS-over-TLS and DNS-over-HTTPS Servers", Work in Progress, Internet-Draft, draft-reddy-dprive-bootstrap-dns-server-08, 6 March 2020,  
<<https://www.ietf.org/archive/id/draft-reddy-dprive-bootstrap-dns-server-08.txt>>.
- [RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794, DOI 10.17487/RFC1794, April 1995,  
<<https://www.rfc-editor.org/info/rfc1794>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

## 9.2. Informative References

- [antipatterns] "AntiPattern", 12 July 2021, <<https://www.agilealliance.org/glossary/antipattern>>.
- [awss3virtualhosting] "Down to the Wire: AWS Delays 'Path-Style' S3 Deprecation at Last Minute", 12 July 2021, <<https://techmonitor.ai/technology/cloud/aws-s3-path-deprecation>>.
- [I-D.btw-add-home] Boucadair, M., Reddy, T., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for Encrypted DNS Discovery", Work in Progress, Internet-Draft, draft-btw-add-home-12, 22 January 2021, <<https://www.ietf.org/archive/id/draft-btw-add-home-12.txt>>.

[I-D.pauly-dprive-oblivious-doh]

Kinnear, E., McManus, P., Pauly, T., Verma, T., and C. A. Wood, "Oblivious DNS Over HTTPS", Work in Progress, Internet-Draft, draft-pauly-dprive-oblivious-doh-11, 17 February 2022, <<https://www.ietf.org/archive/id/draft-pauly-dprive-oblivious-doh-11.txt>>.

[I-D.reddy-add-enterprise]

Reddy, T. and D. Wing, "DNS-over-HTTPS and DNS-over-TLS Server Deployment Considerations for Enterprise Networks", Work in Progress, Internet-Draft, draft-reddy-add-enterprise-00, 23 June 2020, <<https://www.ietf.org/archive/id/draft-reddy-add-enterprise-00.txt>>.

[I-D.richardson-mud-qrcode]

Richardson, M., Latour, J., and H. H. Gharakheili, "On loading MUD URLs from QR codes", Work in Progress, Internet-Draft, draft-richardson-mud-qrcode-07, 21 March 2022, <<https://www.ietf.org/archive/id/draft-richardson-mud-qrcode-07.txt>>.

[mudmaker] "Mud Maker", 2019, <<https://mudmaker.org>>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.

## Appendix A. Appendices

### Authors' Addresses

Michael Richardson  
Sandelman Software Works  
Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)

Wei Pan  
Huawei Technologies  
Email: [william.panwei@huawei.com](mailto:william.panwei@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 7 September 2022

E. Lear  
Cisco Systems  
S. Rose  
NIST  
6 March 2022

Discovering and Retrieving Software Transparency and Vulnerability  
Information  
draft-ietf-opsawg-sbom-access-05

Abstract

To improve cybersecurity posture, automation is necessary to locate what software is running on a device, whether that software has known vulnerabilities, and what, if any recommendations suppliers may have. This memo specifies a model to provide access to this information. It may optionally be discovered through manufacturer usage descriptions.

Status of This Memo

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

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

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

This Internet-Draft will expire on 7 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.



This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. How This Information Is Retrieved . . . . .	5
1.2. Formats . . . . .	5
1.3. Discussion points . . . . .	5
2. The well-known transparency endpoint set . . . . .	6
3. The mud-transparency extension model extension . . . . .	6
4. The mud-sbom augmentation to the MUD YANG model . . . . .	6
5. Examples . . . . .	10
5.1. Without ACLS . . . . .	10
5.2. SBOM Located on the Device . . . . .	12
5.3. Further contact required. . . . .	13
5.4. With ACLS . . . . .	14
6. Security Considerations . . . . .	16
7. IANA Considerations . . . . .	18
7.1. MUD Extension . . . . .	18
7.2. YANG Registration . . . . .	18
7.3. Well-Known Prefix . . . . .	18
8. Acknowledgments . . . . .	19
9. References . . . . .	19
9.1. Normative References . . . . .	19
9.2. Informative References . . . . .	20
Appendix A. Changes from Earlier Versions . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction

A number of activities have been working to improve visibility to what software is running on a system, and what vulnerabilities that software may have[EO2021].

Put simply, we seek to answer two classes of questions \*at scale\*:

- \* Is this system vulnerable to a particular vulnerability?
- \* Which devices in a particular environment contain vulnerabilities that require some action?

This memo doesn't specify the format of this information, but rather only how to locate and retrieve these objects.

Software bills of materials (SBOMs) are descriptions of what software, including versioning and dependencies, a device contains. There are different SBOM formats such as Software Package Data Exchange [SPDX] or CycloneDX[CycloneDX12].

System vulnerabilities may similarly be described using several data formats, including the aforementioned CycloneDX, Common Vulnerability Reporting Framework [CVRF], the Common Security Advisory Format [CSAF]. This information is typically used to report to customers the state of a system.

These two classes of information can be used in concert. For instance, a network management tool may discover that a system makes use of a particular software component that has a known vulnerability, and a vulnerability report may be used to indicate what if any versions of software correct that vulnerability, or whether the system exercises the vulnerable code at all.

Both classes of information elements are optional under the model specified in this memo. One can provide only an SBOM, only vulnerability information, or both an SBOM and vulnerability information.

Note that SBOM formats may also carry other information, the most common being any licensing terms. Because this specification is neutral regarding content, it is left for format developers such as the Linux Foundation, OASIS, and ISO to decide what attributes they will support.

This memo does not specify how vulnerability information may be retrieved directly from the endpoint. That's because vulnerability information changes occur at different rates to software updates. However, some SBOM formats may also contain vulnerability information.

SBOMs and vulnerability information are advertised and retrieved through the use of a YANG augmentation of the Manufacturer User Description (MUD) model [RFC8520]. Note that the schema creates a grouping that can also be used independently of MUD. Moreover, other MUD features, such as access controls, needn't be present.

The mechanisms specified in this document are meant to satisfy several use cases:

- \* A network-layer management system retrieving information from an IoT device as part of its ongoing lifecycle. Such devices may or may not have query interfaces available.
- \* An application-layer management system retrieving vulnerability or SBOM information in order to evaluate the posture of an application server of some form. These application servers may themselves be containers or hypervisors. Discovery of the topology of a server is beyond the scope of this memo.

To satisfy these two key use cases, objects may be found in one of three ways:

- \* on devices themselves
- \* on a web site (e.g., via URI)
- \* through some form of out-of-band contact with the supplier.

In the first case, devices will have interfaces that permit direct retrieval. Examples of these interfaces might be an HTTP, COAP or [OpenC2] endpoint for retrieval. There may also be private interfaces as well.

In the second case, when a device does not have an appropriate retrieval interface, but one is directly available from the manufacturer, a URI to that information **MUST** be discovered.

In the third case, a supplier may wish to make an SBOM or vulnerability information available under certain circumstances, and may need to individually evaluate requests. The result of that evaluation might be the SBOM or vulnerability itself or a restricted URL or no access.

To enable application-layer discovery, this memo defines a well-known URI [RFC8615]. Management or orchestration tools can query this well-known URI to retrieve a system's SBOM or vulnerability information. Further queries may be necessary based on the content and structure of the response.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 1.1. How This Information Is Retrieved

For devices that can emit a URL or can establish a well-known URI, the mechanism may be highly automated. For devices that have a URL in either their documentation or within a QR code on a box, the mechanism is semi-automated (someone has to scan the QR code or enter the URL).

Note that vulnerability and SBOM information is likely to change at different rates. The MUD semantics provide a way for manufacturers to control how often tooling should check for those changes through the cache-validity node.

### 1.2. Formats

There are multiple ways to express both SBOMs and vulnerability information. When these are retrieved either directly from the device or directly from a web server, tools will need to observe the content-type header to determine precisely which format is being transmitted. Because IoT devices in particular have limited capabilities, use of a specific Accept: header in HTTP or the Accept Option in CoAP is NOT RECOMMENDED. Instead, backend tooling is encouraged to support all known formats, and SHOULD silently discard SBOM information sent with a media type that is not understood.

Some formats may support both vulnerability and software inventory information. When both vulnerability and software inventory information is available from the same location, both sbom and vuln nodes MUST indicate that. Network management systems retrieving this information MUST take note that the identical resource is being retrieved rather than retrieving it twice.

### 1.3. Discussion points

The following is discussion to be removed at time of RFC publication.

- \* Is the model structured correctly?
- \* Are there other retrieval mechanisms that need to be specified?
- \* Do we need to be more specific in how to authenticate and retrieve SBOMs?
- \* What are the implications if the MUD URL is an extension in a certificate (e.g. an IDevID cert)?

## 2. The well-known transparency endpoint set

Two well known endpoints are defined:

- \* `"/.well-known/sbom"` retrieves an SBOM.
- \* `"/.well-known/openc2"` is the HTTPS binding to OpenC2.

As discussed previously, the precise format of a response is based on the Content-type provided.

## 3. The mud-transparency extension model extension

We now formally define this extension. This is done in two parts. First, the extension name "transparency" is listed in the "extensions" array of the MUD file. N.B., this schema extension is intended to be used wherever it might be appropriate (e.g., not just MUD).

Second, the "mud" container is augmented with a list of SBOM sources.

This is done as follows:

```
module: ietf-mud-transparency
```

```
augment /mud:mud:
  +--rw transparency
    +--rw (sbom-retrieval-method)?
      +--:(cloud)
        +--rw sboms* [version-info]
          +--rw version-info      string
          +--rw sbom-url?         inet:uri
      +--:(local-well-known)
        +--rw sbom-local-well-known? enumeration
      +--:(sbom-contact-info)
        +--rw sbom-contact-uri    inet:uri
    +--rw (vuln-retrieval-method)?
      +--:(cloud)
        +--rw vuln-url?           inet:uri
      +--:(vuln-contact-info)
        +--rw contact-uri         inet:uri
```

## 4. The mud-sbom augmentation to the MUD YANG model

```
<CODE BEGINS>
file "ietf-mud-transparency@2021-10-22.yang"
module ietf-mud-transparency {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-transparency";
  prefix mudtx;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991";
  }
  import ietf-mud {
    prefix mud;
    reference "RFC 8520";
  }
}
```

organization

"IETF OPSAWG (Ops Area) Working Group";

contact

"WG Web: <http://tools.ietf.org/wg/opsawg/>

WG List: [opsawg@ietf.org](mailto:opsawg@ietf.org)

Editor: Eliot Lear [lear@cisco.com](mailto:lear@cisco.com)

Editor: Scott Rose [scott.rose@nist.gov](mailto:scott.rose@nist.gov)";

description

"This YANG module augments the ietf-mud model to provide for reporting of SBOMs and vulnerability information.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

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 BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here. ";

```
revision 2021-07-06 {
  description
    "Initial proposed standard.";
  reference
    "RFC XXXX: Discovering and Retrieving Software Transparency
    and Vulnerability Information";
}

grouping transparency-extension {
  description
    "Transparency extension grouping";
  container transparency {
    description
      "container of methods to get an SBOM.";
    choice sbom-retrieval-method {
      description
        "How to find SBOM information";
      case cloud {
        list sboms {
          key "version-info";
          description
            "A list of SBOMs tied to different s/w
            or h/w versions.";
          leaf version-info {
            type string;
            description
              "The version to which this SBOM refers.";
          }
          leaf sbom-url {
            type inet:uri;
            description
              "A statically located URI.";
          }
        }
      }
      case local-well-known {
        leaf sbom-local-well-known {
          type enumeration {
            enum http {
              description
                "Use http (insecure) to retrieve
                SBOM information. This method is NOT RECOMMENDED,
                but may be unavoidable for certain classes of
                deployment, where TLS has not or cannot be implemented";
            }
            enum https {
              description
                "Use https (secure) to retrieve SBOM information.";
            }
          }
        }
      }
    }
  }
}
```

```
    }
    enum coap {
      description
        "Use COAP (insecure) to retrieve SBOM. This method
        is NOT RECOMMENDED, although it may be unavoidable
        for certain classes of implementations/deployments.";
    }
    enum coaps {
      description
        "Use COAPS (secure) to retrieve SBOM";
    }
    enum openc2 {
      description
        "Use OpenC2 endpoint.
        This is https://{host}/.well-known/openc2";
    }
  }
  description
    "Which communication protocol to choose.";
}
}
case sbom-contact-info {
  leaf sbom-contact-uri {
    type inet:uri;
    mandatory true;
    description
      "This MUST be either a tel, http, https, or
      mailto uri schema that customers can use to
      contact someone for SBOM information.";
  }
}
}
choice vuln-retrieval-method {
  description
    "How to find vulnerability information";
  case cloud {
    leaf vuln-url {
      type inet:uri;
      description
        "A statically located URL.";
    }
  }
}
case vuln-contact-info {
  leaf contact-uri {
    type inet:uri;
    mandatory true;
    description
      "This MUST be either a tel, http, https, or
```



```
        mailto uri schema that customers can use to
        contact someone for vulnerability information.";
    }
  }
}

augment "/mud:mud" {
  description
    "Add extension for software transparency.";
  uses transparency-extension;
}
}
<CODE ENDS>
```

## 5. Examples

In this example MUD file that uses a cloud service, the modelX presents a location of the SBOM in a URL. Note, the ACLs in a MUD file are NOT required, although they are a very good idea for IP-based devices.

### 5.1. Without ACLS

This first MUD file demonstrates how to get SBOM and vulnerability information without ACLs.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:29:12+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

The second example demonstrates that just SBOM information is included.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:29:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

## 5.2. SBOM Located on the Device

In this example, the SBOM is retrieved from the device, while vulnerability information is available from the cloud. This is likely a common case, because vendors may learn of vulnerability information more frequently than they update software.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "transparency"
    ],
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot-device.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot-device.example.com/modelX.json",
    "mud-signature": "https://iot-device.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:25:14+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot-device.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

### 5.3. Further contact required.

In this example, the network manager must take further steps to retrieve SBOM information. Vulnerability information is still available.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "transparency"
    ],
    "ietf-mud-transparency:transparency": {
      "contact-info": "https://iot-device.example.com/contact-info.html",
      "vuln-url": "https://iot-device.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot-device.example.com/modelX.json",
    "mud-signature": "https://iot-device.example.com/modelX.p7s",
    "last-update": "2021-07-09T06:16:42+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot-device.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

#### 5.4. With ACLS

Finally, here is a complete example where the device provides SBOM and vulnerability information, as well as access-control information.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:30:31+00:00",
    "cache-validity": 48,
  }
}
```

```
"is-supported": true,
"systeminfo": "retrieving vuln and SBOM info via a cloud service",
"mfg-name": "Example, Inc.",
"documentation": "https://iot.example.com/doc/modelX",
"model-name": "modelX",
"from-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-65443-v4fr"
      }
    ]
  }
},
"to-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-65443-v4to"
      }
    ]
  }
},
"ietf-access-control-list:acls": {
  "acl": [
    {
      "name": "mud-65443-v4to",
      "type": "ipv4-acl-type",
      "aces": {
        "ace": [
          {
            "name": "cl0-todev",
            "matches": {
              "ipv4": {
                "ietf-acldns:src-dnsname": "iotserver.example.com"
              }
            },
            "actions": {
              "forwarding": "accept"
            }
          }
        ]
      }
    }
  ]
},
{
  "name": "mud-65443-v4fr",
  "type": "ipv4-acl-type",
```

```
"aces": {
  "ace": [
    {
      "name": "cl0-frdev",
      "matches": {
        "ipv4": {
          "ietf-acldns:dst-dnsname": "iotserver.example.com"
        }
      },
      "actions": {
        "forwarding": "accept"
      }
    }
  ]
}
]
```

At this point, the management system can attempt to retrieve the SBOM, and determine which format is in use through the content-type header on the response to a GET request, independently repeat the process for vulnerability information, and apply ACLs, as appropriate.

## 6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

N.B., for MUD, the mandatory method of retrieval is TLS.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

The ietf-mud-transparency module has no operational impact on the element itself, and is used to discover state information that may be available on or off the element. In as much as the module itself is made writeable, this only indicates a change in how to retrieve what read-only elements. However, that does not mean there are no risks. These are discussed below, and are applicable to all nodes within the transparency container.

If an attacker modifies the elements, they may misdirect automation to retrieve a different set of URLs than was intended by the designer. This in turn leads to two specific sets of risks:

- \* the information retrieved would be false.
- \* the URLs themselves point to malware.

To address either risk, any change in a URL, and in particular to the authority section, should be treated with some suspicion. One mitigation would be to test any cloud-based URL against a reputation service.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

SBOMs provide an inventory of software. If software is available to an attacker, the attacker may well already be able to derive this very same software inventory. Manufacturers MAY restrict access to SBOM information using appropriate authorization semantics within HTTP. In particular, if a system attempts to retrieve an SBOM via HTTP and the client is not authorized, the server MUST produce an appropriate error, with instructions on how to register a particular client. One example may be to issue a certificate to the client for this purpose after a registration process has taken place. Another example would involve the use of OAUTH in combination with a federations of SBOM servers.



Another risk is a skew in the SBOM listing and the actual software inventory of a device/container. For example, a manufacturer may update the SBOM on its server, but an individual device has not been upgraded yet. This may result in an incorrect policy being applied to a device. A unique mapping of a device's software version and its SBOM can minimize this risk.

To further mitigate attacks against a device, manufacturers SHOULD recommend access controls.

Vulnerability information is generally made available to such databases as NIST's National Vulnerability Database. It is possible that vendor may wish to release information early to some customers. We do not discuss here whether that is a good idea, but if it is employed, then appropriate access controls and authorization SHOULD be applied to the vulnerability resource.

## 7. IANA Considerations

### 7.1. MUD Extension

The IANA is requested to add "transparency" to the MUD extensions registry as follows:

Extension Name: transparency  
Standard reference: This document

### 7.2. YANG Registration

The following YANG module should be registered in the "YANG Module Names" registry:

Name: ietf-mud  
URN: urn:ietf:params:xml:ns:yang:ietf-mud-transparency  
Prefix: mudtx  
Registrant contact: The IESG  
Reference: This memo

### 7.3. Well-Known Prefix

The following well known URIs are requested in accordance with [RFC8615]:

URI suffix: "sbom"  
Change controller: "IETF"  
Specification document: This memo  
Related information: See ISO/IEC 19970-2 and SPDX.org

URI suffix: "openc2"  
Change controller: "IETF"  
Specification document: This memo  
Related information: OpenC2 Project

## 8. Acknowledgments

Thanks to Russ Housley, Dick Brooks, Tom Petch, Nicolas Comstedt, who provided review comments.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

## 9.2. Informative References

- [CSAF] OASIS, "Common Security Advisory Format", July 2021, <<https://github.com/oasis-tcs/csaf>>.
- [CVRF] Santos, O., Ed., "Common Vulnerability Reporting Framework (CVRF) Version 1.2", September 2017, <<https://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.pdf>>.
- [CycloneDX12] cyclonedx.org, "CycloneDX XML Reference v1.2", May 2020.
- [EO2021] Biden, J., "Executive Order 14028, Improving the Nations Cybersecurity", May 2021.
- [OpenC2] Lemire, D., Ed., "Specification for Transfer of OpenC2 Messages via HTTPS Version 1.0", July 2019, <<https://docs.oasis-open.org/openc2/open-impl-https/v1.0/open-impl-https-v1.0.html>>.
- [SPDX] The Linux Foundation, "SPDX Specification 2.1", 2016.

## Appendix A. Changes from Earlier Versions

Draft -04: \* Address review comments

Draft -02:

\* include vulnerability information

Draft -01:

\* some modest changes

Draft -00:

\* Initial revision

Authors' Addresses

Eliot Lear  
Cisco Systems  
Richtistrasse 7  
CH-8304 Wallisellen  
Switzerland  
Phone: +41 44 878 9200  
Email: [lear@cisco.com](mailto:lear@cisco.com)

Scott Rose  
NIST  
100 Bureau Dr  
Gaithersburg MD, 20899  
United States of America  
Phone: +1 301-975-8439  
Email: [scott.rose@nist.gov](mailto:scott.rose@nist.gov)

OPSAWG  
Internet-Draft  
Intended status: Informational  
Expires: 8 September 2022

B. Claise  
J. Quilbeuf  
Huawei  
D. Lopez  
Telefonica I+D  
D. Voyer  
Bell Canada  
T. Arumugam  
Cisco Systems, Inc.  
7 March 2022

Service Assurance for Intent-based Networking Architecture  
draft-ietf-opsawg-service-assurance-architecture-03

Abstract

This document describes an architecture for Service Assurance for Intent-based Networking (SAIN). This architecture aims at assuring that service instances are running as expected. As services rely upon multiple sub-services provided by the underlying network devices and functions, getting the assurance of a healthy service is only possible with a holistic view of all involved elements. This architecture not only helps to correlate the service degradation with the network root cause but also the impacted services when a network component fails or degrades.

Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Terminology . . . . .	2
2. Introduction . . . . .	5
3. Architecture . . . . .	7
3.1. Inferring a Service Instance Configuration into an Assurance Graph . . . . .	10
3.1.1. Circular Dependencies . . . . .	12
3.2. Intent and Assurance Graph . . . . .	16
3.3. Subservices . . . . .	16
3.4. Building the Expression Graph from the Assurance Graph . . . . .	17
3.5. Building the Expression from a Subservice . . . . .	18
3.6. Open Interfaces with YANG Modules . . . . .	18
3.7. Handling Maintenance Windows . . . . .	18
3.8. Flexible Architecture . . . . .	19
3.9. Timing . . . . .	20
3.10. New Assurance Graph Generation . . . . .	21
4. Security Considerations . . . . .	21
5. IANA Considerations . . . . .	22
6. Contributors . . . . .	22
7. Open Issues . . . . .	22
8. References . . . . .	22
8.1. Normative References . . . . .	22
8.2. Informative References . . . . .	22
Appendix A. Changes between revisions . . . . .	24
Acknowledgements . . . . .	25
Authors' Addresses . . . . .	25

## 1. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

**SAIN agent:** A functional component that communicates with a device, a set of devices, or another agent to build an expression graph from a received assurance graph and perform the corresponding computation of the health status and symptoms.

**Assurance case:** According to [Piovesan2017]: "An assurance case is a structured argument, supported by evidence, intended to justify that a system is acceptably assured relative to a concern (such as safety or security) in the intended operating environment."

**Assurance graph:** A Directed Acyclic Graph (DAG) representing the assurance case for one or several service instances. The nodes (also known as vertices in the context of DAG) are the service instances themselves and the subservices, the edges indicate a dependency relations.

**SAIN collector:** A functional component that fetches or receives the computer-consumable output of the SAIN agent(s) and displays it in a user friendly form or process it locally.

**DAG:** Directed Acyclic Graph.

**ECMP:** Equal Cost Multiple Paths

**Expression graph:** A generic term for a DAG representing a computation in SAIN. More specific terms are:

- \* **Subservice expressions:** Is an expression graph representing all the computations to execute for a subservice.
- \* **Service expressions:** Is an expression graph representing all the computations to execute for a service instance, i.e., including the computations for all dependent subservices.
- \* **Global computation graph:** Is an expression graph representing all the computations to execute for all services instances (i.e., all computations performed).

**Dependency:** The directed relationship between subservice instances in the assurance graph.

**Informational Dependency:** Type of dependency whose health score does not impact the health score of its parent subservice or service instance(s) in the assurance graph. However, the symptoms should be taken into account in the parent service instance or subservice instance(s), for informational reasons.

**Impacting Dependency:** Type of dependency whose score impacts the score of its parent subservice or service instance(s) in the assurance graph. The symptoms are taken into account in the parent service instance or subservice instance(s), as the impacting reasons.

**Metric:** An information retrieved from the network running the assured service.

**Metric engine:** A functional components that maps metrics to a list of candidate metric implementations depending on the network element.

**Metric implementation:** Actual way of retrieving a metric from a network element.

**Network service YANG module:** describes the characteristics of a service as agreed upon with consumers of that service [RFC8199].

**Service instance:** A specific instance of a service.

**Service configuration orchestrator:** Quoting RFC8199, "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

**SAIN orchestrator:** A functional component that is in charge of fetching the configuration specific to each service instance and converting it into an assurance graph.

**Health status:** Score and symptoms indicating whether a service instance or a subservice is "healthy". A non-maximal score must always be explained by one or more symptoms.

**Health score:** Integer ranging from 0 to 100 indicating the health of a subservice. A score of 0 means that the subservice is broken, a score of 100 means that the subservice in question is operating as expected.

**Subservice:** Part or functionality of the network system that can be independently assured as a single entity in assurance graph.



Strongly connected component: subset of a directed graph such that there is a (directed) path from any node of the subset to any other node. A DAG does not contain any strongly connected component.

Symptom: Reason explaining why a service instance or a subservice is not completely healthy.

## 2. Introduction

Network Service YANG Modules [RFC8199] describe the configuration, state data, operations, and notifications of abstract representations of services implemented on one or multiple network elements.

Quoting RFC8199: "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

Service configuration orchestrators deploy Network Service YANG Modules [RFC8199] that will infer network-wide configuration and, therefore the configuration of the appropriate device modules (Section 3 of [RFC8969]). Network configuration is based on these device YANG modules, with protocol/encoding such as NETCONF/XML [RFC6241], RESTCONF/JSON [RFC8040], gNMI/gRPC/protobuf, etc. Knowing that a configuration is applied doesn't imply that the service is running as expected (e.g., the service might be degraded because of a failure in the network), the network operator must monitor the service operational data at the same time as the configuration (Section 3.3 of [RFC8969]). The industry has been standardizing on telemetry to push network element performance information.

A network administrator needs to monitor her network and services as a whole, independently of the use cases or the management protocols. With different protocols come different data models, and different ways to model the same type of information. When network administrators deal with multiple protocols, the network management must perform the difficult and time-consuming job of mapping data models: the model used for configuration with the model used for monitoring. This problem is compounded by a large, disparate set of data sources (MIB modules, YANG models [RFC7950], IPFIX information

elements [RFC7011], syslog plain text [RFC3164], TACACS+ [RFC8907], RADIUS [RFC2865], etc.). In order to avoid this data model mapping, the industry converged on model-driven telemetry to stream the service operational data, reusing the YANG models used for configuration. Model-driven telemetry greatly facilitates the notion of closed-loop automation whereby events/status from the network drive remediation changes back into the network.

However, it proves difficult for network operators to correlate the service degradation with the network root cause. For example, why does my L3VPN fail to connect? Why is this specific service slow? The reverse, i.e., which services are impacted when a network component fails or degrades, is even more interesting for the operators. For example, which services are impacted when this specific optic dBm begins to degrade? Which applications are impacted by this ECMP imbalance? Is that issue actually impacting any other customers?

Intent-based approaches are often declarative, starting from a statement of "The service works as expected" and trying to enforce it. Such approaches are mainly suited for greenfield deployments.

Aligned with Section 3.3 of [RFC7149], and instead of approaching intent from a declarative way, this architecture focuses on already defined services and tries to infer the meaning of "The service works as expected". To do so, the architecture works from an assurance graph, deduced from the service definition and from the network configuration. In some cases, the assurance graph may also be explicitly completed to add an intent not exposed in the service model itself (e.g. the service must rely on a backup physical path). This assurance graph is decomposed into components, which are then assured independently. The root of the assurance graph represents the service to assure, and its children represent components identified as its direct dependencies; each component can have dependencies as well. The SAIN architecture updates the assurance graph when services are modified or when the network conditions change.

When a service is degraded, the SAIN architecture will highlight, to the best of its knowledge, where in the assurance service graph to look, as opposed to going hop by hop to troubleshoot the issue. Not only can this architecture help to correlate service degradation with network root cause/symptoms, but it can deduce from the assurance graph the number and type of services impacted by a component degradation/failure. This added value informs the operational team where to focus its attention for maximum return. Indeed, the operational team should focus his priority on the degrading/failing components impacting the highest number customers, especially the ones with the SLA contracts involving penalties in case of failure.

This architecture provides the building blocks to assure both physical and virtual entities and is flexible with respect to services and subservices, of (distributed) graphs, and of components (Section 3.8).

### 3. Architecture

The goal of SAIN is to assure that service instances are operating correctly and if not, to pinpoint what is wrong. More precisely, SAIN computes a score for each service instance and outputs symptoms explaining that score, especially why the score is not maximal. The score augmented with the symptoms is called the health status.

The SAIN architecture is a generic architecture, applicable to multiple environments. Obviously wireline but also wireless, but also different domains such as 5G, NFV domain with a virtual infrastructure manager (VIM), etc. And as already noted, for physical or virtual devices, as well as virtual functions. Thanks to the distributed graph design principle, graphs from different environments/orchestrator can be combined together.

As an example of a service, let us consider a point-to-point L2VPN connection (i.e., pseudowire). Such a service would take as parameters the two ends of the connection (device, interface or subinterface, and address of the other end) and configure both devices (and maybe more) so that a L2VPN connection is established between the two devices. Examples of symptoms might be "Interface has high error rate" or "Interface flapping", or "Device almost out of memory".

To compute the health status of such a service, the service definition is decomposed into an assurance graph formed by subservices linked through dependencies. Each subservice is then turned into an expression graph that details how to fetch metrics from the devices and compute the health status of the subservice. The subservice expressions are combined according to the dependencies between the subservices in order to obtain the expression graph which computes the health status of the service.

The overall SAIN architecture is presented in Figure 1. Based on the service configuration, the SAIN orchestrator decomposes the assurance graph, to the best of its knowledge. It then sends to the SAIN agents the assurance graph along some other configuration options. The SAIN agents are responsible for building the expression graph and computing the health statuses in a distributed manner. The collector is in charge of collecting and displaying the current inferred health status of the service instances and subservices. Finally, the automation loop is closed by having the SAIN collector providing feedback to the network/service orchestrator.

In order to make agents, orchestrators and collectors from different vendors interoperable, their interface is defined as a YANG model in a companion RFC [I-D.ietf-opsawg-service-assurance-yang]. In Figure 1, the communications that are normalized by this model are tagged with a "Y". The use of these YANG modules is further explained in Section 3.6.

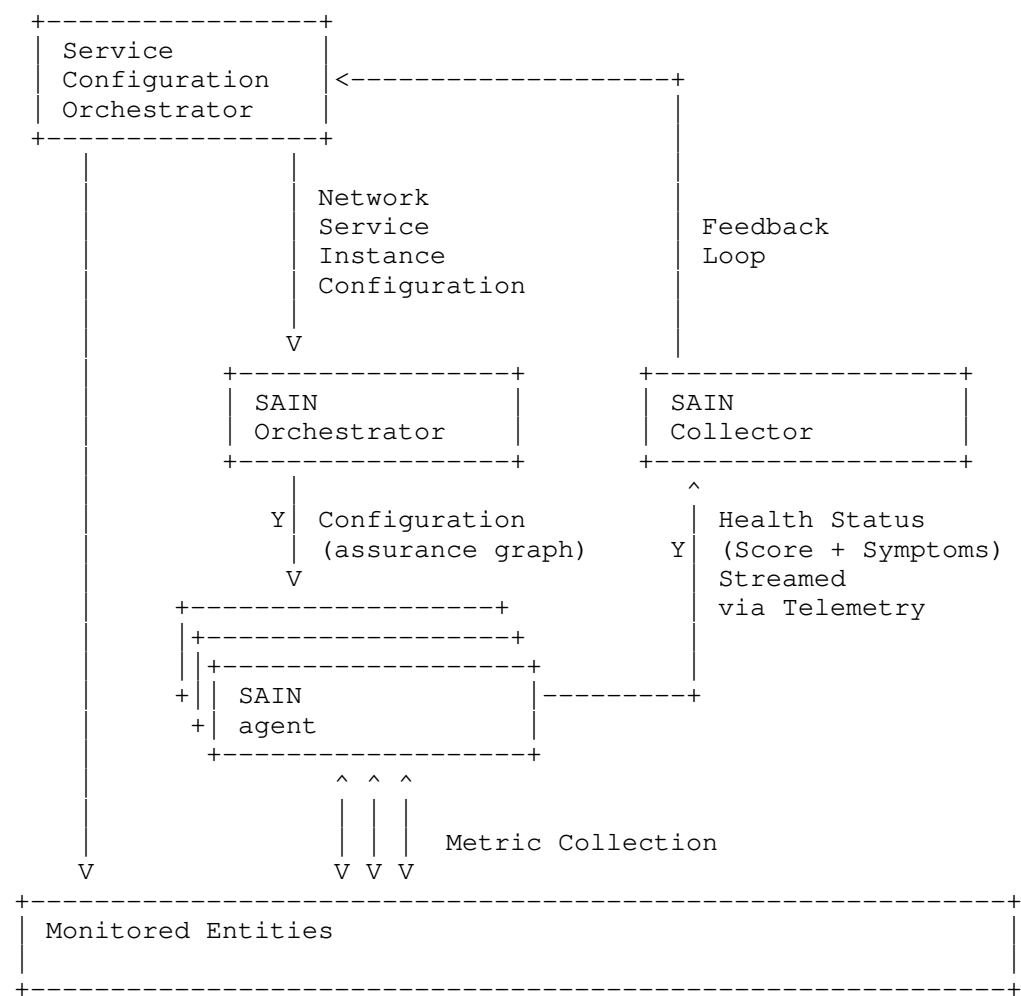


Figure 1: SAIN Architecture

In order to produce the score assigned to a service instance, the architecture performs the following tasks:

- \* Analyze the configuration pushed to the network device(s) for configuring the service instance and decide: which information is needed from the device(s), such a piece of information being called a metric, which operations to apply to the metrics for computing the health status.

- \* Stream (via telemetry [RFC8641]) operational and config metric values when possible, else continuously poll.
- \* Continuously compute the health status of the service instances, based on the metric values.

### 3.1. Inferring a Service Instance Configuration into an Assurance Graph

In order to structure the assurance of a service instance, the service instance is decomposed into so-called subservice instances. Each subservice instance focuses on a specific feature or subpart of the service.

The decomposition into subservices is an important function of this architecture, for the following reasons.

- \* The result of this decomposition provides a relational picture of a service instance, that can be represented as a graph (called assurance graph) to the operator.
- \* Subservices provide a scope for particular expertise and thereby enable contribution from external experts. For instance, the subservice dealing with the optics health should be reviewed and extended by an expert in optical interfaces.
- \* Subservices that are common to several service instances are reused for reducing the amount of computation needed.

The assurance graph of a service instance is a DAG representing the structure of the assurance case for the service instance. The nodes of this graph are service instances or subservice instances. Each edge of this graph indicates a dependency between the two nodes at its extremities: the service or subservice at the source of the edge depends on the service or subservice at the destination of the edge.

Figure 2 depicts a simplistic example of the assurance graph for a tunnel service. The node at the top is the service instance, the nodes below are its dependencies. In the example, the tunnel service instance depends on the "peer1" and "peer2" tunnel interfaces, which in turn depend on the respective physical interfaces, which finally depend on the respective "peer1" and "peer2" devices. The tunnel service instance also depends on the IP connectivity that depends on the IS-IS routing protocol.

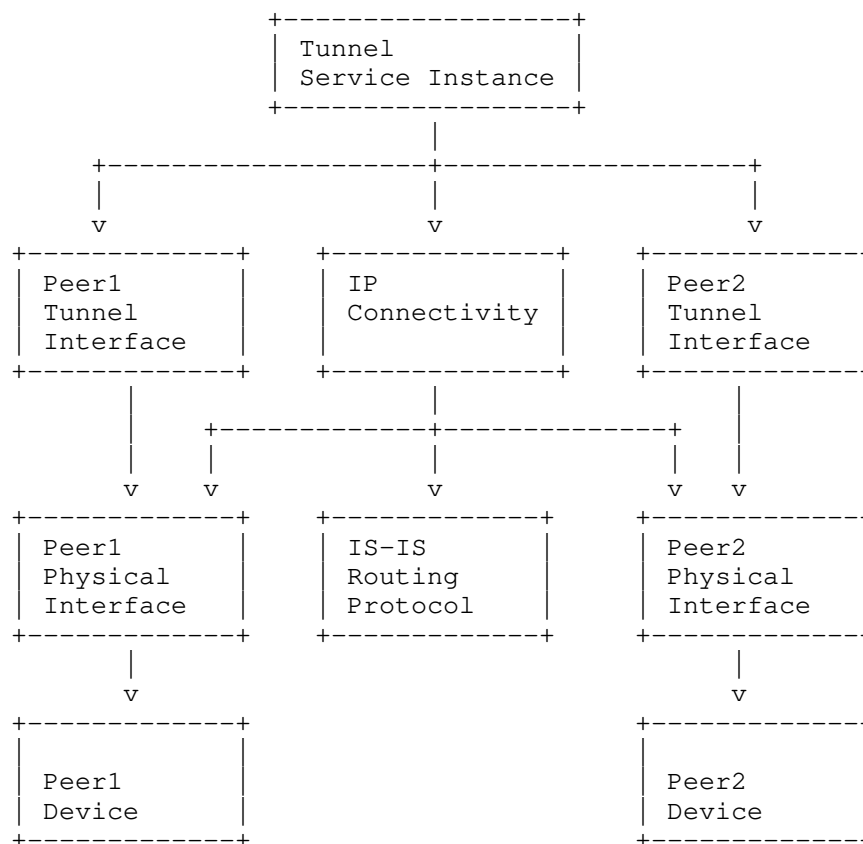


Figure 2: Assurance Graph Example

Depicting the assurance graph helps the operator to understand (and assert) the decomposition. The assurance graph shall be maintained during normal operation with addition, modification and removal of service instances. A change in the network configuration or topology shall be reflected in the assurance graph. As a first example, a change of routing protocol from IS-IS to OSPF would change the assurance graph accordingly. As a second example, assuming that ECMP is in place for the source router for that specific tunnel; in that case, multiple interfaces must now be monitored, on top of the monitoring the ECMP health itself.

### 3.1.1. Circular Dependencies

The edges of the assurance graph represent dependencies. An assurance graph is a DAG if and only if there are no circular dependencies among the subservices, and every assurance graph should avoid circular dependencies. However, in some cases, circular dependencies might appear in the assurance graph.

First, the assurance graph of a whole system is obtained by combining the assurance graph of every service running on that system. Here combining means that two subservices having the same type and the same parameters are in fact the same subservice and thus a single node in the graph. For instance, the subservice of type "device" with the only parameter (the device id) set to "PE1" will appear only once in the whole assurance graph even if several services rely on that device. Now, if two engineers design assurance graphs for two different services, and engineer A decides that an interface depends on the link it is connected to, but engineer B decides that the link depends on the interface it is connected to, then when combining the two assurance graphs, we will have a circular dependency interface -> link -> interface.

Another case possibly resulting in circular dependencies is when subservices are not properly identified. Assume that we want to assure a kubernetes cluster. If we represent the cluster by a subservice and the network service by another subservice, we will likely model that the network service depends on the cluster, because the network service is orchestrated by kubernetes, and that the cluster depends on the network service because it implements the communications. A finer decomposition might distinguish between the resources for executing containers (a part of our cluster subservice) and the communication between the containers (which could be modelled in the same way as communication between routers).

In any case, it is likely that circular dependencies will show up in the assurance graph. A first step would be to detect circular dependencies as soon as possible in the SAIN architecture. Such a detection could be carried out by the SAIN Orchestrator. Whenever a circular dependency is detected, the newly added service would not be monitored until more careful modelling or alignment between the different teams (engineer A and B) remove the circular dependency.

As more elaborate solution we could consider a graph transformation:

- \* Decompose the graph into strongly connected components.
- \* For each strongly connected component:



- Remove all edges between nodes of the strongly connected component
- Add a new "top" node for the strongly connected component
- For each edge pointing to a node in the strongly connected component, change the destination to the "top" node
- Add a dependency from the top node to every node in the strongly connected component.

Such an algorithm would include all symptoms detected by any subservice in one of the strongly component and make it available to any subservice that depends on it. Figure 3 shows an example of such a transformation. On the left-hand side, the nodes c, d, e and f form a strongly connected component. The status of a should depend on the status of c, d, e, f, g, and h, but this is hard to compute because of the circular dependency. On the right hand-side, a depends on all this nodes as well, but there the circular dependency has been removed.

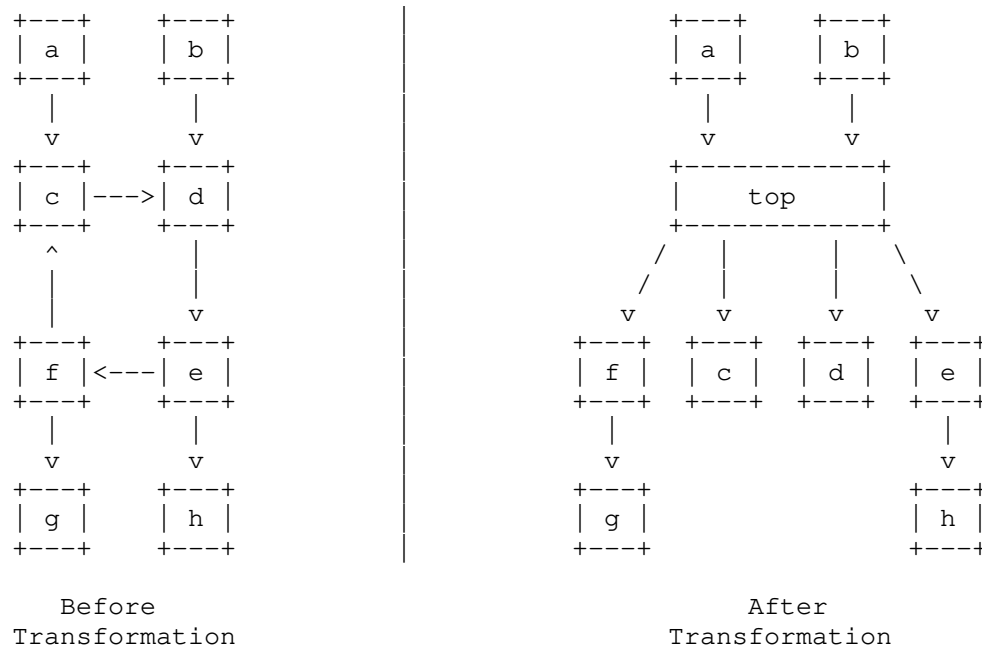


Figure 3: Graph transformation

We consider a concrete example to illustrate this transformation. Let's assume that Engineer A is building an assurance graph dealing with IS-IS and Engineer B is building an assurance graph dealing with OSPF. The graph from Engineer A could contain the following:

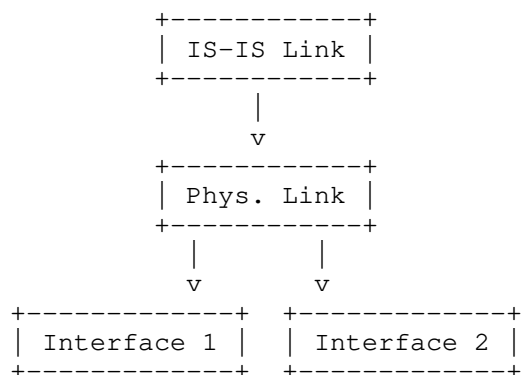


Figure 4: Fragment of assurance graph from Engineer A

The graph from Engineer B could contain the following:

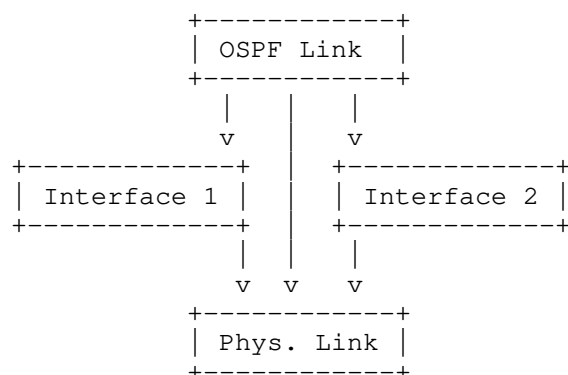


Figure 5: Fragment of assurance graph from Engineer B

Each Interface subservice and the Physical Link subservice are common to both fragments above. Each of these subservice appears only once in the graph merging the two fragments. Dependencies from both fragments are included in the merged graph, resulting in a circular dependency:

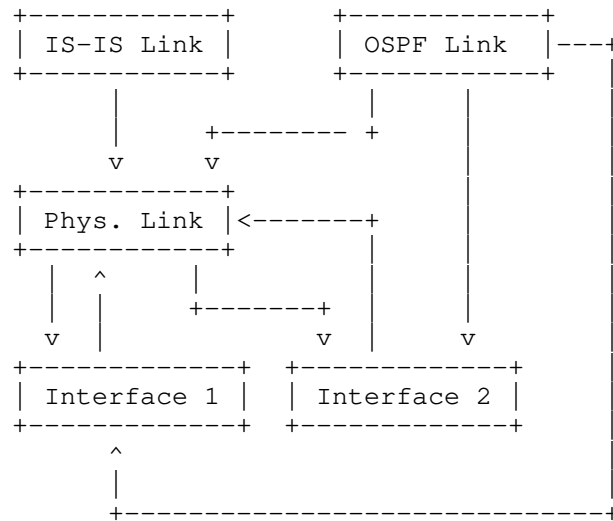


Figure 6: Merging graphs from A and B

The solution presented above would result in graph looking as follows, where a new "empty" node is included. Using that transformation, all dependencies are indirectly satisfied for the nodes outside the circular dependency, in the sense that both IS-IS and OSPF links have indirect dependencies to the two interfaces and the link. However, the dependencies between the link and the interfaces are lost as they were causing the circular dependency.

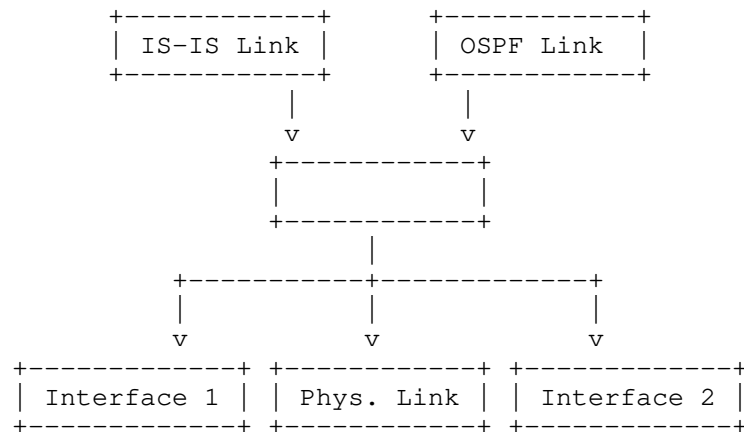


Figure 7: Removing circular dependencies after merging graphs  
from A and B

### 3.2. Intent and Assurance Graph

The SAIN orchestrator analyzes the configuration of a service instance to:

- \* Try to capture the intent of the service instance, i.e., what is the service instance trying to achieve.
- \* Decompose the service instance into subservices representing the network features on which the service instance relies.

The SAIN orchestrator must be able to analyze configuration from various devices and produce the assurance graph.

To schematize what a SAIN orchestrator does, assume that the configuration for a service instance touches two devices and configure on each device a virtual tunnel interface. Then:

- \* Capturing the intent would start by detecting that the service instance is actually a tunnel between the two devices, and stating that this tunnel must be functional. This is the current state of SAIN, however it does not completely capture the intent which might additionally include, for instance, the latency and bandwidth requirements of this tunnel.
- \* Decomposing the service instance into subservices would result in the assurance graph depicted in Figure 2, for instance.

In order for SAIN to be applied, the configuration necessary for each service instance should be identifiable and thus should come from a "service-aware" source. While the Figure 1 makes a distinction between the SAIN orchestrator and a different component providing the service instance configuration, in practice those two components are mostly likely combined. The internals of the orchestrator are currently out of scope of this document.

### 3.3. Subservices

A subservice corresponds to subpart or a feature of the network system that is needed for a service instance to function properly. In the context of SAIN, subservice is actually a shortcut for subservice assurance, that is the method for assuring that a subservice behaves correctly.

Subservices, just as with services, have high-level parameters that specify the type and specific instance to be assured. For example, assuring a device requires the specific `deviceId` as parameter. For example, assuring an interface requires the specific combination of `deviceId` and `interfaceId`.

A subservice is also characterized by a list of metrics to fetch and a list of computations to apply to these metrics in order to infer a health status.

### 3.4. Building the Expression Graph from the Assurance Graph

From the assurance graph is derived a so-called global computation graph. First, each subservice instance is transformed into a set of subservice expressions that take metrics and constants as input (i.e., sources of the DAG) and produce the status of the subservice, based on some heuristics. Then for each service instance, the service expressions are constructed by combining the subservice expressions of its dependencies. The way service expressions are combined depends on the dependency types (impacting or informational). Finally, the global computation graph is built by combining the service expressions. In other words, the global computation graph encodes all the operations needed to produce health statuses from the collected metrics.

Subservices shall be device independent. To justify this, let's consider the interface operational status. Depending on the device capabilities, this status can be collected by an industry-accepted YANG module (IETF, Openconfig), by a vendor-specific YANG module, or even by a MIB module. If the subservice was dependent on the mechanism to collect the operational status, then we would need multiple subservice definitions in order to support all different mechanisms. This also implies that, while waiting for all the metrics to be available via standard YANG modules, SAIN agents might have to retrieve metric values via non-standard YANG models, via MIB modules, Command Line Interface (CLI), etc., effectively implementing a normalization layer between data models and information models.

In order to keep subservices independent from metric collection method, or, expressed differently, to support multiple combinations of platforms, OSes, and even vendors, the architecture introduces the concept of "metric engine". The metric engine maps each device-independent metric used in the subservices to a list of device-specific metric implementations that precisely define how to fetch values for that metric. The mapping is parameterized by the characteristics (model, OS version, etc.) of the device from which the metrics are fetched.

### 3.5. Building the Expression from a Subservice

Additionally, to the list of metrics, each subservice defines a list of expressions to apply on the metrics in order to compute the health status of the subservice. The definition or the standardization of those expressions (also known as heuristic) is currently out of scope of this standardization.

### 3.6. Open Interfaces with YANG Modules

The interfaces between the architecture components are open thanks to the YANG modules specified in YANG Modules for Service Assurance [I-D.ietf-opsawg-service-assurance-yang]; they specify objects for assuring network services based on their decomposition into so-called subservices, according to the SAIN architecture.

This module is intended for the following use cases:

- \* Assurance graph configuration:
  - Subservices: configure a set of subservices to assure, by specifying their types and parameters.
  - Dependencies: configure the dependencies between the subservices, along with their types.
- \* Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

Some examples of YANG instances can be found in Appendix A of [I-D.ietf-opsawg-service-assurance-yang].

### 3.7. Handling Maintenance Windows

Whenever network components are under maintenance, the operator want to inhibit the emission of symptoms from those components. A typical use case is device maintenance, during which the device is not supposed to be operational. As such, symptoms related to the device health should be ignored, as well as symptoms related to the device-specific subservices, such as the interfaces, as their state changes is probably the consequence of the maintenance.

To configure network components as "under maintenance" in the SAIN architecture, the ietf-service-assurance model proposed in [I-D.ietf-opsawg-service-assurance-yang] specifies an "under-maintenance" flag per service or subservice instance. When this flag is set and only when this flag is set, the companion field "maintenance-contact" must be set to a string that identifies the

person or process who requested the maintenance. When a service or subservice is flagged as under maintenance, it may report a generic "Under Maintenance" symptom, for propagation towards subservices that depend on this specific subservice: any other symptom from this service, or by one of its impacting dependencies MUST NOT be reported.

We illustrate this mechanism on three independent examples based on the assurance graph depicted in Figure 2:

- \* Device maintenance, for instance upgrading the device OS. The operator sets the "under-maintenance" flag for the subservice "Peer1" device. This inhibits the emission of symptoms from "Peer1 Physical Interface", "Peer1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.
- \* Interface maintenance, for instance replacing a broken optic. The operator sets the "under-maintenance" flag for the subservice "Peer1 Physical Interface". This inhibits the emission of symptoms from "Peer 1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.
- \* Routing protocol maintenance, for instance modifying parameters or redistribution. The operator sets the "under-maintenance" flag for the subservice "IS-IS Routing Protocol". This inhibits the emission of symptoms from "IP connectivity" and "Tunnel Service Instance". All other subservices are unaffected.

### 3.8. Flexible Architecture

The SAIN architecture is flexible in terms of components. While the SAIN architecture in Figure 1 makes a distinction between two components, the SAIN configuration orchestrator and the SAIN orchestrator, in practice those two components are mostly likely combined. Similarly, the SAIN agents are displayed in Figure 1 as being separate components. Practically, the SAIN agents could be either independent components or directly integrated in monitored entities. A practical example is an agent in a router.

The SAIN architecture is also flexible in terms of services and subservices. Most examples in this document deal with the notion of Network Service YANG modules, with well-known service such as L2VPN or tunnels. However, the concepts of services is general enough to cross into different domains. One of them is the domain of service management on network elements, with also requires its own assurance. Examples includes a DHCP server on a Linux server, a data plane, an IPFIX export, etc. The notion of "service" is generic in this architecture. Indeed, a configured service can itself be a

subservice for someone else. Exactly like a DHCP server/ data plane/ IPFIX export can be considered as subservices for a device, exactly like an routing instance can be considered as a subservice for a L3VPN, exactly like a tunnel can considered as a subservice for an application in the cloud. Exactly like a service function can be be considered as a subservice for a service function chain [RFC7665]. The assurance graph is created to be flexible and open, regardless of the subservice types, locations, or domains.

The SAIN architecture is also flexible in terms of distributed graphs. As shown in Figure 1, our architecture comprises several agents. Each agent is responsible for handling a subgraph of the assurance graph. The collector is responsible for fetching the subgraphs from the different agents and gluing them together. As an example, in the graph from Figure 2, the subservices relative to Peer 1 might be handled by a different agent than the subservices relative to Peer 2 and the Connectivity and IS-IS subservices might be handled by yet another agent. The agents will export their partial graph and the collector will stitch them together as dependencies of the service instance.

And finally, the SAIN architecture is flexible in terms of what it monitors. Most, if not all examples, in this document refer to physical components but this is not a constrain. Indeed, the assurance of virtual components would follow the same principles and an assurance graph composed of virtualized components (or a mix of virtualized and physical ones) is well possible within this architecture.

### 3.9. Timing

The SAIN architecture requires time synchronization, with Network Time Protocol (NTP) [RFC5905] as a candidate, between all elements: monitored entities, SAIN agents, Service Configuration Orchestrator, the SAIN collector, as well as the SAIN Orchestrator. This guarantees the correlations of all symptoms in the system, correlated with the right assurance graph version.

The SAIN agent might have to remove some symptoms for specific subservice symptoms, because there are outdated and not relevant any longer, or simply because the SAIN agent needs to free up some space. Regardless of the reason, it's important for a SAIN collector (re-)connecting to a SAIN agent to understand the effect of this garbage collection. Therefore, the SAIN agent contains a YANG object specifying the date and time at which the symptoms history starts for the subservice instances.



### 3.10. New Assurance Graph Generation

The assurance graph will change along the time, because services and subservices come and go (changing the dependencies between subservices), or simply because a subservice is now under maintenance. Therefore an assurance graph version must be maintained, along with the date and time of its last generation. The date and time of a particular subservice instance (again dependencies or under maintenance) might be kept. From a client point of view, an assurance graph change is triggered by the value of the assurance-graph-version and assurance-graph-last-change YANG leaves. At that point in time, the client (collector) follows the following process:

- \* Keep the previous assurance-graph-last-change value (let's call it time T)
- \* Run through all subservice instance and process the subservice instances for which the last-change is newer than the time T
- \* Keep the new assurance-graph-last-change as the new referenced date and time

## 4. Security Considerations

The SAIN architecture helps operators to reduce the mean time to detect and mean time to repair. As such, it should not cause any security threats. However, the SAIN agents must be secure: a compromised SAIN agent could be sending wrong root causes or symptoms to the management systems.

Except for the configuration of telemetry, the agents do not need "write access" to the devices they monitor. This configuration is applied with a YANG module, whose protection is covered by Secure Shell (SSH) [RFC6242] for NETCONF or TLS [RFC8446] for RESTCONF.

The data collected by SAIN could potentially be compromising to the network or provide more insight into how the network is designed. Considering the data that SAIN requires (including CLI access in some cases), one should weigh data access concerns with the impact that reduced visibility will have on being able to rapidly identify root causes.

If a closed loop system relies on this architecture then the well known issue of those system also applies, i.e., a lying device or compromised agent could trigger partial reconfiguration of the service or network. The SAIN architecture neither augments or reduces this risk.

## 5. IANA Considerations

This document includes no request to IANA.

## 6. Contributors

\* Youssef El Fathi

\* Eric Vyncke

## 7. Open Issues

Refer to the Intent-based Networking NMRG documents (Intent Assurance, Service Intent: synonym for custom service model see [I-D.irtf-nmrg-ibn-concepts-definitions] and [I-D.irtf-nmrg-ibn-intent-classification] ).

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 8.2. Informative References

- [I-D.ietf-opsawg-service-assurance-yang] Claise, B., Quilbeuf, J., Lucente, P., Fasano, P., and T. Arumugam, "YANG Modules for Service Assurance", Work in Progress, Internet-Draft, draft-ietf-opsawg-service-assurance-yang-02, 4 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-service-assurance-yang-02.txt>>.
- [I-D.irtf-nmrg-ibn-concepts-definitions] Clemm, A., Ciavaglia, L., Granville, L. Z., and J. Tantsura, "Intent-Based Networking - Concepts and

Definitions", Work in Progress, Internet-Draft, draft-irtf-nmrg-ibn-concepts-definitions-06, 15 December 2021, <<https://www.ietf.org/archive/id/draft-irtf-nmrg-ibn-concepts-definitions-06.txt>>.

[I-D.irtf-nmrg-ibn-intent-classification]

Li, C., Havel, O., Olariu, A., Martinez-Julia, P., Nobre, J. C., and D. R. Lopez, "Intent Classification", Work in Progress, Internet-Draft, draft-irtf-nmrg-ibn-intent-classification-06, 22 February 2022, <<https://www.ietf.org/archive/id/draft-irtf-nmrg-ibn-intent-classification-06.txt>>.

[Piovesan2017]

Piovesan, A. and E. Griffor, "Reasoning About Safety and Security: The Logic of Assurance", 2017.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.

[RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.

- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8907] Dahm, T., Ota, A., Medway Gash, D.C., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <<https://www.rfc-editor.org/info/rfc8907>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.

#### Appendix A. Changes between revisions

v00 - v01

- \* Cover the feedback received during the WG call for adoption

#### Acknowledgements

The authors would like to thank Stephane Litkowski, Charles Eckel, Rob Wilton, Vladimir Vassiliev, Gustavo Alburquerque, Stefan Vallin, Eric Vyncke, and Mohamed Boucadair for their reviews and feedback.

#### Authors' Addresses

Benoit Claise  
Huawei  
Email: benoit.claise@huawei.com

Jean Quilbeuf  
Huawei  
Email: jean.quilbeuf@huawei.com

Diego R. Lopez  
Telefonica I+D  
Don Ramon de la Cruz, 82  
Madrid 28006  
Spain  
Email: diego.r.lopez@telefonica.com

Dan Voyer  
Bell Canada  
Canada  
Email: daniel.voyer@bell.ca

Thangam Arumugam  
Cisco Systems, Inc.  
Milpitas (California),  
United States of America  
Email: tarumuga@cisco.com

OPSAWG  
Internet-Draft  
Intended status: Standards Track  
Expires: 31 October 2022

B. Claise  
J. Quilbeuf  
Huawei  
P. Lucente  
NTT  
P. Fasano  
TIM S.p.A  
T. Arumugam  
Cisco Systems, Inc.  
29 April 2022

YANG Modules for Service Assurance  
draft-ietf-opsawg-service-assurance-yang-05

Abstract

This document specifies YANG modules representing assurance graphs. These graphs represent the assurance of a given service by decomposing it into atomic assurance elements called subservices. A companion RFC, Service Assurance for Intent-based Networking Architecture, presents an architecture for implementing the assurance of such services.

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

Status of This Memo

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

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

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

This Internet-Draft will expire on 31 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
2. YANG Models Overview . . . . .	3
3. Base ietf-service-assurance YANG module . . . . .	4
3.1. Tree View . . . . .	4
3.2. Concepts . . . . .	5
3.3. YANG Module . . . . .	7
4. Subservice Extension: ietf-service-assurance-device YANG module . . . . .	15
4.1. Tree View . . . . .	15
4.2. Complete Tree View . . . . .	15
4.3. Concepts . . . . .	16
4.4. YANG Module . . . . .	17
5. Subservice Extension: ietf-service-assurance-interface YANG module . . . . .	19
5.1. Tree View . . . . .	19
5.2. Complete Tree View . . . . .	19
5.3. Concepts . . . . .	20
5.4. YANG Module . . . . .	21
6. Security Considerations . . . . .	23
7. IANA Considerations . . . . .	23
7.1. The IETF XML Registry . . . . .	24
7.2. The YANG Module Names Registry . . . . .	24
8. Open Issues . . . . .	24
9. References . . . . .	24
9.1. Normative References . . . . .	24
9.2. Informative References . . . . .	25
Appendix A. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module . . . . .	26
A.1. Tree View . . . . .	26
A.2. Complete Tree View . . . . .	26
A.3. Concepts . . . . .	28
A.4. YANG Module . . . . .	28
Appendix B. Further Extensions: IP Connectivity and IS-IS subservices . . . . .	30
B.1. IP Connectivity Tree View . . . . .	30
B.2. IS-IS Tree View . . . . .	31

B.3. Global Tree View . . . . .	31
B.4. IP Connectivity YANG Module . . . . .	32
B.5. IS-IS YANG Module . . . . .	35
Appendix C. Example of YANG instances . . . . .	37
Appendix D. YANG Library for Service Assurance . . . . .	40
Appendix E. Changes between revisions . . . . .	42
Acknowledgements . . . . .	43
Authors' Addresses . . . . .	43

## 1. Introduction

The "Service Assurance for Intent-based Networking Architecture" [I-D.ietf-opsawg-service-assurance-architecture], specifies the architecture and all of its components for service assurance. This document complements the architecture by providing open interfaces between components. More specifically, the goal is to provide YANG modules for the purpose of service assurance in a format that is:

- \* machine readable
- \* vendor independent
- \* augmentable

### 1.1. 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 BCP 13 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms used in this document are defined in [I-D.ietf-opsawg-service-assurance-architecture]

## 2. YANG Models Overview

The main YANG module, `ietf-service-assurance`, defines objects for assuring network services based on their decomposition into so-called subservices. The subservices are hierarchically organised by dependencies. The subservices, along with the dependencies, constitute an assurance graph. This module should be supported by an agent, able to interact with the devices in order to produce a health status and symptoms for each subservice in the assurance graph. This module is intended for the following use cases:

- \* Assurance graph configuration:



- Subservices: configure a set of subservices to assure, by specifying their types and parameters.
  - Dependencies: configure the dependencies between the subservices, along with their type.
- \* Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

The main module represents the configuration (subservice and dependencies) and operational data (health status and symptoms) in a single tree. Other modules follow the same pattern. Thus, the modules presented in this document conform to the Network Management Datastore Architecture defined in [RFC8342].

The second YANG module, `ietf-service-assurance-device`, extends the `ietf-service-assurance` module to add support for the device subservice. Additional subservice types might be added the same way.

The third YANG module, `ietf-service-assurance-interface`, is another example that extends the `ietf-service-assurance` module. This extension adds support for the interface subservice.

We provide additional examples in the appendix. The module `example-service-assurance-device-acme` extends the `ietf-service-assurance-device` module to customize it for devices of the fictional ACME Corporation. Additional vendor-specific parameters might be added the same way. We also provide the modules `example-service-assurance-ip-connectivity` and `example-service-assurance-is-is` to completely model the example from the SAIN architecture draft [I-D.ietf-opsawg-service-assurance-architecture].

### 3. Base `ietf-service-assurance` YANG module

#### 3.1. Tree View

The following tree diagram [RFC8340] provides an overview of the `ietf-service-assurance` data model.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?                  yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service                  string
            +--rw instance-name            string
      +--ro health-score?                 union
      +--ro symptoms-history-start?        yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                        string
          +--ro health-score-weight?      uint8
          +--ro description?              string
          +--ro start-date-time            yang:date-and-time
          +--ro stop-date-time?            yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type
            | -> /subservices/subservice/type
          +--rw id                        leafref
          +--rw dependency-type?          identityref

```

### 3.2. Concepts

The ietf-service-assurance YANG model assumes an identified number of subservices, to be assured independently. A subservice is a feature or a subpart of the network system that a given service instance might depend on. Example of subservices include:

- \* **device:** whether a device is healthy, and if not, what are the symptoms. Potential symptoms are "CPU overloaded", "Out of RAM", or "Out of TCAM".
- \* **ip-connectivity:** given two IP addresses owned by two devices, what is the quality of the connection between them. Potential symptoms are "No route available" or "ECMP Imbalance".

The first example is a subservice representing a subpart of the network system, while the second is a subservice representing a feature of the network. In both cases, these subservices might depend on other subservices, for instance, the connectivity might depend on a subservice representing the routing mechanism and on a subservice representing ECMP.

The status of each subservice contains a list of symptoms. Each symptom is specified by a unique id and contains a health-score-weight (the impact to the health score incurred by this symptom), a label (text describing what the symptom is), and dates and times at which the symptom was detected and stopped being detected. While the unique id is sufficient as an unique key list, the start-date-time second key help sorting and retrieving relevant symptoms.

The relation between the health score and the health-score-weight of the currently active symptoms is not explicitly defined in this draft. The only requirement is that a non-maximal score must be explained by at least one symptom. A way to enforce that requirement is to first detect symptoms and then compute the health score based on the health-score-weight of the detected symptoms. As an example, this computation could be to sum the health-score-weight of the active symptoms, subtract that value from 100 and change the value to 0 if negative. The relation between health-score and health-score-weight is left to the implementor (of an agent [I-D.ietf-opsawg-service-assurance-architecture]). To consider for implementing this relation: the health-score is mostly for humans, the symptoms are what the closed loop automation can build on.

The assurance of a given service instance can be obtained by composing the assurance of the subservices that it depends on, via the dependency relations.

A subservice declaration MUST provide:

- \* A type: identity inheriting of the base identity for subservice,
- \* An id: string uniquely identifying the subservice among those with the same identity,
- \* One or more parameters, which should be specified in an augmenting model, as described in the next sections.

The type and id uniquely identify a given subservice. They are used to indicate the dependencies. Dependencies have types as well. Two types are specified in the model:

- \* **Impacting:** such a dependency indicates an impact on the health of the dependent,
- \* **Informational:** such a dependency might explain why the dependent has issues but does not impact its health.

To illustrate the difference between "impacting" and "informational", consider the interface subservice, representing a network interface. If the device to which the network interface belongs goes down, the network interface will transition to a down state as well. Therefore, the dependency of the interface subservice towards the device subservice is "impacting". On the other hand, a dependency towards the ecmp-load subservice, which checks that the load between ECMP remains stable throughout time, is only "informational". Indeed, services might be perfectly healthy even if the load distribution between ECMP changed. However, such an instability might be a relevant symptom for diagnosing the root cause of a problem.

Service instances **MUST** be modeled as a particular type of subservice with two parameters, a type and an instance name. The type is the name of the service defined in the network orchestrator, for instance "point-to-point-l2vpn". The instance name is the name assigned to the particular instance to be assured, for instance the name of the customer using that instance.

The "under-maintenance" and "maintenance-contact" flags inhibit the emission of symptoms for that subservice and subservices that depend on them. See Section 3.7 of [I-D.ietf-opsawg-service-assurance-architecture] for a more detailed discussion.

By specifying service instances and their dependencies in terms of subservices, one defines the whole assurance to apply for them. An assurance agent supporting this model should then produce telemetry in return with, for each subservice: a health-status indicating how healthy the subservice is and when the subservice is not healthy, a list of symptoms explaining why the subservice is not healthy.

### 3.3. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance@2022-04-07.yang"
```

```
module ietf-service-assurance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance";
  prefix sain;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
    WG List:   <mailto:opsawg@ietf.org>
    Author:    Benoit Claise <mailto:benoit.claise@huawei.com>
    Author:    Jean Quilbeuf <mailto:jean.quilbeu@huawei.com>";
  description
    "This module defines objects for assuring network services based on
    their decomposition into so-called subservices, according to the
    SAIN (Service Assurance for Intent-based Networking) architecture.

    The subservices hierarchically organised by dependencies constitute
    an assurance graph. This module should be supported by an assurance
    agent, able to interact with the devices in order to produce a
    health status and symptoms for each subservice in the assurance
    graph.

    This module is intended for the following use cases:
    * Assurance graph configuration:
      - subservices: configure a set of subservices to assure, by
        specifying their types and parameters.
      - dependencies: configure the dependencies between the
        subservices, along with their type.
    * Assurance telemetry: export the health status of the subservices,
      along with the observed symptoms.

    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 BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.

    Copyright (c) 2022 IETF Trust and the persons identified as
    authors of the code. All rights reserved.
```

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices. ";

```
revision 2022-04-07 {
  description
    "Shorten prefix. Fix copyright.
     Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2022-01-04 {
  description
    "Explicitely model a missing value";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Made service-instance parameters mandatory.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity subservice-idty {
  description
    "Root identity for all subservice types.";
}

identity service-instance-idty {
  base subservice-idty;
```

```
    description
      "Identity representing a service instance.";
  }

  identity dependency-type {
    description
      "Base identity for representing dependency types.";
  }

  identity informational-dependency {
    base dependency-type;
    description
      "Indicates that symptoms of the dependency might be of interest
       for the dependent, but the status of the dependency should not
       have any impact on the dependent.";
  }

  identity impacting-dependency {
    base dependency-type;
    description
      "Indicates that the status of the dependency directly impacts the
       status of the dependent.";
  }

  grouping symptom {
    description
      "Contains the list of symptoms for a specific subservice.";
    leaf id {
      type string;
      description
        "A unique identifier for the symptom.";
    }
    leaf health-score-weight {
      type uint8 {
        range "0 .. 100";
      }
      description
        "The weight to the health score incurred by this symptom. The
         higher the value, the more of an impact this symptom has. If a
         subservice health score is not 100, there must be at least one
         symptom with a health score weight larger than 0.";
    }
    leaf description {
      type string;
      description
        "Description of the symptom, i.e. text describing what the
         symptom is, to be computer-consumable and be displayed on a
         human interface. ";
    }
  }
```

```
    }
    leaf start-date-time {
        type yang:date-and-time;
        description
            "Date and time at which the symptom was detected.";
    }
    leaf stop-date-time {
        type yang:date-and-time;
        description
            "Date and time at which the symptom stopped being detected.";
    }
}

grouping subservice-dependency {
    description
        "Represent a dependency to another subservice.";
    leaf type {
        type leafref {
            path "/subservices/subservice/type";
        }
        description
            "The type of the subservice to refer to (e.g. device).";
    }
    leaf id {
        type leafref {
            path "/subservices/subservice[type=current()/../type]/id";
        }
        description
            "The identifier of the subservice to refer to.";
    }
    leaf dependency-type {
        type identityref {
            base dependency-type;
        }
        description
            "Represents the type of dependency (i.e. informational,
            impacting).";
    }
    // Augment here to add parameters specific to a new dependency-type.
    // For instance, a specific dependency type could keep symptom
    // whose health-score-weight is larger than a given value.
}

leaf assurance-graph-version {
    type yang:counter64;
    config false;
    mandatory true;
    description
```



```
    "The assurance graph version, which increases by 1 for each new
    version, after the changes (dependencies and/or maintenance
    windows parameters) are applied to the subservice(s).";
}
leaf assurance-graph-last-change {
    type yang:date-and-time;
    config false;
    mandatory true;
    description
        "Date and time at which the assurance graph last changed after the
        changes (dependencies and/or maintenance windows parameters) are
        applied to the subservice(s). These date and time must be more
        recent or equal compared to the more recent value of any changed
        subservices last-change";
}
container subservices {
    description
        "Root container for the subservices.";
    list subservice {
        key "type id";
        description
            "List of subservice configured.";
        leaf type {
            type identityref {
                base subservice-idty;
            }
            description
                "Type of the subservice, for instance device or interface.";
        }
        leaf id {
            type string;
            description
                "Unique identifier of the subservice instance, for each
                type.";
        }
        leaf last-change {
            type yang:date-and-time;
            config false;
            description
                "Date and time at which the assurance graph for this
                subservice instance last changed, i.e. dependencies and/or
                maintenance windows parameters.";
        }
        leaf label {
            type string;
            config false;
            description
                "Label of the subservice, i.e. text describing what the
```

```
    subservice is to be displayed on a human interface.";
}
leaf under-maintenance {
    type boolean;
    default "false";
    description
        "An optional flag indicating whether this particular
        subservice is under maintenance. Under this circumstance, the
        subservice symptoms and the symptoms of its dependencies in
        the assurance graph should not be taken into account.
        Instead, the subservice should send a 'Under Maintenance'
        single symptom.

        The operator changing the under-maintenance value must set
        the maintenance-contact variable.

        When the subservice is not under maintenance any longer, the
        under-maintenance flag must return to its default value and
        the under-maintenance-owner variable deleted.";
}
leaf maintenance-contact {
    when "../under-maintenance = 'true'";
    type string;
    mandatory true;
    description
        "A string used to model an administratively assigned name of
        the resource that changed the under-maintenance value to
        'true'.

        It is suggested that this name contain one or more of the
        following: IP address, management station name,
        network manager's name, location, or phone number. In some
        cases the agent itself will be the owner of an entry. In
        these cases, this string shall be set to a string starting
        with 'monitor'.";
}
choice parameter {
    description
        "Specify the required parameters per subservice type.";
    container service-instance-parameter {
        when "derived-from-or-self(..type,
            'sain:service-instance-idty')";
        description
            "Specify the parameters of a service instance.";
        leaf service {
            type string;
            mandatory true;
            description
```

```
        "Name of the service.";
    }
    leaf instance-name {
        type string;
        mandatory true;
        description
            "Name of the instance for that service.";
    }
}
// Other modules can augment their own cases into here
}
leaf health-score {
    type union {
        type uint8 {
            range "0 .. 100";
        }
        type enumeration {
            enum missing {
                value -1;
                description
                    "Explicitly represent the fact that the health score is
                    missing. This could be used when metrics crucial to
                    establish the health score are not collected anymore.";
            }
        }
    }
}
config false;
description
    "Score value of the subservice health. A value of 100 means
    that subservice is healthy. A value of 0 means that the
    subservice is broken. A value between 0 and 100 means that
    the subservice is degraded.";
}
leaf symptoms-history-start {
    type yang:date-and-time;
    config false;
    description
        "Date and time at which the symptoms history starts for this
        subservice instance, either because the subservice instance
        started at that date and time or because the symptoms before
        that were removed due to a garbage collection process.";
}
container symptoms {
    description
        "Symptoms for the subservice.";
    list symptom {
        key "start-date-time id";
        config false;
    }
}
```

```
        description
            "List of symptoms the subservice. While the start-date-time
            key is not necessary per se, this would get the entries
            sorted by start-date-time for easy consumption.";
        uses symptom;
    }
}
container dependencies {
    description
        "configure the dependencies between the subservices, along
        with their types.";
    list dependency {
        key "type id";
        description
            "List of soft dependencies of the subservice.";
        uses subservice-dependency;
    }
}
}
```

<CODE ENDS>

#### 4. Subservice Extension: ietf-service-assurance-device YANG module

##### 4.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance-device data model.

module: ietf-service-assurance-device

```
augment /sain:subservices/sain:subservice/sain:parameter:
  +--rw parameters
    +--rw device      string
```

##### 4.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance and ietf-service-assurance-device data models.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?                  yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service                  string
            +--rw instance-name            string
        +--:(sain-device:parameters)
          +--rw sain-device:parameters
            +--rw sain-device:device      string
      +--ro health-score?                  union
      +--ro symptoms-history-start?        yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                          string
          +--ro health-score-weight?        uint8
          +--ro description?                 string
          +--ro start-date-time              yang:date-and-time
          +--ro stop-date-time?              yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type
            | -> /subservices/subservice/type
          +--rw id                          leafref
          +--rw dependency-type?            identityref

```

#### 4.3. Concepts

As the number of subservices will grow over time, the YANG module is designed to be extensible. A new subservice type requires the precise specifications of its type and expected parameters. Let us illustrate the example of the new device subservice type. As the name implies, it monitors and reports the device health, along with some symptoms in case of degradation.

For our device subservice definition, the new identity device-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of a device.

The typical parameter for the configuration of the device subservice is the name of the device that we want to assure. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the device-idty subservice type, this new parameter is specified.

#### 4.4. YANG Module

<CODE BEGINS> file "ietf-service-assurance-device@2022-04-07.yang"

```
module ietf-service-assurance-device {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device";
  prefix sain-device;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
    WG List:  <mailto:opsawg@ietf.org>
    Author:   Benoit Claise  <mailto:benoit.claise@huawei.com>
    Author:   Jean Quilbeuf  <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module extends the ietf-service-assurance module to add
    support for the device subservice.
```

Checks whether a network device is healthy.

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 BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

Copyright (c) 2022 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices. ";

```
revision 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Renamed the container for parameters.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity device-idty {
  base sain:subservice-idty;
  description
    "Network Device is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'device-idty')";
  description
    "Specify the required parameters for a new subservice type";
  container parameters {
    description
```

```
        "Specify the required parameters for the device-idty
        subservice type";
    leaf device {
        type string;
        mandatory true;
        description
            "The device to monitor.";
    }
}
}
```

<CODE ENDS>

## 5. Subservice Extension: ietf-service-assurance-interface YANG module

### 5.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance-interface data model.

```
module: ietf-service-assurance-interface

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw device      string
      +--rw interface   string
```

### 5.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, and ietf-service-assurance-interface data models.



```

module: ietf-service-assurance
+--ro assurance-graph-version          yang:counter64
+--ro assurance-graph-last-change      yang:date-and-time
+--rw subservices
  +--rw subservice* [type id]
    +--rw type                          identityref
    +--rw id                            string
    +--ro last-change?                  yang:date-and-time
    +--ro label?                        string
    +--rw under-maintenance?            boolean
    +--rw maintenance-contact           string
    +--rw (parameter)?
      +--:(service-instance-parameter)
        +--rw service-instance-parameter
          +--rw service                  string
          +--rw instance-name            string
      +--:(sain-interface:parameters)
        +--rw sain-interface:parameters
          +--rw sain-interface:device    string
          +--rw sain-interface:interface string
      +--:(sain-device:parameters)
        +--rw sain-device:parameters
          +--rw sain-device:device        string
    +--ro health-score?                 union
    +--ro symptoms-history-start?        yang:date-and-time
+--rw symptoms
  +--ro symptom* [start-date-time id]
    +--ro id                            string
    +--ro health-score-weight?           uint8
    +--ro description?                   string
    +--ro start-date-time                yang:date-and-time
    +--ro stop-date-time?                yang:date-and-time
+--rw dependencies
  +--rw dependency* [type id]
    +--rw type
      | -> /subservices/subservice/type
    +--rw id                            leafref
    +--rw dependency-type?               identityref

```

### 5.3. Concepts

For our interface subservice definition, the new interface-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of an interface.

The typical parameters for the configuration of the interface subservice are the name of the device and, on that specific device, a specific interface. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the interface-idty subservice type, those two new parameters are specified.

#### 5.4. YANG Module

<CODE BEGINS> file "ietf-service-assurance-interface@2022-04-07.yang"

```
module ietf-service-assurance-interface {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface";
  prefix sain-interface;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
    WG List: <mailto:opsawg@ietf.org>
    Author: Benoit Claise <mailto:benoit.claise@huawei.com>
    Author: Jean Quilbeuf <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module extends the ietf-service-assurance module to add
    support for the interface subservice.
```

Checks whether an interface is healthy.

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 BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

Copyright (c) 2022 IETF Trust and the persons identified as authors of the code. All rights reserved.  
Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions

Relating to IETF Documents  
(<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices. ";

```
revision 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Regroup parameters in a container.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity interface-idty {
  base sain:subservice-idty;
  description
    "Checks whether an interface is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'interface-idty')";
  description
    "Specify the required parameters for the interface-idty
    subservice type";
  container parameters {
    description
      "Required parameters for the interface-idty subservice
      type";
    leaf device {
      type string;
      mandatory true;
      description
        "Device supporting the interface.";
    }
  }
}
```

```
    leaf interface {  
        type string;  
        mandatory true;  
        description  
            "Name of the interface.";  
    }  
}  
}
```

<CODE ENDS>

## 6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/ creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- \* /subservices/subservice/type
- \* /subservices/subservice/id
- \* /subservices/subservice/under-maintenance
- \* /subservices/subservice/maintenance-contact

## 7. IANA Considerations

### 7.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

### 7.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

name: ietf-service-assurance  
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance  
prefix: sain  
reference: RFC XXXX

name: ietf-service-assurance-device  
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device  
prefix: sain-device  
reference: RFC XXXX

name: ietf-service-assurance-interface  
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface  
prefix: sain-interface  
reference: RFC XXXX

## 8. Open Issues

-None

## 9. References

### 9.1. Normative References

- [I-D.ietf-opsawg-service-assurance-architecture]  
Claise, B., Quilbeuf, J., Lopez, D. R., Voyer, D., and T. Arumugam, "Service Assurance for Intent-based Networking Architecture", Work in Progress, Internet-Draft, draft-ietf-opsawg-service-assurance-architecture-03, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-service-assurance-architecture-03.txt>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

## 9.2. 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, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<https://www.rfc-editor.org/info/rfc7895>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

#### Appendix A. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module

##### A.1. Tree View

The following tree diagram [RFC8340] provides an overview of the example-service-assurance-device-acme data model.

```
module: example-service-assurance-device-acme

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw device                string
      +--rw acme-specific-parameter  string
```

##### A.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, and example-service-assurance-device-acme data models.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?
      |      yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
      |
      |   +--:(service-instance-parameter)
      |   |   +--rw service-instance-parameter
      |   |   |   +--rw service            string
      |   |   |   +--rw instance-name      string
      |   +--:(sain-device:parameters)
      |   |   +--rw sain-device:parameters
      |   |   |   +--rw sain-device:device  string
      |   +--:(example-device-acme:parameters)
      |   |   +--rw example-device-acme:parameters
      |   |   |   +--rw example-device-acme:device
      |   |   |   |   string
      |   |   |   +--rw example-device-acme:acme-specific-parameter
      |   |   |   |   string
      |   +--:(sain-interface:parameters)
      |   |   +--rw sain-interface:parameters
      |   |   |   +--rw sain-interface:device      string
      |   |   |   +--rw sain-interface:interface   string
      +--ro health-score?                  union
      +--ro symptoms-history-start?
      |      yang:date-and-time
      +--rw symptoms
      |
      |   +--ro symptom* [start-date-time id]
      |   |   +--ro id                      string
      |   |   +--ro health-score-weight?   uint8
      |   |   +--ro description?           string
      |   |   +--ro start-date-time        yang:date-and-time
      |   |   +--ro stop-date-time?        yang:date-and-time
      +--rw dependencies
      |
      |   +--rw dependency* [type id]
      |   |   +--rw type
      |   |   |   -> /subservices/subservice/type
      |   |   +--rw id                      leafref
      |   |   +--rw dependency-type?       identityref

```



### A.3. Concepts

Under some circumstances, vendor-specific subservice types might be required. As an example of this vendor-specific implementation, this section shows how to augment the `ietf-service-assurance-device` module to add custom support for the device subservice, specific to the ACME Corporation. The specific version adds a new parameter, named `acme-specific-parameter`.

### A.4. YANG Module

```
module example-service-assurance-device-acme {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-device-acme";
  prefix example-device-acme;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }
  import ietf-service-assurance-device {
    prefix sain-device;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
    WG List:  <mailto:opsawg@ietf.org>
    Author:   Benoit Claise <mailto:benoit.claise@huawei.com>
    Author:   Jean Quilbeuf <mailto:jean.quilbeuf@huawei.com>";
  description
    "This module extends the ietf-service-assurance-device module to
    add specific support for devices of ACME Corporation.

    ACME Network Device is healthy.

    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 BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.
```

Copyright (c) 2022 IETF Trust and the persons identified as

authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices. ";

```
revision 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Renamed the parameters container.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity device-acme-idty {
  base sain-device:device-idty;
  description
    "Network Device is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'device-acme-idty')";
  description
```

```

    "Specify the required parameters for a new subservice type";
  container parameters {
    description
      "Specify the required parameters for the device-acme-idty
      subservice type";
    leaf device {
      type string;
      mandatory true;
      description
        "The device to monitor.";
    }
    leaf acme-specific-parameter {
      type string;
      mandatory true;
      description
        "The ACME Corporation sepcific parameter.";
    }
  }
}
}
}

```

## Appendix B. Further Extensions: IP Connectivity and IS-IS subservices

In this section, we provide two additional YANG models to completely cover the example from Figure 2 in [I-D.ietf-opsawg-service-assurance-architecture]. The complete normalization of these modules is to be done in future work.

### B.1. IP Connectivity Tree View

That subservice represents the unicast connectivity between two IP addresses located on to different devices. Such a subservice could report symptoms such as "No route found". The following tree diagram [RFC8340] provides an overview of the example-service-assurance-ip-connectivity data model.

```

module: example-service-assurance-ip-connectivity

augment /sain:subservices/sain:subservice/sain:parameter:
  +--rw parameters
    +--rw device1      string
    +--rw address1     inet:ip-address
    +--rw device2      string
    +--rw address2     inet:ip-address

```

To specify the connectivity that we are interested in, we specify two IP addresses and two devices. The subservice assures that the connectivity between IP address 1 on device 1 and IP address 2 on device 2 is healthy.

### B.2. IS-IS Tree View

The following tree diagram [RFC8340] provides an overview of the example-service-assurance-is-is data model.

```
module: example-service-assurance-is-is
```

```
  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw instance-name    string
```

The parameter of this subservice is the name of the IS-IS instance to assure.

### B.3. Global Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, example-service-assurance-device-acme, example-service-assurance-ip-connectivity and example-service-assurance-is-is data models.

```
module: ietf-service-assurance
  +--ro assurance-graph-version      yang:counter64
  +--ro assurance-graph-last-change  yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                                identityref
      +--rw id                                string
      +--ro last-change?
        | yang:date-and-time
      +--ro label?                            string
      +--rw under-maintenance?              boolean
      +--rw maintenance-contact              string
      +--rw (parameter)?
        | +--:(service-instance-parameter)
        | | +--rw service-instance-parameter
        | | | +--rw service                string
        | | | +--rw instance-name          string
        | | +--:(example-ip-connectivity:parameters)
        | | | +--rw example-ip-connectivity:parameters
        | | | | +--rw example-ip-connectivity:device1    string
        | | | | +--rw example-ip-connectivity:address1
```

```

|         |         |         inet:ip-address
|         |         +---rw example-ip-connectivity:device2      string
|         |         +---rw example-ip-connectivity:address2
|         |         |         inet:ip-address
+---: (example-is-is:parameters)
|         |         +---rw example-is-is:parameters
|         |         |         +---rw example-is-is:instance-name      string
+---: (sain-device:parameters)
|         |         +---rw sain-device:parameters
|         |         |         +---rw sain-device:device      string
+---: (example-device-acme:parameters)
|         |         +---rw example-device-acme:parameters
|         |         |         +---rw example-device-acme:device
|         |         |         |         string
|         |         |         +---rw example-device-acme:acme-specific-parameter
|         |         |         |         string
+---: (sain-interface:parameters)
|         |         +---rw sain-interface:parameters
|         |         |         +---rw sain-interface:device      string
|         |         |         +---rw sain-interface:interface    string
+---ro health-score?                                     union
+---ro symptoms-history-start?
|         |         yang:date-and-time
+---rw symptoms
|         |         +---ro symptom* [start-date-time id]
|         |         |         +---ro id                        string
|         |         |         +---ro health-score-weight?    uint8
|         |         |         +---ro description?            string
|         |         |         +---ro start-date-time          yang:date-and-time
|         |         |         +---ro stop-date-time?          yang:date-and-time
+---rw dependencies
|         |         +---rw dependency* [type id]
|         |         |         +---rw type
|         |         |         |         -> /subservices/subservice/type
|         |         |         +---rw id                        leafref
|         |         |         +---rw dependency-type?         identityref

```

#### B.4. IP Connectivity YANG Module

```

module example-service-assurance-ip-connectivity {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-ip-connectivity";
  prefix example-ip-connectivity;

  import ietf-inet-types {
    prefix inet;
    reference

```

```
    "RFC 6991: Common YANG Data Types";
}
import ietf-service-assurance {
    prefix sain;
    reference
        "RFC xxxx: YANG Modules for Service Assurance";
}

organization
    "IETF OPSAWG Working Group";
contact
    "WG Web:    <https://datatracker.ietf.org/wg/opsawg/>
    WG List:    <mailto:opsawg@ietf.org>
    Author:     Benoit Claise <mailto:benoit.claise@huawei.com>
    Author:     Jean Quilbeuf <mailto:jean.quilbeuf@huawei.com>";
description
    "This example module extends the ietf-service-assurance module to
    add support for the subservice ip-connectivity.

    Checks whether the ip connectivity between two ip addresses
    belonging to two network devices is healthy.

    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 BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.

    Copyright (c) 2022 IETF Trust and the persons identified as
    authors of the code. All rights reserved.
    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Revised BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see the
    RFC itself for full legal notices.  ";

revision 2022-04-07 {
    description
        "Fix mandatory in augment error by moving when clause.
        Shorten prefix. Fix module description.
        Fix module description";
    reference
        "RFC xxxx: YANG Modules for Service Assurance";
```

```
}
revision 2021-06-28 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity ip-connectivity-idty {
  base sain:subservice-idty;
  description
    "Checks connectivity between two IP addresses.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'ip-connectivity-idty')";
  description
    "Specify the required parameters for the ip-connectivity-idty
    subservice type";
  container parameters {
    description
      "Required parameters for the ip-connectivity-idty
      subservice type";
    leaf device1 {
      type string;
      mandatory true;
      description
        "Device at the first end of the connection.";
    }
    leaf address1 {
      type inet:ip-address;
      mandatory true;
      description
        "Address at the first end of the connection.";
    }
    leaf device2 {
      type string;
      mandatory true;
      description
        "Device at the second end of the connection.";
    }
    leaf address2 {
      type inet:ip-address;
      mandatory true;
      description
        "Address at the second end of the connection.";
    }
  }
}
```

```
}  
}
```

#### B.5. IS-IS YANG Module

```
module example-service-assurance-is-is {  
  yang-version 1.1;  
  namespace "urn:example:example-service-assurance-is-is";  
  prefix example-is-is;  
  
  import ietf-service-assurance {  
    prefix sain;  
    reference  
      "RFC xxxx: YANG Modules for Service Assurance";  
  }  
  
  organization  
    "IETF OPSAWG Working Group";  
  contact  
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>  
    WG List:  <mailto:opsawg@ietf.org>  
    Author:   Benoit Claise <mailto:benoit.claise@huawei.com>  
    Author:   Jean Quilbeuf <mailto:jean.quilbeuf@huawei.com>";  
  description  
    "This module extends the ietf-service-assurance module to  
    add support for the subservice is-is.  
  
    Checks whether an IS-IS instance is healthy.  
  
    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 BCP 14 (RFC 2119)  
    (RFC 8174) when, and only when, they appear in all  
    capitals, as shown here.  
  
    Copyright (c) 2022 IETF Trust and the persons identified as  
    authors of the code.  All rights reserved.  
  
    Redistribution and use in source and binary forms, with or  
    without modification, is permitted pursuant to, and subject  
    to the license terms contained in, the Revised BSD License  
    set forth in Section 4.c of the IETF Trust's Legal Provisions  
    Relating to IETF Documents  
    (https://trustee.ietf.org/license-info).  
    This version of this YANG module is part of RFC XXXX; see the  
    RFC itself for full legal notices.  ";
```



```
revision 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity is-is-idty {
  base sain:subservice-idty;
  description
    "Health of IS-IS routing protocol.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'is-is-idty')";
  description
    "Specify the required parameters for a new subservice
    type";
  container parameters {
    description
      "Specify the required parameters for the IS-IS subservice
      type";
    leaf instance-name {
      type string;
      mandatory true;
      description
        "The instance to monitor.";
    }
  }
}
}
```

## Appendix C. Example of YANG instances

This section contains examples of YANG instances that conform to the YANG modules. The validity of these data instances has been checked using yangson (<https://yangson.labs.nic.cz/>). Yangson requires a YANG library [RFC7895] to define the complete model against which the data instance must be validated. We provide in Appendix D the JSON library file, named "ietf-service-assurance-library.json", that we used for validation.

We provide below the contents of the file "example\_configuration\_instance.json" which contains the configuration data that models the Figure 2 of [I-D.ietf-opsawg-service-assurance-architecture]. The instance can be validated with yangson by using the invocation "yangson -v example\_configuration\_instance.json ietf-service-assurance-library.json", assuming all the files (YANG and JSON) defined in this draft reside in the current folder.

```
{
  "ietf-service-assurance:subservices": {
    "subservice": [
      {
        "type": "service-instance-idty",
        "id": "simple-tunnel/example",
        "service-instance-parameter": {
          "service": "simple-tunnel",
          "instance-name": "example"
        },
        "dependencies": {
          "dependency": [
            {
              "type": "ietf-service-assurance-interface:interface-idty",
              "id": "interface/peer1/tunnel0",
              "dependency-type": "impacting-dependency"
            },
            {
              "type": "ietf-service-assurance-interface:interface-idty",
              "id": "interface/peer2/tunnel9",
              "dependency-type": "impacting-dependency"
            },
            {
              "type":
                "example-service-assurance-ip-connectivity:ip-connectivity-idty",
              "id": "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
              "dependency-type": "impacting-dependency"
            }
          ]
        }
      }
    ]
  }
}
```

```
    }
  },
  {
    "type":
    "example-service-assurance-ip-connectivity:ip-connectivity-idty",
    "id": "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
    "example-service-assurance-ip-connectivity:parameters": {
      "device1": "Peer1",
      "address1": "2001:db8::1",
      "device2": "Peer2",
      "address2": "2001:db8::2"
    },
    "dependencies": {
      "dependency": [
        {
          "type": "ietf-service-assurance-interface:interface-idty",
          "id": "interface/peer1/physical0",
          "dependency-type": "impacting-dependency"
        },
        {
          "type": "ietf-service-assurance-interface:interface-idty",
          "id": "interface/peer2/physical5",
          "dependency-type": "impacting-dependency"
        },
        {
          "type": "example-service-assurance-is-is:is-is-idty",
          "id": "is-is/instance1",
          "dependency-type": "impacting-dependency"
        }
      ]
    }
  },
  {
    "type": "example-service-assurance-is-is:is-is-idty",
    "id": "is-is/instance1",
    "example-service-assurance-is-is:parameters": {
      "instance-name": "instance1"
    }
  },
  {
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer1/tunnel0",
    "ietf-service-assurance-interface:parameters": {
      "device": "Peer1",
      "interface": "tunnel0"
    },
    "dependencies": {
      "dependency": [
```

```
        {
            "type": "ietf-service-assurance-interface:interface-idty",
            "id": "interface/peer1/physical0",
            "dependency-type": "impacting-dependency"
        }
    ]
}
},
{
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer1/physical0",
    "ietf-service-assurance-interface:parameters": {
        "device": "Peer1",
        "interface": "physical0"
    },
    "dependencies": {
        "dependency": [
            {
                "type": "ietf-service-assurance-device:device-idty",
                "id": "interface/peer1",
                "dependency-type": "impacting-dependency"
            }
        ]
    }
},
{
    "type": "ietf-service-assurance-device:device-idty",
    "id": "interface/peer1",
    "ietf-service-assurance-device:parameters": {
        "device": "Peer1"
    }
},
{
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer2/tunnel9",
    "ietf-service-assurance-interface:parameters": {
        "device": "Peer2",
        "interface": "tunnel9"
    },
    "dependencies": {
        "dependency": [
            {
                "type": "ietf-service-assurance-interface:interface-idty",
                "id": "interface/peer2/physical15",
                "dependency-type": "impacting-dependency"
            }
        ]
    }
}
```

```

    },
    {
      "type": "ietf-service-assurance-interface:interface-idty",
      "id": "interface/peer2/physical5",
      "ietf-service-assurance-interface:parameters": {
        "device": "Peer2",
        "interface": "physical5"
      },
      "dependencies": {
        "dependency": [
          {
            "type": "ietf-service-assurance-device:device-idty",
            "id": "interface/peer2",
            "dependency-type": "impacting-dependency"
          }
        ]
      }
    },
  ],
}

```

#### Appendix D. YANG Library for Service Assurance

This section provides the JSON encoding of the YANG library [RFC7895] listing all modules defined in this draft and their dependencies. This library can be used to validate data instances using yangson, as explained in the previous section.

```

{
  "ietf-yang-library:modules-state": {
    "module-set-id": "ietf-service-assurance@2022-04-07",
    "module": [
      {
        "name": "ietf-service-assurance",
        "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-service-assurance",
        "revision": "2022-04-07",
        "conformance-type": "implement"
      }
    ]
  }
}

```

```
    },
    {
      "name": "ietf-service-assurance-device",
      "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "ietf-service-assurance-interface",
      "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-device-acme",
      "namespace":
        "urn:example:example-service-assurance-device-acme",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-is-is",
      "namespace": "urn:example:example-service-assurance-is-is",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-ip-connectivity",
      "namespace":
        "urn:example:example-service-assurance-ip-connectivity",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "ietf-yang-types",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
      "revision": "2021-04-14",
      "conformance-type": "import"
    },
    {
      "name": "ietf-inet-types",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
      "revision": "2021-02-22",
      "conformance-type": "import"
    }
  ]
}
```

```
}  
}
```

#### Appendix E. Changes between revisions

##### v04 - v05

- \* Remove Guidelines section
- \* Move informative parts (examples) to appendix
- \* Minor text edits and reformulations

##### v03 - v04

- \* Fix YANG errors
- \* Change is-is and ip-connectivity subservices from ietf to example.
- \* Mention that models are NMDA compliant
- \* Fix typos, reformulate for clarity

##### v02 - v03

- \* Change counter32 to counter64 to avoid resetting too frequently
- \* Explain why relation between health-score and symptom's health-score-weight is not defined and how it could be defined

##### v01 - v02

- \* Explicitly represent the fact that the health-score could not be computed (value -1)

##### v00 - v01

- \* Added needed subservice to model example from architecture draft
- \* Added guideline section for naming models
- \* Added data instance examples and validation procedure
- \* Added the "parameters" container in the interface YANG module to correct a bug.

## Acknowledgements

The authors would like to thank Jan Lindblad for his help during the design of these YANG modules. The authors would like to thank Stephane Litkowski, Charles Eckel, Mohamed Boucadair and Tom Petch for their reviews.

## Authors' Addresses

Benoit Claise  
Huawei  
Email: benoit.claise@huawei.com

Jean Quilbeuf  
Huawei  
Email: jean.quilbeuf@huawei.com

Paolo Lucente  
NTT  
Siriusdreef 70-72  
2132 Hoofddorp  
Netherlands  
Email: paolo@ntt.net

Paolo Fasano  
TIM S.p.A  
via G. Reiss Romoli, 274  
10148 Torino  
Italy  
Email: paolo2.fasano@telecomitalia.it

Thangam Arumugam  
Cisco Systems, Inc.  
Milpitas (California),  
United States  
Email: tarumuga@cisco.com



OPSAWG Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 6 November 2022

B. Wu, Ed.  
Q. Wu, Ed.  
Huawei  
M. Boucadair, Ed.  
Orange  
O. Gonzalez de Dios  
Telefonica  
B. Wen  
Comcast  
5 May 2022

A YANG Model for Network and VPN Service Performance Monitoring  
draft-ietf-opsawg-yang-vpn-service-pm-08

Abstract

The data model for network topologies defined in RFC 8345 introduces vertical layering relationships between networks that can be augmented to cover network and service topologies. This document defines a YANG module for performance monitoring (PM) of both networks and VPN services that can be used to monitor and manage network performance on the topology at higher layer or the service topology between VPN sites.

Status of This Memo

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

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

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

This Internet-Draft will expire on 6 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
2.1. Acronyms . . . . .	3
3. Network and VPN Service Performance Monitoring Model Usage .	4
3.1. Collecting Data via Pub/Sub Mechanism . . . . .	5
3.2. Collecting Data On-demand . . . . .	6
4. Description of The Data Model . . . . .	6
4.1. Layering Relationship between Multiple Layers of Topology . . . . .	6
4.2. Network Level . . . . .	9
4.3. Node Level . . . . .	9
4.4. Link and Termination Point Level . . . . .	10
5. Network and VPN Service Performance Monitoring YANG Module .	14
6. Security Considerations . . . . .	29
7. IANA Considerations . . . . .	30
8. Acknowledgements . . . . .	31
9. Contributors . . . . .	31
10. References . . . . .	31
10.1. Normative References . . . . .	31
10.2. Informative References . . . . .	34
Appendix A. Illustrative Examples . . . . .	35
A.1. VPN Performance Subscription Example . . . . .	35
A.2. Example of VPN Performance Snapshot . . . . .	37
A.3. Example of Percentile Monitoring . . . . .	39
Authors' Addresses . . . . .	39

## 1. Introduction

[RFC8969] describes a framework for automating service and network management with YANG [RFC6020] models. It defines that the performance measurement telemetry model should be tied to the services (such as a Layer 3 VPN or Layer 2 VPN) or to the network models to monitor the overall network performance and the Service Level Agreements (SLAs).

The performance of VPN services is associated with the performance changes of the underlay network that carries VPN services, such as the delay of the underlay tunnels and the packet loss status of the device interfaces. Additionally, the integration of Layer 2/Layer 3 VPN performance and network performance data enables the orchestrator to subscribe to VPN service performance in a unified manner. Therefore, this document defines a YANG module for both network and VPN service performance monitoring (PM). The module can be used to monitor and manage network performance on the topology level or the service topology between VPN sites, in particular.

This document does not introduce new metrics for network performance or mechanisms for measuring network performance, but uses the existing mechanisms and statistics to display the performance monitoring statistics at the network and service layers. All these metrics are defined as unidirectional metrics.

The YANG module defined in this document is designed as an augmentation to the network topology YANG model defined in [RFC8345] and draws on relevant YANG types defined in [RFC6991], [RFC8345], [RFC8532], and [RFC9181].

Appendix A provides a set of examples to illustrate the use of the module.

## 2. Terminology

The following terms are defined in [RFC7950] and are used in this specification:

- \* augment
- \* data model
- \* data node

The terminology for describing YANG data models is found in [RFC7950].

The tree diagrams used in this document follow the notation defined in [RFC8340].

### 2.1. Acronyms

The following acronyms are used in the document:

L2VPN    Layer 2 Virtual Private Network  
L3VPN    Layer 3 Virtual Private Network

L2NM	L2VPN Network Model
L3NM	L3VPN Network Model
MPLS	Multiprotocol Label Switching
OAM	Operations, Administration, and Maintenance
OWAMP	One-Way Active Measurement Protocol
PE	Provider Edge
PM	Performance Monitoring
SLA	Service Level Agreement
TP	Termination Point, as defined in [RFC8345] section 4.2
TWAMP	Two-Way Active Measurement Protocol
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network

### 3. Network and VPN Service Performance Monitoring Model Usage

Models are key for automating network management operations. According to [RFC8969], together with service and network models, performance measurement telemetry models are needed to monitor network performance to meet specific service requirements (typically captured in an SLA).

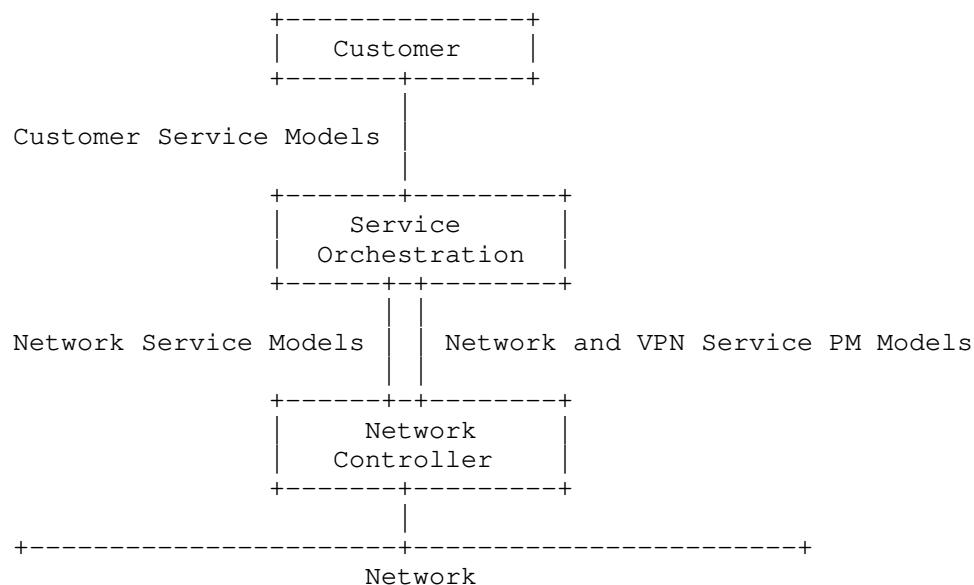


Figure 1: Reference Architecture

As shown in Figure 1, in the context of the layering model architecture described in [RFC8309], the network and VPN service performance monitoring (PM) model can be used to expose a set of

performance information to the above layer. Such information can be used by an orchestrator to subscribe to performance data. The network controller will then notify the orchestrator about corresponding parameter changes.

Before using the model, the controller needs to establish complete topology visibility of the network and VPN. For example, the controller can use network information from [RFC8345], [I-D.ietf-opsawg-sap] or VPN information from [RFC9182], [I-D.ietf-opsawg-l2nm]. Then the controller derives network or VPN level performance data by aggregating (and filtering) lower-level data collected via monitoring counters of the involved devices.

The network or VPN performance data can be based on different sources. For example, the performance monitoring data per link in the underlying network can be collected using a network performance measurement method such as One-Way Active Measurement Protocol (OWAMP) [RFC4656], Two-Way Active Measurement Protocol (TWAMP) [RFC5357], Simple Two-way Active Measurement Protocol (STAMP) [RFC8762], and Multiprotocol Label Switching (MPLS) Loss and Delay Measurement [RFC6374]. The performance monitoring information reflecting the quality of the network or VPN service (e.g., end-to-end network performance data between source node and destination node in the network or between VPN sites) can be computed and aggregated, for example, using the information from the Traffic Engineering Database (TED), [RFC7471] [RFC8570] [RFC8571] or LMAP [RFC8194].

The measurement and report intervals that are associated with these performance data usually depend on the configuration of the specific measurement method or collection method or various combinations. This document defines a network-wide measurement interval to align measurement requirements for networks or VPN services.

In addition, the amount of performance data collected from the devices can be huge. To avoid receiving a large amount of operational data of VPN instances, VPN interfaces, or tunnels, the network controller can specifically subscribe to metric-specific data using the tagging methods defined in [I-D.ietf-netmod-node-tags].

### 3.1. Collecting Data via Pub/Sub Mechanism

Some applications such as service-assurance applications, which must maintain a continuous view of operational data and state, can use the subscription model specified in [RFC8641] to subscribe to the specific network performance data or VPN service performance data they are interested in, at the data source. For example, network or VPN topology updates may be obtained through on-change notifications [RFC8641]. For dynamic PM data, various notifications can be

specified to obtain more complete data. A periodic notification [RFC8641] can be specified to obtain real-time performance data, a replay notification defined in [RFC5277] or [RFC8639] can be specified to obtain historical data, or alarm notifications [RFC8632] can be specified to get alarms for the metrics which exceed or fall below the performance threshold.

The data source can, then, use the network and VPN service assurance model defined in this document and the YANG Push model [RFC8641] to distribute specific telemetry data to target recipients.

### 3.2. Collecting Data On-demand

To obtain a snapshot of a large amount of performance data from a network topology or VPN network, service-assurance applications may retrieve information using the network and VPN service PM model through a NETCONF [RFC6241] or a RESTCONF [RFC8040] interface. For example, a specified "link-id" of a VPN can be used as a filter in a RESTCONF GET request to retrieve per-link VPN PM data.

## 4. Description of The Data Model

This document defines the YANG module, "ietf-network-vpn-pm", which is an augmentation to the "ietf-network" and "ietf-network-topology" modules.

The performance monitoring data augments the service topology as shown in Figure 2.

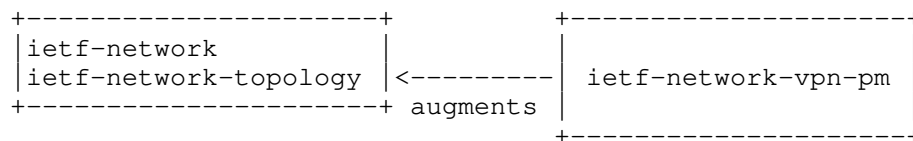


Figure 2: Module Augmentation

### 4.1. Layering Relationship between Multiple Layers of Topology

[RFC8345] defines a YANG data model for network/service topologies and inventories. The service topology described in [RFC8345] includes the virtual topology for a service layer above Layer 1 (L1), Layer 2 (L2), and Layer 3 (L3). This service topology has the generic topology elements of node, link, and terminating point. One typical example of a service topology is described in Figure 3 of [RFC8345]: two VPN service topologies instantiated over a common L3 topology. Each VPN service topology is mapped onto a subset of nodes from the common L3 topology.

Figure 3 illustrates an example of a topology that maps between the VPN service topology and an underlying network:

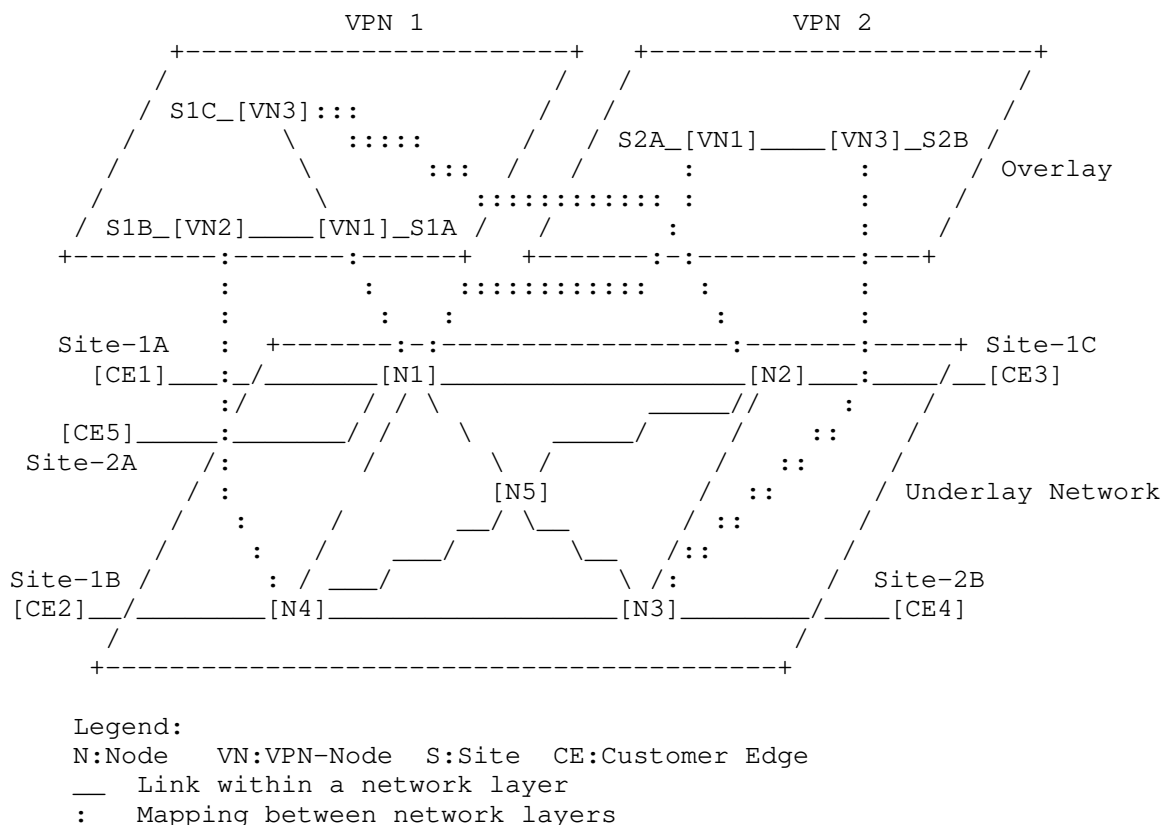


Figure 3: Example of Topology Mapping Between VPN Service Topology and Underlying Network

As shown in Figure 3, two VPN services topologies are built on top of one common underlying physical network:

VPN 1: This service topology supports hub-spoke communications for 'customer 1' connecting the customer's access at three sites: 'Site-1A', 'Site-1B', and 'Site-1C'. These sites are connected to nodes that are mapped to node 1 (N1), node 2 (N2), and node 4 (N4) in the underlying physical network. 'Site-1A' plays the role of hub while 'Site-1B' and 'Site-1C' are configured as spoke.

VPN 2: This service supports any-to-any communications for 'customer

2' connecting the customer's access at two sites: 'Site-2A' and 'Site-2B'. These sites are connected to nodes that are mapped to nodes 1 (N1) and node 3 (N3)5 in the underlying physical network. 'Site-2A' and 'Site-2B' have 'any-to-any' role.

Apart from the association between the VPN topology and the underlay topology, VPN Network PM can also provide the performance status of the underlay network and VPN services. For example, network PM can provide link PM statistics and port statistics. VPN PM can provide statistics on VPN access interfaces, the number of current VRF routes or L2VPN MAC entry of VPN nodes, and performance statistics on the logical point-to-point link between source and destination VPN nodes or between source and destination VPN access interfaces. Figure 4 illustrates an example of VPN PM and the difference between two VPN PM measurement methods. One is the VPN tunnel PM and the other is inter-VPN-access interface PM.

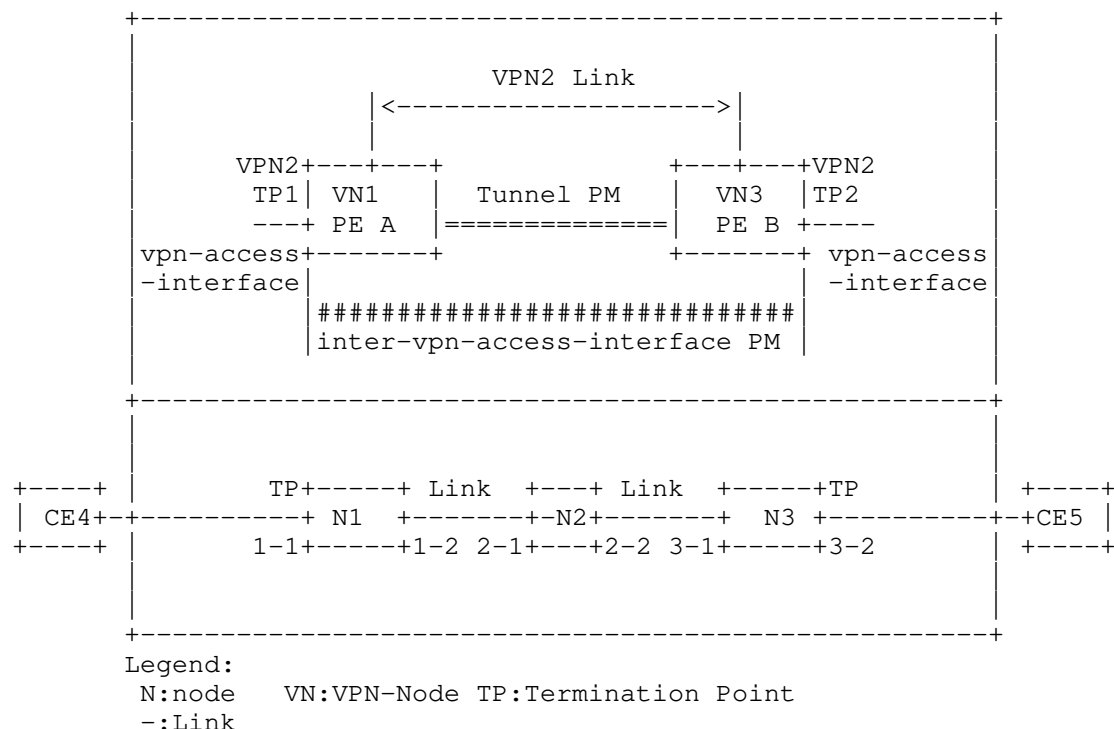


Figure 4: An Example of VPN PM



#### 4.2. Network Level

For network performance monitoring, the container of "networks" in [RFC8345] does not need to be extended.

For VPN service performance monitoring, the container "service-type" is defined to indicate the VPN type, e.g., L3VPN or Virtual Private LAN Service (VPLS). The values are taken from [RFC9181]. When a network topology instance contains the L3VPN or other L2VPN network type, it represents a VPN instance that can perform performance monitoring.

The tree in Figure 5 is a part of ietf-network-vpn-pm tree. It defines the following set of network level attributes:

"vpn-id": Refers to an identifier of VPN service defined in [RFC9181]. This identifier is used to correlate the performance status with the network service configuration.

"vpn-service-topology": Indicates the type of the VPN topology. This model supports "any-to-any", "Hub and Spoke" (where Hubs can exchange traffic), and "Hub and Spoke disjoint" (where Hubs cannot exchange traffic) that are taken from [RFC9181]. These VPN topology types can be used to describe how VPN sites communicate with each other.

```
module: ietf-network-vpn-pm
  augment /nw:networks/nw:network/nw:network-types:
    +--rw service-type!
      +--rw service-type? identityref
  augment /nw:networks/nw:network:
    +--rw vpn-pm-attributes
      +--rw vpn-id?          vpn-common:vpn-id
      +--rw vpn-service-topology? identityref
```

Figure 5: Network Level YANG Tree of the Hierarchies

#### 4.3. Node Level

The tree in Figure 6 is the node part of ietf-network-vpn-pm tree.

For network performance monitoring, a container of "pm-attributes" is augmented to the list of "node" that are defined in [RFC8345]. The container includes the following attributes:

"node-type": Indicates the device type of Provider Edge (PE),

Provider (P) device, or Autonomous System Border Router (ASBR) as defined in [RFC4026] and [RFC4364], so that the performance metric between any two nodes each with specific node type can be reported.

"entry-summary": Lists a set of IPv4 statistics, IPv6 statistics, and MAC statistics. The detailed statistics are specified separately.

For VPN service performance monitoring, the model defines one attribute:

"role": Defines the role in a particular VPN service topology. The roles are taken from [RFC9181] (e.g., any-to-any-role, spoke-role, hub-role).

```
augment /nw:networks/nw:network/nw:node:
  +--rw pm-attributes
    +--rw node-type?          identityref
    +--ro entry-summary
      +--ro ipv4-num
        +--ro maximum-routes?    uint32
        +--ro total-active-routes? uint32
      +--ro ipv6-num
        +--ro maximum-routes?    uint32
        +--ro total-active-routes? uint32
      +--ro mac-num
        +--ro mac-num-limit?      uint32
        +--ro total-active-mac-num? uint32
    +--rw role?                identityref
```

Figure 6: Node Level YANG Tree of the Hierarchies

#### 4.4. Link and Termination Point Level

The tree in Figure 7 is the link and termination point (TP) part of ietf-network-vpn-pm tree.

The 'links' are classified into two types: topology link defined in [RFC8345] and abstract link of a VPN between PEs defined in this module.

The performance data of a link is a collection of counters and gauges that report the performance status.

```

augment /nw:networks/nw:network/nt:link:
  +---rw pm-attributes
    +---rw low-percentile?                percentile
    +---rw intermediate-percentile?       percentile
    +---rw high-percentile?              percentile
    +---rw measurement-interval?         uint32
    +---ro start-time?                   yang:date-and-time
    +---ro end-time?                     yang:date-and-time
    +---ro pm-source?                    identityref
    +---ro one-way-pm-statistics
      +---ro loss-statistics
        +---ro packet-loss-count?        yang:counter64
        +---ro loss-ratio?               percentage
      +---ro delay-statistics
        +---ro unit-value?               identityref
        +---ro min-delay-value?          yang:gauge64
        +---ro max-delay-value?          yang:gauge64
        +---ro low-delay-percentile?     yang:gauge64
        +---ro intermediate-delay-percentile? yang:gauge64
        +---ro high-delay-percentile?    yang:gauge64
      +---ro jitter-statistics
        +---ro unit-value?               identityref
        +---ro min-jitter-value?         yang:gauge64
        +---ro max-jitter-value?         yang:gauge64
        +---ro low-jitter-percentile?    yang:gauge64
        +---ro intermediate-jitter-percentile? yang:gauge64
        +---ro high-jitter-percentile?   yang:gauge64
    +---ro one-way-pm-statistics-per-class* [class-id]
      +---ro class-id                    string
      +---ro loss-statistics
        +---ro packet-loss-count?        yang:counter64
        +---ro loss-ratio?               percentage
      +---ro delay-statistics
        +---ro unit-value?               identityref
        +---ro min-delay-value?          yang:gauge64
        +---ro max-delay-value?          yang:gauge64
        +---ro low-delay-percentile?     yang:gauge64
        +---ro intermediate-delay-percentile? yang:gauge64
        +---ro high-delay-percentile?    yang:gauge64
      +---ro jitter-statistics
        +---ro unit-value?               identityref
        +---ro min-jitter-value?         yang:gauge64
        +---ro max-jitter-value?         yang:gauge64
        +---ro low-jitter-percentile?    yang:gauge64
        +---ro intermediate-jitter-percentile? yang:gauge64
        +---ro high-jitter-percentile?   yang:gauge64
  +---rw vpn-pm-type
    +---rw inter-vpn-access-interface

```

```

    |   +--rw inter-vpn-access-interface?   empty
    +--rw underlay-tunnel!
        +--ro vpn-underlay-transport-type?   identityref
augment /nw:networks/nw:network/nw:node/nt:termination-point:
+--ro pm-statistics
+--ro reference-time?           yang:date-and-time
+--ro inbound-octets?           yang:counter64
+--ro inbound-unicast?         yang:counter64
+--ro inbound-nunicast?        yang:counter64
+--ro inbound-discards?        yang:counter64
+--ro inbound-errors?          yang:counter64
+--ro inbound-unknown-protocol? yang:counter64
+--ro outbound-octets?          yang:counter64
+--ro outbound-unicast?         yang:counter64
+--ro outbound-nunicast?       yang:counter64
+--ro outbound-discards?       yang:counter64
+--ro outbound-errors?         yang:counter64
+--ro vpn-network-access* [network-access-id]
+--ro network-access-id        vpn-common:vpn-id
+--ro reference-time?          yang:date-and-time
+--ro inbound-octets?          yang:counter64
+--ro inbound-unicast?         yang:counter64
+--ro inbound-nunicast?        yang:counter64
+--ro inbound-discards?        yang:counter64
+--ro inbound-errors?          yang:counter64
+--ro inbound-unknown-protocol? yang:counter64
+--ro outbound-octets?          yang:counter64
+--ro outbound-unicast?         yang:counter64
+--ro outbound-nunicast?       yang:counter64
+--ro outbound-discards?       yang:counter64
+--ro outbound-errors?         yang:counter64

```

Figure 7: Link and Termination point Level YANG Tree of the hierarchies

For the data nodes of 'link' depicted in Figure 7, the YANG module defines the following minimal set of link-level performance attributes:

**Percentile parameters:** The module supports reporting delay and jitter metric by percentile values. By default, low percentile (10th percentile), intermediate percentile (50th percentile), high percentile (90th percentile) are used. Setting a percentile to 0.00 indicates the client is not interested in receiving particular percentile. If all percentile nodes are set to 0.00, this represents that no percentile related nodes will be reported for a given performance metric (e.g., one-way delay, one-way delay variation) and only peak/min values will be reported. For

example, a client can inform the server that it is interested in receiving only high percentiles. Then for a given link, at a given "start-time", "end-time" and "measurement-interval", the 'high-delay-percentile' and 'high-jitter-percentile' will be reported. An example to illustrate the use of percentiles is provided in Appendix A.3.

PM source ("pm-source"): Indicates the performance monitoring source. The data for the topology link can be based, e.g., on BGP-LS [RFC8571]. The statistics of the VPN abstract links can be collected based upon VPN OAM mechanisms, e.g., OAM mechanisms referenced in [RFC9182], or Ethernet service OAM [ITU-T-Y-1731] referenced in [I-D.ietf-opsawg-l2nm]. Alternatively, the data can be based upon the underlay technology OAM mechanisms, for example, Generic Routing Encapsulation (GRE) tunnel OAM.

Measurement interval ("measurement-interval"): Specifies the performance measurement interval, in seconds.

Start time ("start-time"): Indicates the start time of the performance measurement for link statistics.

End time ("end-time"): Indicates the end time of the performance measurement for link statistics.

Reference time ("reference-time"): Indicates the timestamp when the counters are obtained.

Loss statistics: A set of one-way loss statistics attributes that are used to measure end to end loss between VPN sites or between any two network nodes. The exact loss value or the loss percentage can be reported.

Delay statistics: A set of one-way delay statistics attributes that are used to measure end to end latency between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.

Jitter statistics: A set of one-way IP Packet Delay Variation [RFC3393] statistics attributes that are used to measure end to end jitter between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.

PM statistics per class ("one-way-pm-statistics-per-class"): Lists p

formance measurement statistics for the topology link or the abstract underlay link between VPN PEs with given "class-id" names. The list is defined separately from "one-way-pm-statistics", which is used to collect generic metrics for unspecified "class-id" names.

VPN PM type ("vpn-pm-type"): Indicates the VPN performance type, which can be inter-vpn-access-interface PM or VPN underlay-tunnel PM. These two methods are common VPN measurement methods. The inter-VPN-access-interface PM is to monitor the performance of logical point-to-point VPN connections between a source and a destination VPN access interfaces. And the underlay-tunnel PM is to monitor the performance of underlay tunnels of VPNs. The inter-VPN-access-interface PM includes PE-PE monitoring. Therefore, usually only one of the two methods is used. The inter-VPN-access-interface PM is defined as an empty leaf, which is not bound to a specific VPN access interface. The source or destination VPN access interface of the measurement can be augmented as needed.

VPN underlay transport type ("vpn-underlay-transport-type"): Indicates the abstract link protocol-type of a VPN, such as GRE or IP-in-IP. The leaf refers to an identifier of the "underlay-transport" defined in [RFC9181], which describes the transport technology to carry the traffic of the VPN service.

For the data nodes of 'termination-point' depicted in Figure 7, the module defines the following minimal set of statistics:

Inbound statistics: A set of inbound statistics attributes that are used to measure the inbound statistics of the termination point, such as received packets, received packets with errors, etc.

Outbound statistics: A set of outbound statistics attributes that are used to measure the outbound statistics of the termination point, such as sent packets, packets that could not be sent due to errors, etc.

VPN network access ("vpn-network-access"): Lists counters of the VPN network access defined in [RFC9182] or [I-D.ietf-opsawg-l2nm]. When multiple VPN network accesses are created using the same physical port, finer-grained metrics can be monitored. If a TP is associated with only a single VPN, this list is not required.

## 5. Network and VPN Service Performance Monitoring YANG Module

The "ietf-network-vpn-pm" module uses types defined in [RFC8345], [RFC6991], [RFC8532], and [RFC9181].

```
<CODE BEGINS> file "ietf-network-vpn-pm@2022-05-05.yang"
module ietf-network-vpn-pm {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm";
  prefix nvp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Types";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and
      Layer 3 VPNs.";
  }
  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network
      Topologies, Section 6.1";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "RFC 8345: A YANG Data Model for Network
      Topologies, Section 6.2";
  }
  import ietf-lime-time-types {
    prefix lime;
    reference
      "RFC 8532: Generic YANG Data Model for the Management of
      Operations, Administration, and Maintenance (OAM) Protocols
      That Use Connectionless Communications";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
    WG List:  <mailto:opsawg@ietf.org>

    Editor: Bo Wu
           <lane.wubo@huawei.com>
    Editor: Mohamed Boucadair
           <mohamed.boucadair@orange.com>
    Editor: Qin Wu
```

```
<bill.wu@huawei.com>
Author: Oscar Gonzalez de Dios
       <oscar.gonzalezdedios@telefonica.com>
Author: Bin Wen
       <bin_wen@comcast.com>;
description
  "This module defines a model for Network and VPN Service
  Performance monitoring.

  Copyright (c) 2022 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Revised BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX
  (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
  for full legal notices.";

// RFC Ed.: update the date below with the date of RFC
// publication and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note.

revision 2022-05-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Model for Network and VPN Service
    Performance Monitoring";
}

identity node-type {
  description
    "Base identity for node type";
}

identity pe {
  base node-type;
  description
    "Provider Edge (PE) node type. A PE is the name of the device
    or set of devices at the edge of the provider network with the
    functionality that is needed to interface with the customer.";
}
```



```
identity p {
  base node-type;
  description
    "Provider router node type. That is, a router
    in the core network that does not have interfaces
    directly toward a customer.";
}

identity asbr {
  base node-type;
  description
    "Autonomous System Border Router (ASBR) node type.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
}

identity pm-source-type {
  description
    "Base identity from which specific performance monitoring
    mechanism types are derived.";
}

identity pm-source-bgppls {
  base pm-source-type;
  description
    "Indicates BGP-LS as the performance monitoring metric source";
  reference
    "RFC 8571: BGP - Link State (BGP-LS) Advertisement of
    IGP Traffic Engineering Performance Metric Extensions";
}

identity pm-source-owamp {
  base pm-source-type;
  description
    "Indicates One-Way Active Measurement Protocol (OWAMP)
    as the performance monitoring metric source.";
  reference
    "RFC 4656: A One-Way Active Measurement Protocol (OWAMP)";
}

identity pm-source-twamp {
  base pm-source-type;
  description
    "Indicates Two-Way Active Measurement Protocol (TWAMP)
    as the performance monitoring metric source.";
  reference
    "RFC 5357: A Two-Way Active Measurement Protocol (TWAMP)";
}
```

```
identity pm-source-stamp {
  base pm-source-type;
  description
    "Indicates Simple Two-way Active Measurement Protocol (STAMP)
    as the performance monitoring metric source.";
  reference
    "RFC 8762: Simple Two-Way Active Measurement Protocol";
}

identity pm-source-y-1731 {
  base pm-source-type;
  description
    "Indicates Ethernet OAM Y.1731 as the performance monitoring
    metric source.";
  reference
    "ITU-T Y.1731: Operations, administration and
    maintenance (OAM) functions and mechanisms
    for Ethernet-based networks";
}

typedef percentage {
  type decimal64 {
    fraction-digits 5;
    range "0..100";
  }
  description
    "Percentage.";
}

typedef percentile {
  type decimal64 {
    fraction-digits 2;
    range "0..100";
  }
  description
    "The percentile is a value between 0 and 100,
    e.g. 10.00, 99.90 ,99.99 etc..
    For example, for a given one-way delay measurement,
    if the percentile is set to 95.00 and
    the 95th percentile one-way delay is 2 milliseconds,
    then the 95 percent of the sample value
    is less than or equal to 2 milliseconds.";
}

grouping entry-summary {
  description
    "Entry summary grouping used for network topology
    augmentation.";
```

```
container entry-summary {
  config false;
  description
    "Container for VPN or network entry summary.";
  container ipv4-num {
    leaf maximum-routes {
      type uint32;
      description
        "Indicates the maximum number of IPv4 routes
        for the VPN.";
    }
    leaf total-active-routes {
      type uint32;
      description
        "Indicates total active IPv4 routes for the VPN.";
    }
    description
      "IPv4-specific parameters.";
  }
  container ipv6-num {
    leaf maximum-routes {
      type uint32;
      description
        "Indicates the maximum number of IPv6 routes
        for the VPN.";
    }
    leaf total-active-routes {
      type uint32;
      description
        "Indicates total active IPv6 routes for the VPN.";
    }
    description
      "IPv6-specific parameters.";
  }
  container mac-num {
    leaf mac-num-limit {
      type uint32;
      description
        "Maximum number of MAC addresses.";
    }
    leaf total-active-mac-num {
      type uint32;
      description
        "Total active MAC entries for the VPN.";
    }
    description
      "MAC statistics.";
  }
}
```

```
    }
  }

  grouping link-loss-statistics {
    description
      "Grouping for per link error statistics.";
    container loss-statistics {
      description
        "One-way link loss summarized information.";
      reference
        "RFC 4656: A One-way Active Measurement Protocol (OWAMP)
        ITU-T Y.1731: Operations, administration and
        maintenance (OAM) functions and mechanisms
        for Ethernet-based networks";
      leaf packet-loss-count {
        type yang:counter64;
        description
          "Total received packet drops count.";
      }
      leaf loss-ratio {
        type percentage;
        description
          "Loss ratio of the packets. Express as percentage
          of packets lost with respect to packets sent.";
      }
    }
  }

  grouping link-delay-statistics {
    description
      "Grouping for per link delay statistics.";
    container delay-statistics {
      description
        "One-way link delay summarized information.";
      reference
        "RFC 4656: A One-way Active Measurement Protocol (OWAMP)
        ITU-T Y.1731: Operations, administration and
        maintenance (OAM) functions and mechanisms
        for Ethernet-based networks";
      leaf unit-value {
        type identityref {
          base lime:time-unit-type;
        }
        default "lime:milliseconds";
        description
          "Time units, where the options are s, ms, ns, etc.";
      }
      leaf min-delay-value {
```

```
        type yang:gauge64;
        description
            "Minimum observed one-way delay.";
    }
    leaf max-delay-value {
        type yang:gauge64;
        description
            "Maximum observed one-way delay.";
    }
    leaf low-delay-percentile {
        type yang:gauge64;
        description
            "Low percentile of observed one-way delay with
             specific measurement method.";
    }
    leaf intermediate-delay-percentile {
        type yang:gauge64;
        description
            "Intermediate percentile of observed one-way delay with
             specific measurement method.";
    }
    leaf high-delay-percentile {
        type yang:gauge64;
        description
            "High percentile of observed one-way delay with
             specific measurement method.";
    }
}

grouping link-jitter-statistics {
    description
        "Grouping for per link jitter statistics.";
    container jitter-statistics {
        description
            "One-way link jitter summarized information.";
        reference
            "RFC 3393: IP Packet Delay Variation Metric
             for IP Performance Metrics (IPPM)
             RFC 4656: A One-way Active Measurement Protocol (OWAMP)
             ITU-T Y.1731: Operations, administration and
             maintenance (OAM) functions and mechanisms
             for Ethernet-based networks";
        leaf unit-value {
            type identityref {
                base lime:time-unit-type;
            }
            default "lime:milliseconds";
        }
    }
}
```

```
        description
            "Time units, where the options are s, ms, ns, etc.";
    }
    leaf min-jitter-value {
        type yang:gauge64;
        description
            "Minimum observed one-way jitter.";
    }
    leaf max-jitter-value {
        type yang:gauge64;
        description
            "Maximum observed one-way jitter.";
    }
    leaf low-jitter-percentile {
        type yang:gauge64;
        description
            "Low percentile of observed one-way jitter.";
    }
    leaf intermediate-jitter-percentile {
        type yang:gauge64;
        description
            "Intermediate percentile of observed one-way jitter.";
    }
    leaf high-jitter-percentile {
        type yang:gauge64;
        description
            "High percentile of observed one-way jitter.";
    }
}

grouping tp-svc-telemetry {
    leaf reference-time {
        type yang:date-and-time;
        config false;
        description
            "Indicates the time when the statistics are collected.";
    }
    leaf inbound-octets {
        type yang:counter64;
        description
            "The total number of octets received on the
            interface, including framing characters.";
    }
    leaf inbound-unicast {
        type yang:counter64;
        description
            "The total number of inbound unicast packets.";
```

```
}
leaf inbound-nunicast {
  type yang:counter64;
  description
    "The total number of inbound non-unicast
    (i.e., broadcast or multicast) packets.";
}
leaf inbound-discards {
  type yang:counter64;
  description
    "The number of inbound packets that were chosen to be
    discarded even though no errors had been detected.
    Possible reasons for discarding such a packet could
    be to free up buffer space, not enough buffer for
    too much data, etc.";
}
leaf inbound-errors {
  type yang:counter64;
  description
    "The number of inbound packets that contained errors.";
}
leaf inbound-unknown-protocol {
  type yang:counter64;
  description
    "The number of packets received via the interface
    which were discarded because of an unknown or
    unsupported protocol.";
}
leaf outbound-octets {
  type yang:counter64;
  description
    "The total number of octets transmitted out of the
    interface, including framing characters.";
}
leaf outbound-unicast {
  type yang:counter64;
  description
    "The total number of outbound unicast packets.";
}
leaf outbound-nunicast {
  type yang:counter64;
  description
    "The total number of outbound non unicast
    (i.e., broadcast or multicast) packets.";
}
leaf outbound-discards {
  type yang:counter64;
  description
```

```
        "The number of outbound packets which were chosen
        to be discarded even though no errors had been
        detected to prevent their being transmitted.
        Possible reasons for discarding such a packet could
        be to free up buffer space, not enough buffer for
        too much data, etc.";
    }
    leaf outbound-errors {
        type yang:counter64;
        description
            "The number of outbound packets that contained
            errors.";
    }
    description
        "Grouping for interface service telemetry.";
}

augment "/nw:networks/nw:network/nw:network-types" {
    description
        "Defines the service topologies types.";
    container service-type {
        presence "Indicates network service topology.";
        leaf service-type {
            type identityref {
                base vpn-common:service-type;
            }
            description
                "The presence identifies the network service type,
                e.g., L3VPN, VPLS, etc.";
        }
        description
            "Container for VPN service type.";
    }
}

augment "/nw:networks/nw:network" {
    when 'nw:network-types/nvp:service-type' {
        description
            "Augments only for VPN Network topology.";
    }
    description
        "Augments the network with service topology attributes";
    container vpn-pm-attributes {
        leaf vpn-id {
            type vpn-common:vpn-id;
            description
                "VPN identifier.";
        }
    }
}
```



```
    leaf vpn-service-topology {
      type identityref {
        base vpn-common:vpn-topology;
      }
      description
        "VPN service topology, e.g., hub-spoke, any-to-any,
        hub-spoke-disjoint.";
    }
    description
      "Container for VPN topology attributes.";
  }
}

augment "/nw:networks/nw:network/nw:node" {
  description
    "Augments the network node with other general attributes.";
  container pm-attributes {
    leaf node-type {
      type identityref {
        base node-type;
      }
      description
        "Node type, e.g., PE, P, ASBR.";
    }
    description
      "Container for node attributes.";
    uses entry-summary;
  }
}

augment "/nw:networks/nw:network/nw:node/pm-attributes" {
  when '../nw:network-types/nvp:service-type' {
    description
      "Augments only for VPN node attributes.";
  }
  description
    "Augments the network node with VPN specific attributes.";
  leaf role {
    type identityref {
      base vpn-common:role;
    }
    default "vpn-common:any-to-any-role";
    description
      "Role of the node in the VPN.";
  }
}

augment "/nw:networks/nw:network/nt:link" {
```

```
description
  "Augments the network topology link with performance
   monitoring attributes.";
container pm-attributes {
  description
    "Container for PM attributes.";
  leaf low-percentile {
    type percentile;
    default "10.00";
    description
      "Low percentile to report. Setting low-percentile
       into 0.00 indicates the client is not interested
       in receiving low percentile.";
  }
  leaf intermediate-percentile {
    type percentile;
    default "50.00";
    description
      "Intermediate percentile to report. Setting
       intermediate-percentile into 0.00 indicates the client
       is not interested in receiving intermediate percentile.";
  }
  leaf high-percentile {
    type percentile;
    default "95.00";
    description
      "High percentile to report. Setting high-percentile
       into 0.00 indicates the client is not interested in
       receiving high percentile.";
  }
  leaf measurement-interval {
    type uint32 {
      range "1..max";
    }
    units "seconds";
    default "60";
    description
      "Indicates the time interval to perform PM measurement.";
  }
  leaf start-time {
    type yang:date-and-time;
    config false;
    description
      "The time that the current measurement started.";
  }
  leaf end-time {
    type yang:date-and-time;
    config false;
  }
}
```

```
        description
            "The time that the current measurement ended.";
    }
    leaf pm-source {
        type identityref {
            base pm-source-type;
        }
        config false;
        description
            "The OAM tool used to collect the PM data.";
    }
    container one-way-pm-statistics {
        config false;
        description
            "Container for link telemetry attributes.";
        uses link-loss-statistics;
        uses link-delay-statistics;
        uses link-jitter-statistics;
    }
    list one-way-pm-statistics-per-class {
        key "class-id";
        config false;
        description
            "The list of PM data based on class of service.";
        leaf class-id {
            type string;
            description
                "The class-id is used to identify the class of service.
                This identifier is internal to the administration.";
        }
        uses link-loss-statistics;
        uses link-delay-statistics;
        uses link-jitter-statistics;
    }
}

augment "/nw:networks/nw:network/nt:link/pm-attributes" {
    when '../..//nw:network-types/nvp:service-type' {
        description
            "Augments only for VPN Network topology.";
    }
    description
        "Augments the network topology link with VPN service
        performance monitoring attributes.";
    container vpn-pm-type {
        description
            "The VPN PM type of this logical point-to-point
```

```
    unidirectional VPN link.";
  container inter-vpn-access-interface {
    description
      "Indicates inter-vpn-access-interface PM, which is to
       monitor the performance of logical point-to-point VPN
       connections between a source and a destination
       VPN access interfaces.";
    leaf inter-vpn-access-interface {
      type empty;
      description
        "This is a placeholder for inter-vpn-access-interface PM,
         which is not bound to a specific VPN access interface.
         The source or destination VPN access interface
         of the measurement can be augmented as needed.";
    }
  }
  container underlay-tunnel {
    presence "Enables VPN underlay tunnel PM";
    description
      "Indicates underlay-tunnel PM, which is to monitor
       the performance of underlay tunnels of VPNs.";
    leaf vpn-underlay-transport-type {
      type identityref {
        base vpn-common:protocol-type;
      }
      config false;
      description
        "The leaf indicates the underlay transport type of
         a VPN service, e.g., GRE, LDP, etc.";
    }
  }
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
  description
    "Augments the network topology termination point with
     performance monitoring attributes.";
  container pm-statistics {
    config false;
    description
      "Container for termination point PM attributes.";
    uses tp-svc-telemetry;
  }
}

augment "/nw:networks/nw:network/nw:node"
+ "/nt:termination-point/pm-statistics" {
```

```
when '../.../nw:network-types/nvp:service-type' {
  description
    "Augments only for VPN Network topology.";
}
description
  "Augments the network topology termination-point with
  VPN service performance monitoring attributes";
list vpn-network-access {
  key "network-access-id";
  description
    "The list of PM based on VPN network accesses.";
  leaf network-access-id {
    type vpn-common:vpn-id;
    description
      "References to an identifier for the VPN network
      access, e.g. L3VPN or VPLS.";
  }
  uses tp-svc-telemetry;
}
}
}
}
<CODE ENDS>
```

## 6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- \* `"/nw:networks/nw:network/nw:network-types"`: This subtree specifies the VPN service type. Unauthorized access to this subtree may render the VPN service type invalid.
- \* `"/nw:networks/nw:network/nvp:vpn-pm-attributes"`: This subtree specifies the VPN service identifier and VPN service topology. Unauthorized access to this subtree may disable the the VPN PM or render the VPN service topology invalid.
- \* `"/nw:networks/nw:network/nw:node/nvp:pm-attributes"`: This subtree specifies the network node type and VPN service topology role. Unauthorized access to this subtree may render the node type or VPN service topology invalid.
- \* `"/nw:networks/nw:network/nt:link/nvp:pm-attributes"`: This subtree specifies the PM of the network link and VPN link. Unauthorized access to this subtree can impact the network and VPN monitoring.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via `get`, `get-config`, or `notification`) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- \* `"/nw:networks/nw:network/nw:node/nvp:pm-attributes/nvp:vpn-summary-statistics"`: Unauthorized access to this subtree can disclose the operational state information of VPN instances.
- \* `"/nw:networks/nw:network/nt:link/nvp:pm-attributes/nvp:one-way-pm-statistics"`: Unauthorized access to this subtree can disclose the operational state information of network links or VPN abstract links.
- \* `"/nw:networks/nw:network/nw:node/nt:termination-point/nvp:pm-statistics"`: Unauthorized access to this subtree can disclose the operational state information of network termination points or VPN network accesses.

## 7. IANA Considerations

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: `urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm`  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

Name: ietf-network-vpn-pm  
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm  
Maintained by IANA: N  
Prefix: nvp  
Reference: RFC XXXX (RFC Ed.: replace XXXX with actual RFC number and remove this note.)

## 8. Acknowledgements

Thanks to Joe Clarke, Adrian Farrel, Tom Petch, Greg Mirsky, Roque Gagliano, Erez Segev, and Dhruv Dhody for reviewing and providing important input to this document.

## 9. Contributors

The following authors contributed significantly to this document:

Michale Wang  
Huawei  
Email: wangzitao@huawei.com

Roni Even  
Huawei  
Email: ron.even.tlv@gmail.com

Change Liu  
China Unicom  
Email: liuc131@chinaunicom.cn

Honglei Xu  
China Telecom  
Email: xuhl6@chinatelecom.cn

## 10. References

### 10.1. Normative References

[ITU-T-Y-1731]  
ITU-T, "Operator Ethernet Service Definition", August 2015, <<https://www.itu.int/rec/T-REC-Y.1731/en>>.

- [RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <<https://www.rfc-editor.org/info/rfc3393>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<https://www.rfc-editor.org/info/rfc6374>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.



- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8532] Kumar, D., Wang, Z., Wu, Q., Ed., Rahman, R., and S. Raghavan, "Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8532, DOI 10.17487/RFC8532, April 2019, <<https://www.rfc-editor.org/info/rfc8532>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <<https://www.rfc-editor.org/info/rfc8762>>.

- [RFC9181] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., and Q. Wu, "A Common YANG Data Model for Layer 2 and Layer 3 VPNs", RFC 9181, DOI 10.17487/RFC9181, February 2022, <<https://www.rfc-editor.org/info/rfc9181>>.

## 10.2. Informative References

- [I-D.ietf-netmod-node-tags]  
Wu, Q., Claise, B., Liu, P., Du, Z., and M. Boucadair, "Data Node Tags in YANG Modules", Work in Progress, Internet-Draft, draft-ietf-netmod-node-tags-07, 29 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-netmod-node-tags-07.txt>>.
- [I-D.ietf-opsawg-l2nm]  
Boucadair, M., Dios, O. G. D., Barguil, S., and L. A. Munoz, "A YANG Network Data Model for Layer 2 VPNs", Work in Progress, Internet-Draft, draft-ietf-opsawg-l2nm-15, 29 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-l2nm-15.txt>>.
- [I-D.ietf-opsawg-sap]  
Boucadair, M., Dios, O. G. D., Barguil, S., Wu, Q., and V. Lopez, "A Network YANG Model for Service Attachment Points (SAPs)", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-04, 11 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-sap-04.txt>>.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <<https://www.rfc-editor.org/info/rfc4026>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<https://www.rfc-editor.org/info/rfc5277>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.

- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/info/rfc8570>>.
- [RFC8632] Vallin, S. and M. Bjorklund, "A YANG Data Model for Alarm Management", RFC 8632, DOI 10.17487/RFC8632, September 2019, <<https://www.rfc-editor.org/info/rfc8632>>.
- [RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <<https://www.rfc-editor.org/info/rfc8639>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.
- [RFC9182] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., Munoz, L., and A. Aguado, "A YANG Network Data Model for Layer 3 VPNs", RFC 9182, DOI 10.17487/RFC9182, February 2022, <<https://www.rfc-editor.org/info/rfc9182>>.

## Appendix A. Illustrative Examples

### A.1. VPN Performance Subscription Example

The example shown in Figure 8 illustrates how a client subscribes to the performance monitoring information between nodes ('node-id') A and B in the L3 network topology. The performance monitoring parameter that the client is interested in is end-to-end loss.

```
POST /restconf/operations
    /ietf-subscribed-notifications:establish-subscription
{
  "ietf-subscribed-notifications:input":{
    "stream-subtree-filter":{
      "ietf-network:networks":{
        "network":{
          "network-id":"foo:l3-network",
          "ietf-network-vpn-pm:service-type":{
            "ietf-vpn-common:l3vpn":{}
          }
        }
      }
    }
  }
```

```

    },
    "node":[
      {
        "node-id":"A",
        "ietf-network-vpn-pm:pm-attributes":{
          "node-type":"PE"
        },
        "termination-point":{
          "tp-id":"1-0-1"
        }
      },
      {
        "node-id":"B",
        "ietf-network-vpn-pm:pm-attributes":{
          "node-type":"PE"
        },
        "termination-point":{
          "tp-id":"2-0-1"
        }
      }
    ],
    "ietf-network-topology:link":{
      "link-id":"A-B",
      "source":{
        "source-node":"A"
      },
      "destination":{
        "dest-node":"B"
      },
      "ietf-network-vpn-pm:pm-attributes":{
        "one-way-pm-statistics":{
          "loss-statistics":{
            "packet-loss-count":{}
          }
        },
        "vpn-underlay-transport-type":"ietf-vpn-common:gre"
      }
    }
  }
},
"ietf-yang-push:periodic":{
  "ietf-yang-push:period":"500"
}
}

```

Figure 8: Pub/Sub Retrieval

#### A.2. Example of VPN Performance Snapshot

This example, depicted in Figure 9, illustrates an VPN PM instance example in which a client uses RESTCONF [RFC8040] to fetch the performance data of the link and TP belonged to "VPN1".

```

{
  "ietf-network:networks": {
    "network": {
      "network-id": "foo:vpn1",
      "node": [
        {
          "node-id": "A",
          "ietf-network-vpn-pm:pm-attributes": {
            "node-type": "PE"
          },
          "termination-point": {
            "tp-id": "1-0-1",
            "ietf-network-vpn-pm:pm-statistics": {
              "inbound-octets": "100",
              "outbound-octets": "150"
            }
          }
        },
        {
          "node-id": "B",
          "ietf-network-vpn-pm:pm-attributes": {
            "node-type": "PE"
          },
          "termination-point": {
            "tp-id": "2-0-1",
            "ietf-network-vpn-pm:pm-statistics": {
              "inbound-octets": "150",
              "outbound-octets": "100"
            }
          }
        }
      ],
      "ietf-network-topology:link": {
        "link-id": "A-B",
        "source": { "source-node": "A" },
        "destination": { "dest-node": "B" },
        "ietf-network-pm:pm-attributes": {
          "one-way-pm-statistics": {
            "loss-statistics": { "packet-loss-count": "120" }
          },
          "vpn-underlay-transport-type": "ietf-vpn-common:gre"
        }
      }
    }
  }
}

```

Figure 9

### A.3. Example of Percentile Monitoring

The following shows an example of a percentile measurement for a VPN link.

```
{
  "ietf-network-topology:link": [
    {
      "link-id": "foo:vpn1-link1",
      "source": {
        "source-node": "vpn-node1"
      },
      "destination": {
        "dest-node": "vpn-node3"
      },
      "ietf-network-vpn-pm:pm-attributes": {
        "low-percentile": "20.00",
        "intermediate-percentile": "50.00",
        "high-percentile": "90.00",
        "one-way-pm-statistics": {
          "delay-statistics": {
            "unit-value": "lime:milliseconds",
            "min-delay-value": "43",
            "max-delay-value": "99",
            "low-delay-percentile": "64",
            "intermediate-delay-percentile": "77",
            "high-delay-percentile": "98"
          }
        },
        "vpn-pm-type": {
          "inter-vpn-access-interface": [null]
        }
      }
    }
  ]
}
```

### Authors' Addresses

Bo Wu (editor)  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing  
Jiangsu, 210012  
China  
Email: lana.wubo@huawei.com

Qin Wu (editor)  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing  
Jiangsu, 210012  
China  
Email: bill.wu@huawei.com

Mohamed Boucadair (editor)  
Orange  
Rennes 35000  
France  
Email: mohamed.boucadair@orange.com

Oscar Gonzalez de Dios  
Telefonica  
Madrid  
Spain  
Email: oscar.gonzalezdedios@telefonica.com

Bin Wen  
Comcast  
Email: bin\_wen@comcast.com



Network Working Group  
Internet-Draft  
Updates: 8520 (if approved)  
Intended status: Standards Track  
Expires: 28 November 2021

E. Lear  
Cisco Systems  
C. Bormann  
Universität Bremen TZI  
27 May 2021

Ownership and licensing statements in YANG  
draft-lear-opsawg-ol-01

Abstract

This memo provides for an extension to RFC 8520 that allows MUD file authors to specify ownership and licensing of MUD files themselves. This memo updates RFC 8520. However, it can also be used for purposes outside of MUD, and the grouping is structured as such.

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 November 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. The owner-license extension and model . . . . .	3
3. The YANG schema for ownership and licensing . . . . .	3
4. Extension for MUD . . . . .	5
5. Example . . . . .	5
6. Security Considerations . . . . .	6
7. IANA Considerations . . . . .	6
7.1. MUD Extension . . . . .	6
8. Normative References . . . . .	6
Appendix A. Changes from Earlier Versions . . . . .	7
Authors' Addresses . . . . .	7

## 1. Introduction

[RFC8520] Manufacturer Usage Descriptions (MUD) can be used to describe devices and their requirements to the network infrastructure. The original version of the specification does not provide for a means to specify ownership and licensing of the MUD file itself. This can hinder those wishing to use, modify, or adapt MUD files for the purpose of offering them, when the manufacturer is not involved.

\* \*Issue\*: Should this be an owner or an originator?

To avoid any confusion, we define an extension that allows for specifying of owners and licensing terms for MUD files.

Those generating MUD files SHOULD use this extension, and thus this extension updates RFC 8520.

There are two ways to specify a license: a URL pointing to the license itself or an SPDX tag [SPDX]. If an SPDX tag is supplied consumers MUST interpret that tag through its meaning as specified by [SPDX].

\* \*Issue\*: Should we simply say that a URI contains a colon and SPDX license identifier doesn't?

This grouping may be used for other YANG models that reside as static objects.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. The owner-license extension and model

Because the model is primarily motivated by MUD, and based on the way that YANG trees are formed, the model appears as an augmentation for MUD.

```
module: ietf-ol
```

```
augment /mud:mud:
  +--rw ol
    +--rw owners*          string
    +--rw (license-type)?
      +--:(spdx-lt)
        | +--rw spdx-tag?    string
      +--:(url)
        +--rw license-info?  inet:uri
```

- \* *Issue*: Should different owners possibly have their own license types? (Logical and, derived works.)
- \* *Issue*: Should a single owner possibly have multiple license types? (Logical or, multi-licensing.)

## 3. The YANG schema for ownership and licensing

The following grouping and augmentation are proposed.

```
<CODE BEGINS>
file "ietf-ol@2021-05-21.yang"
module ietf-ol {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ol";
  prefix ol;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-mud {
    prefix mud;
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG
    Web: http://tools.ietf.org/wg/opsawg/
```

```
WG List: opsawg@ietf.org
Author: Eliot Lear lear@cisco.com
Author: Carsten Bormann <cabo@tzi.org>;
description
  "This YANG module to indicate ownership and licensing.

  Copyright (c) 2021 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject to
  the license terms contained in, the Simplified BSD License set
  forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX
  (https://www.rfc-editor.org/info/rfcXXXX);
  see the RFC itself for full legal notices.

  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 BCP 14 (RFC 2119) (RFC 8174) when, and only when,
  they appear in all capitals, as shown here. ";

revision 2021-05-21 {
  description
    "Initial proposed standard.";
  reference
    "RFC XXXX: Extension for ownership and licensing";
}

grouping ol-extension {
  description
    "OL extension grouping";
  container ol {
    description
      "container relating to ownership and licensing.";
    leaf-list owners {
      type string;
      description
        "A list of owners, may be in the form of a copyright
        Example: Copyright (c) Jane Smith. All rights Reserved.";
    }
    choice license-type {
      description "Either choose a standard license type or point
        to one of your own.";
```

```
    case spdx-lt {
      leaf spdx-tag {
        type string;
        description "SPDX License Identifier, as indicated at
                     https://spdx.org/licenses/";
      }
    }
    case url {
      leaf license-info {
        type inet:uri;
        description "A URL pointing to licensing information.";
      }
    }
  }
}
}

augment "/mud:mud" {
  description
    "Add extension for Ownership and licensing.";
  uses ol-extension;
}
}
<CODE ENDS>
```

#### 4. Extension for MUD

MUD files using this extension MUST include "ol" in the extensions array, as specified by [RFC8520].

#### 5. Example

In this example, the Frobmaster company is using the 0BSD SPDX tag to indicate a relatively open license. The "ol" extension and container are present. There is a single owner listed.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol"
    ],
    "ol": {
      "owners": [
        "Copyright (c) FrobMaster 2021. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mud-url": "https://frobs.example.com/mud/Frob.json",
    "mud-signature": "https://frobs.example.com/mud/Frob.p7s",
    "last-update": "2021-05-24T11:26:04+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "This device helps produce frobs",
    "mfg-name": "FrobMaster",
    "documentation": "https://frobs.example.com/doc",
    "model-name": "Frobinator"
  }
}
```

\* \*Task\*: need Makefile for validating this against MUD. (Probably put this into a separate file, too.)

\* \*Issue\*: Should we give an example for usage outside yang?

## 6. Security Considerations

No security considerations have been identified.

\* \*Issue\*: Should we maybe point out any specific secscons of 8520?

## 7. IANA Considerations

### 7.1. MUD Extension

The IANA is requested to add "ol" to the MUD extensions registry as follows:

Extension Name: ol  
Standard reference: This document

## 8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [SPDX] The Linux Foundation, "SPDX Specification 2.1", 2016.

#### Appendix A. Changes from Earlier Versions

Draft -01: \* Add some issues \* correct spacing

Draft -00:

\* Initial revision

#### Authors' Addresses

Eliot Lear  
Cisco Systems  
Richtistrasse 7  
CH-8304 Wallisellen  
Switzerland

Phone: +41 44 878 9200  
Email: [lear@cisco.com](mailto:lear@cisco.com)

Carsten Bormann  
Universität Bremen TZI  
Postfach 330440  
D-28359 Bremen  
Germany

Email: [cabo@tzi.org](mailto: cabo@tzi.org)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 7 April 2022

M. Tuexen, Ed.  
Muenster Univ. of Appl. Sciences  
F. Rizzo  
Politecnico di Torino  
J. Bongertz  
Airbus DS CyberSecurity  
G. Combs  
Wireshark  
G. Harris

E. Chaudron  
Red Hat  
M. Richardson  
Sandelman  
4 October 2021

PCAP Next Generation (pcapng) Capture File Format  
draft-tuexen-opsawg-pcapng-04

## Abstract

This document describes a format to record captured packets to a file. This format is extensible; Wireshark can currently read and write it, and libpcap can currently read some pcapng files.

## Status of This Memo

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

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

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

This Internet-Draft will expire on 7 April 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.



This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
2.1. Acronyms . . . . .	3
3. General File Structure . . . . .	4
3.1. General Block Structure . . . . .	4
3.2. Block Types . . . . .	5
3.3. Logical Block Hierarchy . . . . .	6
3.4. Physical File Layout . . . . .	7
3.5. Options . . . . .	8
3.5.1. Custom Options . . . . .	12
3.6. Data format . . . . .	13
3.6.1. Endianness . . . . .	13
3.6.2. Alignment . . . . .	14
4. Block Definition . . . . .	14
4.1. Section Header Block . . . . .	14
4.2. Interface Description Block . . . . .	18
4.3. Enhanced Packet Block . . . . .	25
4.3.1. Enhanced Packet Block Flags Word . . . . .	30
4.4. Simple Packet Block . . . . .	31
4.5. Name Resolution Block . . . . .	33
4.6. Interface Statistics Block . . . . .	37
4.7. Decryption Secrets Block . . . . .	40
4.8. Custom Block . . . . .	44
5. Vendor-Specific Custom Extensions . . . . .	46
5.1. Supported Use-Cases . . . . .	46
5.2. Controlling Copy Behavior . . . . .	47
5.3. Strings vs. Octets . . . . .	47
5.4. Endianness Issues . . . . .	47
6. Recommended File Name Extension: .pcapng . . . . .	48
7. Conclusions . . . . .	49
8. Implementations . . . . .	49
9. Security Considerations . . . . .	49
10. IANA Considerations . . . . .	49
10.1. Standardized Block Type Codes . . . . .	50
11. Contributors . . . . .	53
12. Acknowledgments . . . . .	53
13. References . . . . .	53

13.1. Normative References . . . . .	53
13.2. Informative References . . . . .	53
Appendix A. Packet Block (obsolete!) . . . . .	53
Authors' Addresses . . . . .	56

## 1. Introduction

The problem of exchanging packet traces becomes more and more critical every day; unfortunately, no standard solutions exist for this task right now. One of the most accepted packet interchange formats is the one defined by libpcap, which is rather old and is lacking in functionality for more modern applications particularly from the extensibility point of view.

This document proposes a new format for recording packet traces. The following goals are being pursued:

**Extensibility:** It should be possible to add new standard capabilities to the file format over time, and third parties should be able to enrich the information embedded in the file with proprietary extensions, with tools unaware of newer extensions being able to ignore them.

**Portability:** A capture trace must contain all the information needed to read data independently from network, hardware and operating system of the machine that made the capture.

**Merge/Append data:** It should be possible to add data at the end of a given file, and the resulting file must still be readable.

## 2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 2.1. Acronyms

The following acronyms are used throughout this document:

SHB: Section Header Block

IDB: Interface Description Block

ISB: Interface Statistics Block

EPB: Enhanced Packet Block

SPB: Simple Packet Block

NRB: Name Resolution Block

CB: Custom Block

### 3. General File Structure

#### 3.1. General Block Structure

A capture file is organized in blocks, that are appended one to another to form the file. All the blocks share a common format, which is shown in Figure 1.

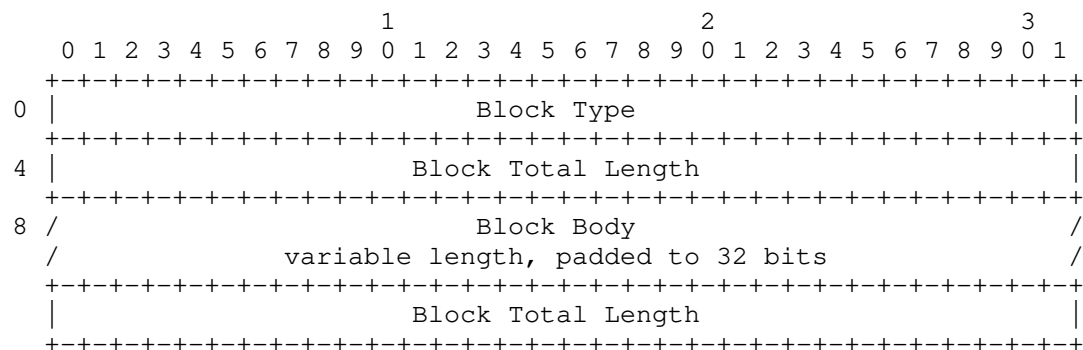


Figure 1: Basic block structure.

The fields have the following meaning:

- \* Block Type (32 bits): a unique unsigned value that identifies the block. Values whose Most Significant Bit (MSB) is equal to 1 are reserved for local use. They can be used to make extensions to the file format to save private data to the file. The list of currently defined types can be found in Section 10.1.
- \* Block Total Length (32 bits): an unsigned value giving the total size of this block, in octets. For instance, the length of a block that does not have a body is 12 octets: 4 octets for the Block Type, 4 octets for the initial Block Total Length and 4 octets for the trailing Block Total Length. This value MUST be a multiple of 4.
- \* Block Body: content of the block.

- \* Block Total Length: total size of this block, in octets. This field is duplicated to permit backward file navigation.

This structure, shared among all blocks, makes it easy to process a file and to skip unneeded or unknown blocks. Some blocks can contain other blocks inside (nested blocks). Some of the blocks are mandatory, i.e. a capture file is not valid if they are not present, other are optional.

The General Block Structure allows defining other blocks if needed. A parser that does not understand them can simply ignore their content.

### 3.2. Block Types

The currently standardized Block Type codes are specified in Section 10.1; they have been grouped in the following four categories:

The following MANDATORY block MUST appear at least once in each file:

- \* Section Header Block (Section 4.1): it defines the most important characteristics of the capture file.

The following OPTIONAL blocks MAY appear in a file:

- \* Interface Description Block (Section 4.2): it defines the most important characteristics of the interface(s) used for capturing traffic. This block is required in certain cases, as described later.
- \* Enhanced Packet Block (Section 4.3): it contains a single captured packet, or a portion of it. It represents an evolution of the original, now obsolete, Packet Block (Appendix A). If this appears in a file, an Interface Description Block is also required, before this block.
- \* Simple Packet Block (Section 4.4): it contains a single captured packet, or a portion of it, with only a minimal set of information about it. If this appears in a file, an Interface Description Block is also required, before this block.
- \* Name Resolution Block (Section 4.5): it defines the mapping from numeric addresses present in the packet capture and the canonical name counterpart.

- \* Interface Statistics Block (Section 4.6): it defines how to store some statistical data (e.g. packet dropped, etc) which can be useful to understand the conditions in which the capture has been made. If this appears in a file, an Interface Description Block is also required, before this block.
- \* Custom Block (Section 4.8): it contains vendor-specific data in a portable fashion.

The following OBSOLETE block SHOULD NOT appear in newly written files (but is documented in the Appendix for reference):

- \* Packet Block (Appendix A): it contains a single captured packet, or a portion of it. It is OBSOLETE, and superseded by the Enhanced Packet Block (Section 4.3).

The following EXPERIMENTAL blocks are considered interesting but the authors believe that they deserve more in-depth discussion before being defined:

- \* Alternative Packet Blocks
- \* Compression Block
- \* Encryption Block
- \* Fixed Length Block
- \* Directory Block
- \* Traffic Statistics and Monitoring Blocks
- \* Event/Security Blocks

Requests for new standardized Block Type codes should be made by creating a pull request to update this document as described in Section 10.1.

### 3.3. Logical Block Hierarchy

The blocks build a logical hierarchy as they refer to each other. Figure 2 shows the logical hierarchy of the currently defined blocks in the form of a "tree view":

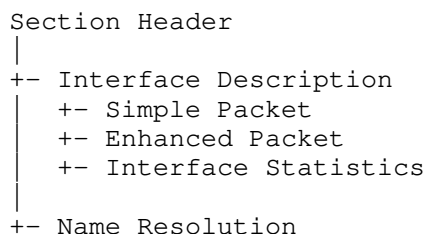


Figure 2: Logical Block Hierarchy of a pcapng File

For example: each captured packet refers to a specific capture interface, the interface itself refers to a specific section.

### 3.4. Physical File Layout

The file MUST begin with a Section Header Block. However, more than one Section Header Block can be present in the capture file, each one covering the data following it until the next one (or the end of file). A Section includes the data delimited by two Section Header Blocks (or by a Section Header Block and the end of the file), including the first Section Header Block.

In case an application cannot read a Section because of different version number, it MUST skip everything until the next Section Header Block. Note that, in order to properly skip the blocks until the next section, all blocks MUST have the fields Type and Length at the beginning. In order to properly skip blocks in the backward direction, all blocks MUST have the Length repeated at the end of the block. These are mandatory requirements that MUST be maintained in future versions of the block format.

Figure 3 shows a typical file layout, with a single Section Header that covers the whole file.

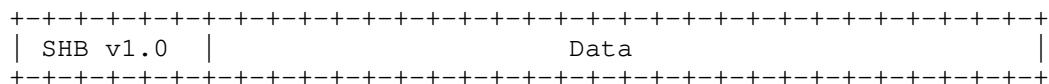


Figure 3: File structure example: Typical layout with a single Section Header Block

Figure 4 shows a file that contains three headers, and is normally the result of file concatenation. An application that understands only version 1.0 of the file format skips the intermediate section and restart processing the packets after the third Section Header.

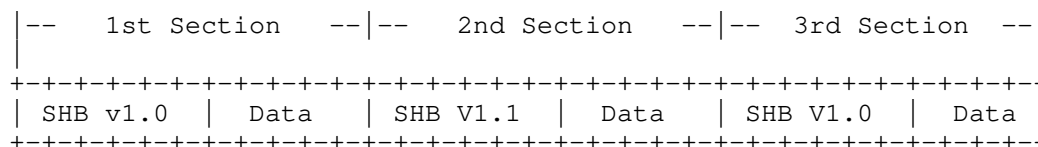


Figure 4: File structure example: three Section Header Blocks in a single file

Figure 5 shows a file comparable to a "classic libpcap" file - the minimum for a useful capture file. It contains a single Section Header Block (SHB), a single Interface Description Block (IDB) and a few Enhanced Packet Blocks (EPB).

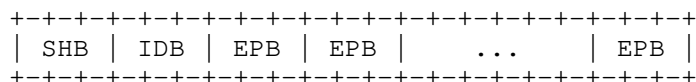


Figure 5: File structure example: a pcapng file similar to a classical libpcap file

Figure 6 shows a complex example file. In addition to the minimum file above, it contains packets captured from three interfaces, capturing on the third of which begins after packets have arrived on other interfaces, and also includes some Name Resolution Blocks (NRB) and an Interface Statistics Block (ISB).

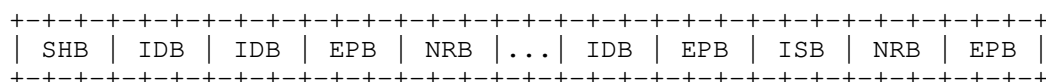


Figure 6: File structure example: complex pcapng file

The last example should make it obvious that the block structure makes the file format very flexible compared to the classical libpcap format.

### 3.5. Options

All the block bodies MAY embed optional fields. Optional fields can be used to insert some information that may be useful when reading data, but that is not really needed for packet processing. Therefore, each tool can either read the content of the optional fields (if any), or skip some of them or even all at once.

A block that may contain options must be structured so that the number of octets of data in the Block Body that precede the options can be determined from that data; that allows the beginning of the

options to be found. That is true for all standard blocks that support options; for Custom Blocks that support options, the Custom Data must be structured in such a fashion. This means that the Block Length field (present in the General Block Structure, see Section 3.1) can be used to determine how many octets of optional fields, if any, are present in the block. That number can be used to determine whether the block has optional fields (if it is zero, there are no optional fields), to check, when processing optional fields, whether any optional fields remain, and to skip all the optional fields at once.

Options are a list of Type - Length - Value fields, each one containing a single value:

- \* Option Type (16 bits): an unsigned value that contains the code that specifies the type of the current TLV record. Option types whose Most Significant Bit is equal to one are reserved for local use; therefore, there is no guarantee that the code used is unique among all capture files (generated by other applications), and is most certainly not portable. For cross-platform globally unique vendor-specific extensions, the Custom Option MUST be used instead, as defined in Section 3.5.1).
- \* Option Length (16 bits): an unsigned value that contains the actual length of the following 'Option Value' field without the padding octets.
- \* Option Value (variable length): the value of the given option, padded to a 32-bit boundary. The actual length of this field (i.e. without the padding octets) is specified by the Option Length field.

Requests for new standardized option codes for a given block should be made by creating a pull request to update this document as described in Section 10.1.



A given option may have a fixed length, in which case all instances of that option have a length that is equal to the specified fixed length, or a variable length, in which case the option has a minimum length and all instances of that option must have a length equal to or greater than the specified minimum length. The length of fixed-length options, and the minimum length of variable-length options, is specified in the description of the option; if the minimum length of a variable-length option is not specified, a zero-length option is valid. Software that reads these files SHOULD report options that have an invalid length as errors; the software MAY stop processing the file if it sees an option that has invalid length, or MAY ignore the option and continue processing it. Software that writes these files MUST NOT write files with options that have invalid lengths.

If an option's value is a string, the value is not necessarily zero-terminated. Software that reads these files MUST NOT assume that strings are zero-terminated, and MUST treat a zero-value octet as a string terminator.

Some options may be repeated several times; for example, a block can have multiple comments, and an Interface Description Block can give multiple IPv4 or IPv6 addresses for the interface if it has multiple IPv4 or IPv6 addresses assigned to it. Other options may appear at most once in a given block.

The option list is terminated by a option which uses the special 'End of Option' code (opt\_endofopt). Code that writes pcapng files MUST put an opt\_endofopt option at the end of an option list. Code that reads pcapng files MUST NOT assume an option list will have an opt\_endofopt option at the end; it MUST also check for the end of the block, and SHOULD treat blocks where the option list has no opt\_endofopt option as if the option list had an opt\_endofopt option at the end.

The format of the optional fields is shown in Figure 7.

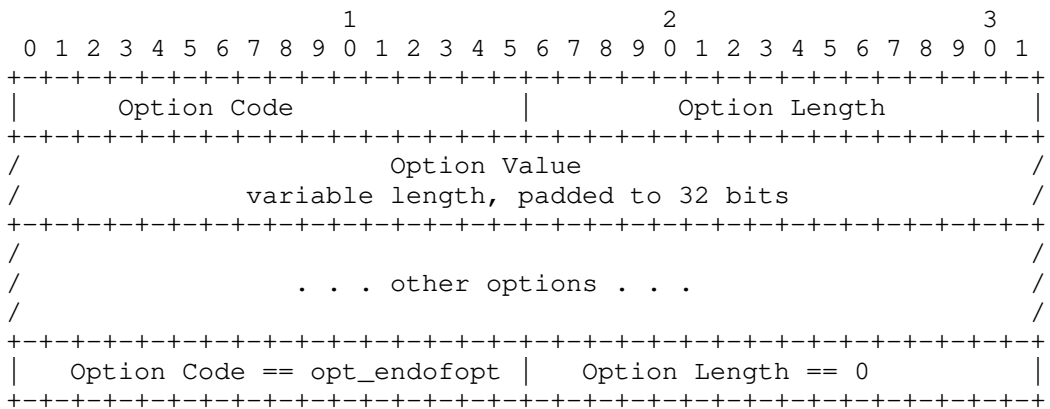


Figure 7: Options Format

The following codes can always be present in any optional field:

Name	Code	Length	Multiple allowed?
opt_endofopt	0	0	no
opt_comment	1	variable	yes
opt_custom	2988/2989/19372/19373	variable, minimum 4	yes

Table 1: Common Options

- opt\_endofopt:  
The opt\_endofopt option delimits the end of the optional fields. This option MUST NOT be repeated within a given list of options.
- opt\_comment:  
The opt\_comment option is a UTF-8 string containing human-readable comment text that is associated to the current block. Line separators SHOULD be a carriage-return + linefeed ('`\r\n`') or just linefeed ('`\n`'); either form may appear and be considered a line separator. The string is not zero-terminated.

Examples: "This packet is the beginning of all of our problems",  
 "Packets 17-23 showing a bogus TCP retransmission!\r\n This is  
 reported in bugzilla entry 1486.\nIt will be fixed in the future."

opt\_custom:

This option is described in detail in Section 3.5.1.

### 3.5.1. Custom Options

Customs Options are used for portable, vendor-specific data related to the block they're in. A Custom Option can be in any block type that can have options, can be repeated any number of times in a block, and may come before or after other option types - except the opt\_endofopt option, which is always the last option. Different Custom Options, of different type codes and/or different Private Enterprise Numbers, may be used in the same pcapng file. See Section 5 for additional details.

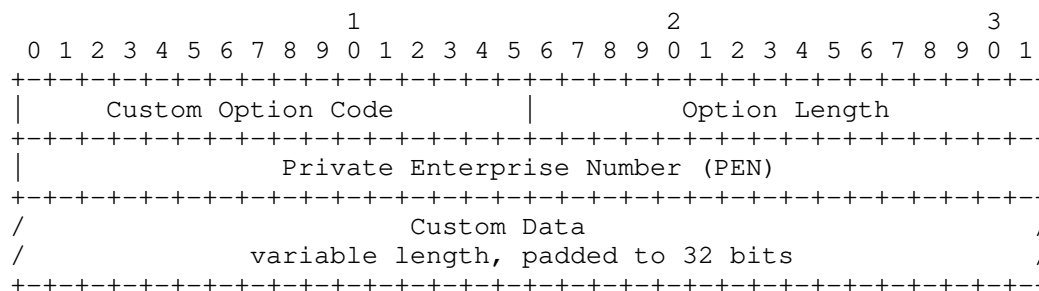


Figure 8: Custom Options Format

The Custom Option has the following fields:

- \* Custom Option Code: The code number for the Custom Option, which can be one of the following decimal numbers:

2988:

This option code identifies a Custom Option containing a UTF-8 string in the Custom Data portion. The string is not zero-terminated. This Custom Option can be safely copied to a new file if the pcapng file is manipulated by an application; otherwise 19372 should be used instead. See Section 5.2 for details.

2989:

This option code identifies a Custom Option containing binary octets in the Custom Data portion. This Custom Option can be safely copied to a new file if the pcapng file is manipulated by an application; otherwise 19372 should be used instead. See Section 5.2 for details.

19372:

This option code identifies a Custom Option containing a UTF-8 string in the Custom Data portion. The string is not zero-terminated. This Custom Option should not be copied to a new file if the pcapng file is manipulated by an application. See Section 5.2 for details.

19373:

This option code identifies a Custom Option containing binary octets in the Custom Data portion. This Custom Option should not be copied to a new file if the pcapng file is manipulated by an application. See Section 5.2 for details.

- \* Option Length: as described in Section 3.1, this contains the length of the option's value, which includes the 4-octet Private Enterprise Number and variable-length Custom Data fields, without the padding octets.
- \* Private Enterprise Number: An IANA-assigned Private Enterprise Number identifying the organization which defined the Custom Option. See Section 5.1 for details. The PEN MUST be encoded using the same endianness as the Section Header Block it is within the scope of.
- \* Custom Data: the custom data, padded to a 32 bit boundary.

### 3.6. Data format

#### 3.6.1. Endianness

Data contained in each section will always be saved according to the characteristics (little endian / big endian) of the capturing machine. This refers to all the fields that are saved as numbers and that span over two or more octets.

The approach of having each section saved in the native format of the generating host is more efficient because it avoids translation of data when reading / writing on the host itself, which is the most common case when generating/processing capture captures.

Please note: The endianness is indicated by the Section Header Block (Section 4.1). Since this block can appear several times in a pcapng file, a single file can contain both endianness variants.

### 3.6.2. Alignment

All fields of this specification use proper alignment for 16- and 32-bit values. This makes it easier and faster to read/write file contents if using techniques like memory mapped files.

The alignment octets (marked in this document e.g. with "padded to 32 bits") MUST be filled with zeroes.

Please note: 64-bit values are not aligned to 64-bit boundaries. This is because the file is naturally aligned to 32-bit boundaries only. Special care MUST be taken when reading and writing such values. (Note also that some 64-bit values are represented as a 64-bit integer in the endianness of the machine that wrote the file, and others are represented as 2 32-bit values, one containing the upper 32 bits of the value and one containing the lower 32 bits of the value, each written as 32-bit integers in the endianness of the machine that wrote the file. Neither of these formats guarantee 64-bit alignment.)

## 4. Block Definition

This section details the format of the blocks currently defined.

### 4.1. Section Header Block

The Section Header Block (SHB) is mandatory. It identifies the beginning of a section of the capture file. The Section Header Block does not contain data but it rather identifies a list of blocks (interfaces, packets) that are logically correlated. Its format is shown in Figure 9.

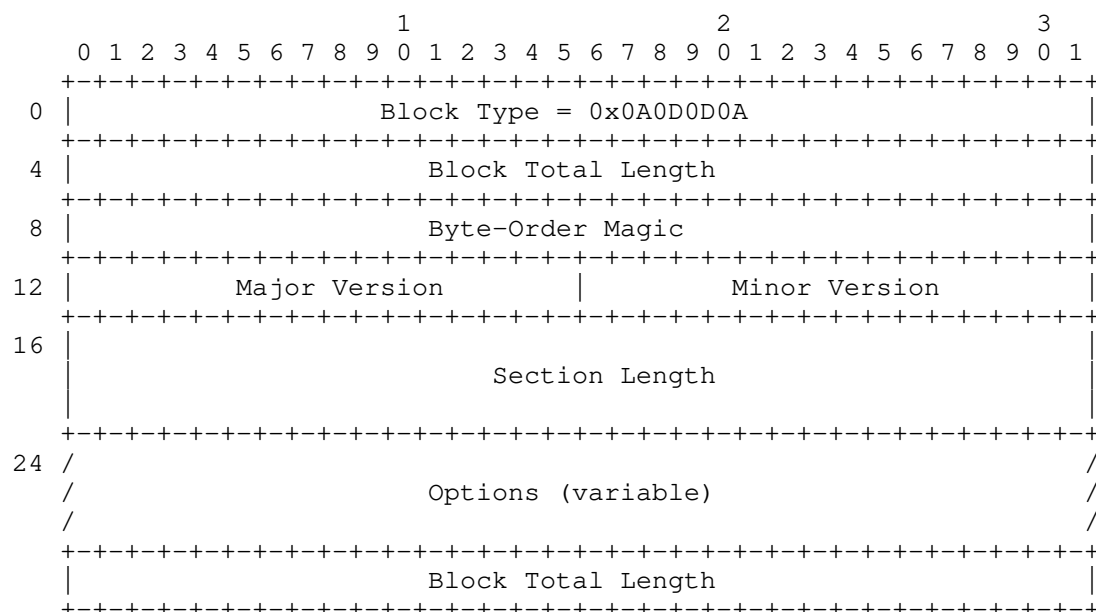


Figure 9: Section Header Block Format

The meaning of the fields is:

- \* **Block Type:** The block type of the Section Header Block is the integer corresponding to the 4-char string "\n\r\r\n" (0x0A0D0D0A). This particular value is used for 2 reasons:
  1. This number is used to detect if a file has been transferred via FTP or HTTP from a machine to another with an inappropriate ASCII conversion. In this case, the value of this field will differ from the standard one ("\n\r\r\n") and the reader can detect a possibly corrupted file.
  2. This value is palindromic, so that the reader is able to recognize the Section Header Block regardless of the endianness of the section. The endianness is recognized by reading the Byte Order Magic, which is located 8 octets after the Block Type.
- \* **Block Total Length:** total size of this block, as described in Section 3.1.

- \* Byte-Order Magic (32 bits): an unsigned magic number, whose value is the hexadecimal number 0x1A2B3C4D. This number can be used to distinguish sections that have been saved on little-endian machines from the ones saved on big-endian machines, and to heuristically identify pcapng files.
- \* Major Version (16 bits): an unsigned value, giving the number of the current major version of the format. The value for the current version of the format is 1.
- \* Minor Version (16 bits): an unsigned value, giving the number of the current minor version of the format. The value for the current version of the format is 0.
- \* Section Length (64 bits): a signed value specifying the length in octets of the following section, excluding the Section Header Block itself. This field can be used to skip the section, for faster navigation inside large files. If the Section Length is -1 (0xFFFFFFFFFFFFFFFF), this means that the size of the section is not specified, and the only way to skip the section is to parse the blocks that it contains. Please note that if this field is valid (i.e. not negative), its value is always a multiple of 4, as all the blocks are aligned to and padded to 32-bit (4 octet) boundaries. Also, special care should be taken in accessing this field: since the alignment of all the blocks in the file is 32-bits, this field is not guaranteed to be aligned to a 64-bit boundary. This could be a problem on 64-bit processors.
- \* Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

Writers of pcapng files MUST NOT write SHBs with a Major Version other than 1 or a Minor Version other than 0. If they do so, they will write a file that many readers of pcapng files, such as programs using libpcap to read pcapng files (including, but not limited to, tcpdump), Wireshark, and possibly other programs not to be able to read their files.

Some pcapng file writers have used a minor version of 2, but the file format did not change incompatibly (new block types were added); Readers of pcapng files MUST treat a Minor Version of 2 as equivalent to a Minor Version of 0 (and, if they also write a pcapng file based on the results of reading one or more pcapng files, they MUST NOT, as per the previous sentence, write an SHB with a Minor Version of 2, even if they read an SHB with a Minor Version of 2). As indicated above, using a minor version number other than 0 when writing a section of a pcapng file will produce a section that most existing software will not be able to read; future versions of some of that

software will be able to read sections with a version of 1.2, but older copies of that software that are not updated to the latest version will still not be able to read them.

The Major Version would be changed only if a new version of this specification, for a later major version of the file format, were created. Such a version would only be created if the format were to change in such a way that code that reads the new format could not read the old format (i.e., code to read both formats would have to check the version number and use different code paths for the two formats) and code that reads the old format could not read the new format. An incompatible change to the format of an existing block or an existing option would be such a change; the addition of a new block or a new option would not be such a change. An example of such an incompatible change would be the addition of an additional field to the Section Header Block, following the Minor Version field and before the Snaplen field; software expecting the new SHB format would not correctly read the old SHB format, and software expecting the old SHB format would not correctly read the new SHB format. (Note that a change to the SHB must leave the Block Type, Block Total Length, Byte-Order Magic, Major Version, and Minor Version fields at the same offsets from the beginning of the SHB and with the same lengths, must keep the value of the Block Type the same, must keep the two possible values of the Byte-Order Magic the same, depending on the block's byte order, so that the rest of the SHB can be correctly interpreted.)

The Minor Version would be changed only if a new version of this specification, for a later minor version of the file format, were created. Such a version would only be created if the format were to change in such a way that code that reads the new format could read the old format without checking the version number but code that reads the old format could not read all files in the new format. A backward-compatible change to the format of an existing block or an existing option would be such a change; the addition of a new block or a new option would not be such a change. An example of such a backward-compatible but not forward-compatible change would be a change to the Interface Description block (see below) to use the current Reserved field to indicate the presence of additional fields before the Options, with a zero value indicate no such fields are present.

I.e., adding new block types or options would not require that either the Major Version or the Minor Version be changed, as code that does not know about the block type or option should just skip it; only if skipping a block or option does not work should the minor version number be changed.



Aside from the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
shb_hardware	2	variable	no
shb_os	3	variable	no
shb_userappl	4	variable	no

Table 2: Section Header Block Options

#### shb\_hardware:

The shb\_hardware option is a UTF-8 string containing the description of the hardware used to create this section. The string is not zero-terminated.

Examples: "x86 Personal Computer", "Sun Sparc Workstation".

#### shb\_os:

The shb\_os option is a UTF-8 string containing the name of the operating system used to create this section. The string is not zero-terminated.

Examples: "Windows XP SP2", "openSUSE 10.2".

#### shb\_userappl:

The shb\_userappl option is a UTF-8 string containing the name of the application used to create this section. The string is not zero-terminated.

Examples: "dumpcap V0.99.7".

[Open issue: does a program which re-writes a capture file change the original hardware/os/application info?]

## 4.2. Interface Description Block

An Interface Description Block (IDB) is the container for information describing an interface on which packet data is captured.

Tools that write / read the capture file associate an incrementing unsigned 32-bit number (starting from '0') to each Interface Definition Block, called the Interface ID for the interface in question. This number is unique within each Section and identifies

the interface to which the IDB refers; it is only unique inside the current section, so, two Sections can have different interfaces identified by the same Interface ID values. This unique identifier is referenced by other blocks, such as Enhanced Packet Blocks and Interface Statistic Blocks, to indicate the interface to which the block refers (such the interface that was used to capture the packet that an Enhanced Packet Block contains or to which the statistics in an Interface Statistic Block refer).

There must be an Interface Description Block for each interface to which another block refers. Blocks such as an Enhanced Packet Block or an Interface Statistics Block contain an Interface ID value referring to a particular interface, and a Simple Packet Block implicitly refers to an interface with an Interface ID of 0. If the file does not contain any blocks that use an Interface ID, then the file does not need to have any IDBs.

An Interface Description Block is valid only inside the section to which it belongs. The structure of a Interface Description Block is shown in Figure 10.

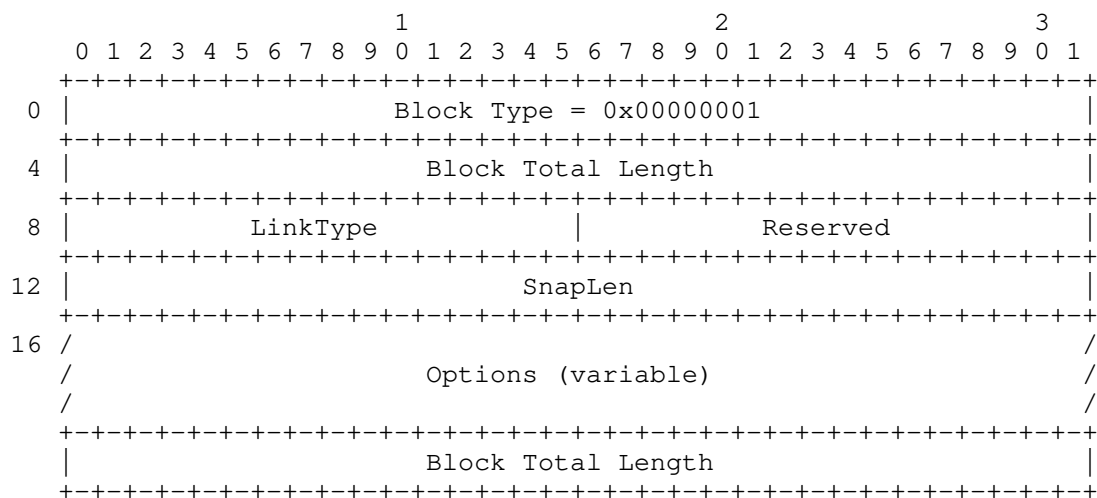


Figure 10: Interface Description Block Format

The meaning of the fields is:

- \* Block Type: The block type of the Interface Description Block is 1.
- \* Block Total Length: total size of this block, as described in Section 3.1.

- \* LinkType (16 bits): an unsigned value that defines the link layer type of this interface. The list of Standardized Link Layer Type codes is available in [LINKTYPES].
- \* Reserved (16 bits): not used - MUST be filled with 0 by pcapng file writers, and MUST be ignored by pcapng file readers.
- \* SnapLen (32 bits): an unsigned value indicating the maximum number of octets captured from each packet. The portion of each packet that exceeds this value will not be stored in the file. A value of zero indicates no limit.
- \* Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
if_name	2	variable	no
if_description	3	variable	no
if_IPv4addr	4	8	yes
if_IPv6addr	5	17	yes
if_MACaddr	6	6	no
if_EUIaddr	7	8	no
if_speed	8	8	no
if_tsresol	9	1	no
if_tzone	10	4	no
if_filter	11	variable, minimum 1	no
if_os	12	variable	no
if_fcslen	13	1	no
if_tsoffset	14	8	no
if_hardware	15	variable	no
if_txspeed	16	8	no
if_rxspeed	17	8	no

Table 3: Interface Description Block Options

**if\_name:**

The if\_name option is a UTF-8 string containing the name of the device used to capture data. The string is not zero-terminated.

Examples: "eth0",

"\Device\NPF\_{AD1CE675-96D0-47C5-ADD0-2504B9126B68}"/>.

**if\_description:**

The `if_description` option is a UTF-8 string containing the description of the device used to capture data. The string is not zero-terminated.

Examples: "Wi-Fi", "Local Area Connection", "Wireless Network Connection", "First Ethernet Interface".

`if_IPv4addr`:

The `if_IPv4addr` option is an IPv4 network address and corresponding netmask for the interface. The first four octets are the IP address, and the next four octets are the netmask. This option can be repeated multiple times within the same Interface Description Block when multiple IPv4 addresses are assigned to the interface. Note that the IP address and netmask are both treated as four octets, one for each octet of the address or mask; they are not 32-bit numbers, and thus the endianness of the SHB does not affect this field's value.

Examples: '192 168 1 1 255 255 255 0'.

`if_IPv6addr`:

The `if_IPv6addr` option is an IPv6 network address and corresponding prefix length for the interface. The first 16 octets are the IP address and the next octet is the prefix length. This option can be repeated multiple times within the same Interface Description Block when multiple IPv6 addresses are assigned to the interface.

Example: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344/64 is written (in hex) as '20 01 0d b8 85 a3 08 d3 13 19 8a 2e 03 70 73 44 40'.

`if_MACaddr`:

The `if_MACaddr` option is the Interface Hardware MAC address (48 bits), if available.

Example: '00 01 02 03 04 05'.

`if_EUIaddr`:

The `if_EUIaddr` option is the Interface Hardware EUI address (64 bits), if available.

Example: '02 34 56 FF FE 78 9A BC'.

`if_speed`:

The `if_speed` option is a 64-bit unsigned value indicating the interface speed, in bits per second.

Example: the 64-bit decimal number 100000000 for 100Mbps.

**if\_tsresol:**

The if\_tsresol option identifies the resolution of timestamps. If the Most Significant Bit is equal to zero, the remaining bits indicates the resolution of the timestamp as a negative power of 10 (e.g. 6 means microsecond resolution, timestamps are the number of microseconds since 1970-01-01 00:00:00 UTC). If the Most Significant Bit is equal to one, the remaining bits indicates the resolution as negative power of 2 (e.g. 10 means 1/1024 of second). If this option is not present, a resolution of  $10^{-6}$  is assumed (i.e. timestamps have the same resolution of the standard 'libpcap' timestamps).

Example: '6'.

**if\_tzone:**

The if\_tzone option identifies the time zone for GMT support (TODO: specify better).

Example: TODO: give a good example.

**if\_filter:**

The if\_filter option identifies the filter (e.g. "capture only TCP traffic") used to capture traffic. The first octet of the Option Data keeps a code of the filter used (e.g. if this is a libpcap string, or BPF bytecode, and more). More details about this format will be presented in Appendix XXX (TODO). (TODO: better use different options for different fields? e.g. if\_filter\_pcap, if\_filter\_bpf, ...)

Example: '00'"tcp port 23 and host 192.0.2.5".

**if\_os:**

The if\_os option is a UTF-8 string containing the name of the operating system of the machine in which this interface is installed. This can be different from the same information that can be contained by the Section Header Block (Section 4.1) because the capture can have been done on a remote machine. The string is not zero-terminated.

Examples: "Windows XP SP2", "openSUSE 10.2".

**if\_fcslen:**

The if\_fcslen option is an 8-bit unsigned integer value that specifies the length of the Frame Check Sequence (in bits) for this interface. For link layers whose FCS length can

change during time, the Enhanced Packet Block `epb_flags` Option can be used in each Enhanced Packet Block (see Section 4.3.1).

Example: '4'.

`if_tsoffset:`

The `if_tsoffset` option is a 64-bit signed integer value that specifies an offset (in seconds) that must be added to the timestamp of each packet to obtain the absolute timestamp of a packet. If the option is missing, the timestamps stored in the packet MUST be considered absolute timestamps. The time zone of the offset can be specified with the option `if_tzone`. TODO: won't a `if_tsoffset_low` for fractional second offsets be useful for highly synchronized capture systems?

Example: '1234'.

`if_hardware:`

The `if_hardware` option is a UTF-8 string containing the description of the interface hardware. The string is not zero-terminated.

Examples: "Broadcom NetXtreme", "Intel(R) PRO/1000 MT Network Connection", "NETGEAR WNA1000Mv2 N150 Wireless USB Micro Adapter".

`if_txspeed:`

The `if_txrxspeeds` option is a 64-bit unsigned value indicating the interface transmit speed in bits per second.

Example: the 64-bit decimal number 1024000 for 1024Kbps.

`if_rxspeed:`

The `if_rxspeed` option is a 64-bit unsigned value indicating the interface receive speed, in bits per second.

Example: the 64-bit decimal number 8192000 for 8192Kbps.

If the interface transmit speed and receive speed are the same, the `if_speed` option MUST be used and the `if_txspeed` and `if_rxspeed` options MUST NOT be used. If the transmit speed is unknown, the `if_speed` and `if_txspeed` options MUST NOT be used; if the receive speed is unknown, the `if_speed` and `if_rxspeed` options MUST NOT be used.

#### 4.3. Enhanced Packet Block

An Enhanced Packet Block (EPB) is the standard container for storing the packets coming from the network. The Enhanced Packet Block is optional because packets can be stored either by means of this block or the Simple Packet Block, which can be used to speed up capture file generation; or a file may have no packets in it. The format of an Enhanced Packet Block is shown in Figure 11.

The Enhanced Packet Block is an improvement over the original, now obsolete, Packet Block (Appendix A):

- \* it stores the Interface Identifier as a 32-bit integer value. This is a requirement when a capture stores packets coming from a large number of interfaces;
- \* unlike the Packet Block (Appendix A), the number of packets dropped by the capture system between this packet and the previous one is not stored in the header, but rather in an option of the block itself.



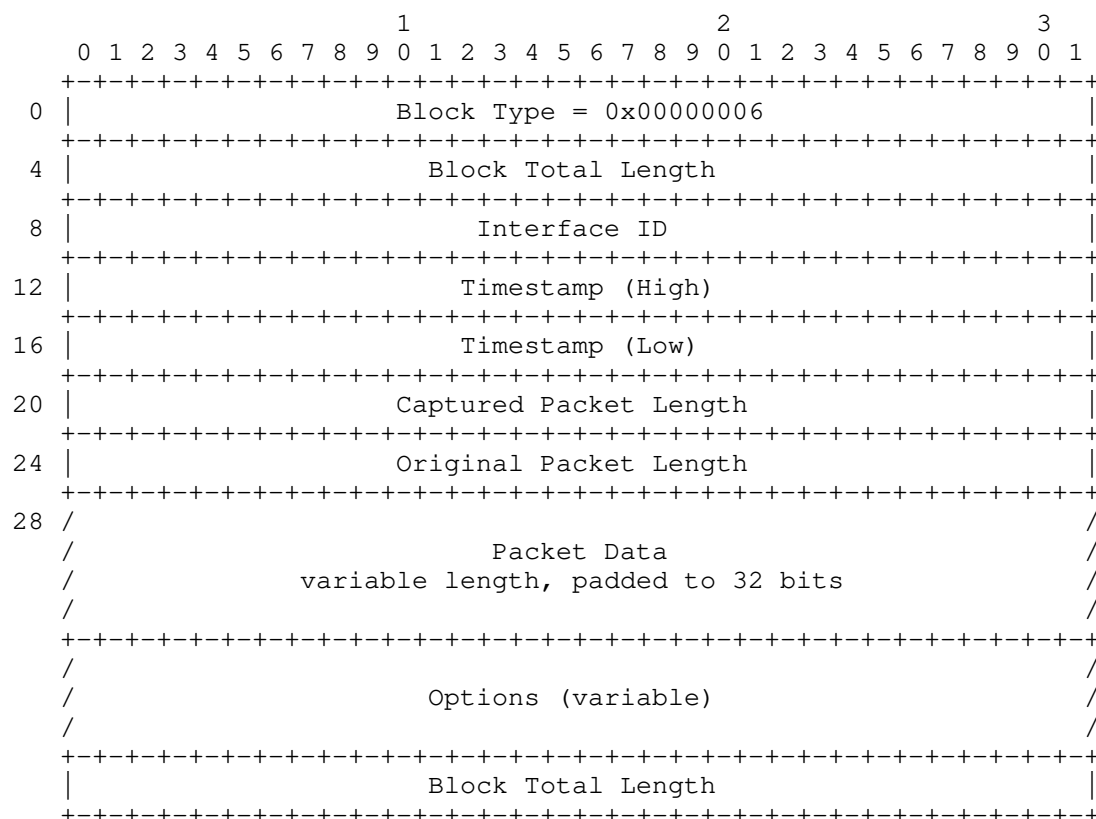


Figure 11: Enhanced Packet Block Format

The Enhanced Packet Block has the following fields:

- \* Block Type: The block type of the Enhanced Packet Block is 6.
- \* Block Total Length: total size of this block, as described in Section 3.1.
- \* Interface ID (32 bits): an unsigned value that specifies the interface on which this packet was received or transmitted; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by the same number (see Section 4.2) of this field. The interface ID MUST be valid, which means that an matching interface description block MUST exist.

- \* **Timestamp (High) and Timestamp (Low):** upper 32 bits and lower 32 bits of a 64-bit timestamp. The timestamp is a single 64-bit unsigned integer that represents the number of units of time that have elapsed since 1970-01-01 00:00:00 UTC. The length of a unit of time is specified by the 'if\_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet. Note that, unlike timestamps in the libpcap file format, timestamps in Enhanced Packet Blocks are not saved as two 32-bit values that represent the seconds and microseconds that have elapsed since 1970-01-01 00:00:00 UTC. Timestamps in Enhanced Packet Blocks are saved as two 32-bit words that represent the upper and lower 32 bits of a single 64-bit quantity.
- \* **Captured Packet Length (32 bits):** an unsigned value that indicates the number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 10). The value of this field does not include the padding octets added at the end of the Packet Data field to align the Packet Data field to a 32-bit boundary.
- \* **Original Packet Length (32 bits):** an unsigned value that indicates the actual length of the packet when it was transmitted on the network. It can be different from the Captured Packet Length if the packet has been truncated by the capture process.
- \* **Packet Data:** the data coming from the network, including link-layer headers. The actual length of this field is Captured Packet Length plus the padding to a 32-bit boundary. The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].
- \* **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
epb_flags	2	4	no
epb_hash	3	variable, minimum hash type-dependent	yes
epb_dropcount	4	8	no
epb_packetid	5	8	no
epb_queue	6	4	no
epb_verdict	7	variable, minimum verdict type-dependent	yes

Table 4: Enhanced Packet Block Options

**epb\_flags:**

The `epb_flags` option is a 32-bit flags word containing link-layer information. A complete specification of the allowed flags can be found in Section 4.3.1.

Example: '0'.

**epb\_hash:**

The `epb_hash` option contains a hash of the packet. The first octet specifies the hashing algorithm, while the following octets contain the actual hash, whose size depends on the hashing algorithm, and hence from the value in the first octet. The hashing algorithm can be: 2s complement (algorithm octet = 0, size = XXX), XOR (algorithm octet = 1, size=XXX), CRC32 (algorithm octet = 2, size = 4), MD-5 (algorithm octet = 3, size = 16), SHA-1 (algorithm octet = 4, size = 20), Toeplitz (algorithm octet = 5, size = 4). The hash covers only the packet, not the header added by the capture driver: this gives the possibility to calculate it inside the network card. The hash allows easier comparison/merging of different capture files, and reliable data transfer between the data acquisition system and the capture library.

Examples: '02 EC 1D 87 97', '03 45 6E C2 17 7C 10 1E 3C 2E 99 6E C2 9A 3D 50 8E'.

**epb\_dropcount:**

The `epb_dropcount` option is a 64-bit unsigned integer value specifying the number of packets lost (by the interface and the operating system) between this packet and the preceding one for the same interface or, for the first packet for an interface, between this packet and the start of the capture process.

Example: '0'.

**epb\_packetid:**

The `epb_packetid` option is a 64-bit unsigned integer that uniquely identifies the packet. If the same packet is seen by multiple interfaces and there is a way for the capture application to correlate them, the same `epb_packetid` value must be used. An example could be a router that captures packets on all its interfaces in both directions. When a packet hits interface A on ingress, an EPB entry gets created, TTL gets decremented, and right before it egresses on interface B another EPB entry gets created in the trace file. In this case, two packets are in the capture file, which are not identical but the `epb_packetid` can be used to correlate them.

Example: '0'.

**epb\_queue:**

The `epb_queue` option is a 32-bit unsigned integer that identifies on which queue of the interface the specific packet was received.

Example: '0'.

**epb\_verdict:**

The `epb_verdict` option stores a verdict of the packet. The verdict indicates what would be done with the packet after processing it. For example, a firewall could drop the packet. This verdict can be set by various components, i.e. Hardware, Linux's eBPF TC or XDP framework, etc. etc. The first octet specifies the verdict type, while the following octets contain the actual verdict data, whose size depends on the verdict type, and hence from the value in the first octet. The verdict type can be: Hardware (type octet = 0, size = variable), Linux\_eBPF\_TC (type octet = 1, size = 8 (64-bit unsigned integer), value = TC\_ACT\_\* as defined in the Linux `pkt_cls.h` ([https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/include/uapi/linux/pkt\\_cls.h](https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/include/uapi/linux/pkt_cls.h))), Linux\_eBPF\_XDP (type octet = 2, size = 8 (64-bit

unsigned integer), value = xdp\_action as defined in the Linux pbf.h (<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/include/uapi/linux/bpf.h>) include).

Example: '02 00 00 00 00 00 00 00 02' for Linux\_eBPF\_XDP with verdict XDP\_PASS.

#### 4.3.1. Enhanced Packet Block Flags Word

The Enhanced Packet Block Flags Word is a 32-bit value that contains link-layer information about the packet.

The word is encoded as an unsigned 32-bit integer, using the endianness of the Section Header Block scope it is in. In the following table, the bits are numbered with 0 being the least-significant bit and 31 being the most-significant bit of the 32-bit unsigned integer. The meaning of the bits is the following:

Bit Number	Description
0-1	Inbound / Outbound packet (00 = information not available, 01 = inbound, 10 = outbound)
2-4	Reception type (000 = not specified, 001 = unicast, 010 = multicast, 011 = broadcast, 100 = promiscuous).
5-8	FCS length, in octets (0000 if this information is not available). This value overrides the if_fcslen option of the Interface Description Block, and is used with those link layers (e.g. PPP) where the length of the FCS can change during time.
9-15	Reserved (MUST be set to zero).
16-31	link-layer-dependent errors (Bit 31 = symbol error, Bit 30 = preamble error, Bit 29 = Start Frame Delimiter error, Bit 28 = unaligned frame error, Bit 27 = wrong Inter Frame Gap error, Bit 26 = packet too short error, Bit 25 = packet too long error, Bit 24 = CRC error, other?? are 16 bit enough?).

Table 5

NOTE: in earlier versions of this specification, the bits were specified as being numbered with 0 being the most-significant bit and 31 being the least-significant bit of the 32-bit unsigned integer, rather than with 0 being the least-significant bit and 31 being the most-significant bit. Several implementations number the bits with 0 being the least-significant bit, and no known implementations number them with 0 being the most-significant bit, so the specification was changed to reflect that reality.

#### 4.4. Simple Packet Block

The Simple Packet Block (SPB) is a lightweight container for storing the packets coming from the network. Its presence is optional.

A Simple Packet Block is similar to an Enhanced Packet Block (see Section 4.3), but it is smaller, simpler to process and contains only a minimal set of information. This block is preferred to the standard Enhanced Packet Block when performance or space occupation are critical factors, such as in sustained traffic capture applications. A capture file can contain both Enhanced Packet Blocks and Simple Packet Blocks: for example, a capture tool could switch from Enhanced Packet Blocks to Simple Packet Blocks when the hardware resources become critical.

The Simple Packet Block does not contain the Interface ID field. Therefore, it MUST be assumed that all the Simple Packet Blocks have been captured on the interface previously specified in the first Interface Description Block.

Figure 12 shows the format of the Simple Packet Block.

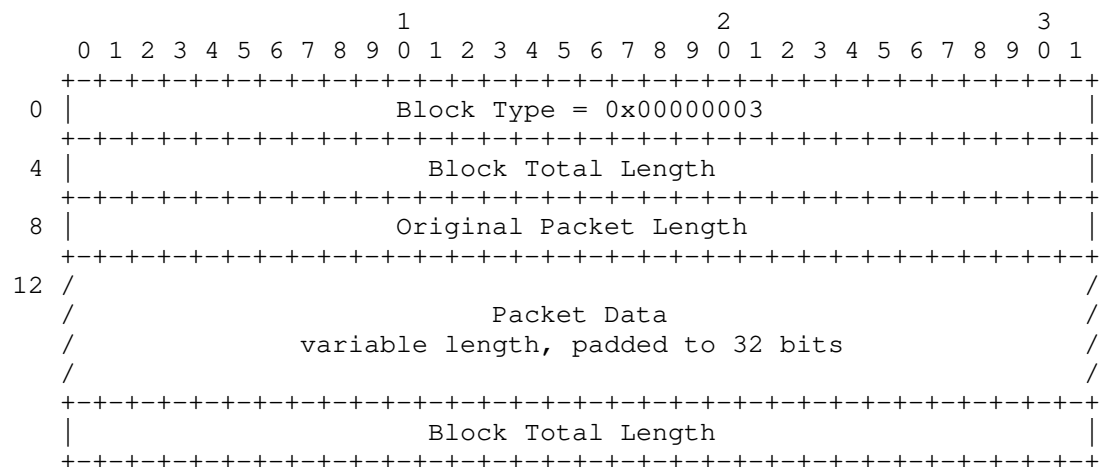


Figure 12: Simple Packet Block Format

The Simple Packet Block has the following fields:

- \* Block Type: The block type of the Simple Packet Block is 3.
- \* Block Total Length: total size of this block, as described in Section 3.1.
- \* Original Packet Length (32 bits): an unsigned value indicating the actual length of the packet when it was transmitted on the network. It can be different from length of the Packet Data field's length if the packet has been truncated by the capture process, in which case the SnapLen value in Section 4.2 will be less than this Original Packet Length value, and the SnapLen value MUST be used to determine the size of the Packet Data field length.
- \* Packet Data: the data coming from the network, including link-layer headers. The length of this field can be derived from the field Block Total Length, present in the Block Header, and it is the minimum value among the SnapLen (present in the Interface Description Block) and the Original Packet Length (present in this header). The format of the data within this Packet Data field depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].

The Simple Packet Block does not contain the timestamp because this is often one of the most costly operations on PCs. Additionally, there are applications that do not require it; e.g. an Intrusion Detection System is interested in packets, not in their timestamp.

A Simple Packet Block cannot be present in a Section that has more than one interface because of the impossibility to refer to the correct one (it does not contain any Interface ID field).

The Simple Packet Block is very efficient in term of disk space: a snapshot whose length is 100 octets requires only 16 octets of overhead, which corresponds to an efficiency of more than 86%.

#### 4.5. Name Resolution Block

The Name Resolution Block (NRB) is used to support the correlation of numeric addresses (present in the captured packets) and their corresponding canonical names and it is optional. Having the literal names saved in the file prevents the need for performing name resolution at a later time, when the association between names and addresses may be different from the one in use at capture time. Moreover, the NRB avoids the need for issuing a lot of DNS requests every time the trace capture is opened, and also provides name resolution when reading the capture with a machine not connected to the network.

A Name Resolution Block is often placed at the beginning of the file, but no assumptions can be taken about its position. Multiple NRBs can exist in a pcapng file, either due to memory constraints or because additional name resolutions were performed by file processing tools, like network analyzers.

A Name Resolution Block need not contain any Records, except the `nrb_record_end` Record which MUST be the last Record. The addresses and names in NRB Records MAY be repeated multiple times; i.e., the same IP address may resolve to multiple names, the same name may resolve to the multiple IP addresses, and even the same address-to-name pair may appear multiple times, in the same NRB or across NRBs.

The format of the Name Resolution Block is shown in Figure 13.



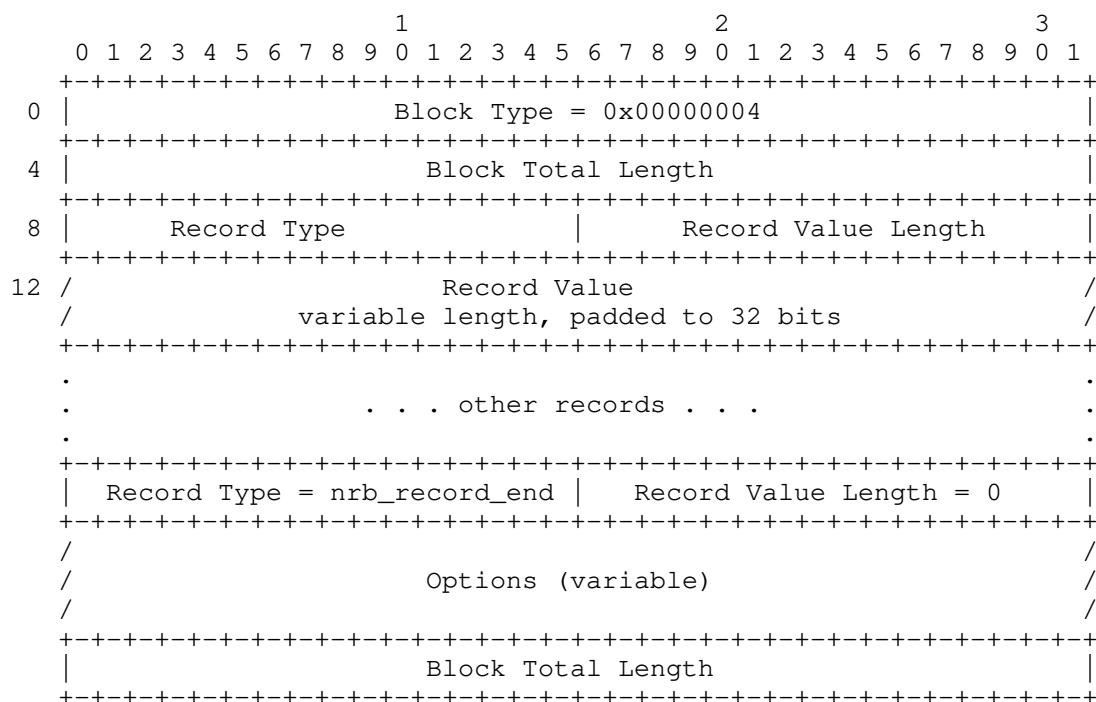


Figure 13: Name Resolution Block Format

The Name Resolution Block has the following fields:

- \* Block Type: The block type of the Name Resolution Block is 4.
- \* Block Total Length: total size of this block, as described in Section 3.1.

This is followed by zero or more Name Resolution Records (in the TLV format), each of which contains an association between a network address and a name. An `nrb_record_end` MUST be added after the last Record, and MUST exist even if there are no other Records in the NRB. There are currently three possible types of records:

Name	Code	Length
nrb_record_end	0x0000	0
nrb_record_ipv4	0x0001	variable
nrb_record_ipv6	0x0002	variable

Table 6: Name Resolution Block Records

**nrb\_record\_end:**

The nrb\_record\_end record delimits the end of name resolution records. This record is needed to determine when the list of name resolution records has ended and some options (if any) begin.

**nrb\_record\_ipv4:**

The nrb\_record\_ipv4 record specifies an IPv4 address (contained in the first 4 octets), followed by one or more zero-terminated UTF-8 strings containing the DNS entries for that address. The minimum valid Record Length for this Record Type is thus 6: 4 for the IP octets, 1 character, and a zero-value octet terminator. Note that the IP address is treated as four octets, one for each octet of the IP address; it is not a 32-bit word, and thus the endianness of the SHB does not affect this field's value.

Example: '127 0 0 1'"localhost".

[Open issue: is an empty string (i.e., just a zero-value octet) valid?]

**nrb\_record\_ipv6:**

The nrb\_record\_ipv6 record specifies an IPv6 address (contained in the first 16 octets), followed by one or more zero-terminated strings containing the DNS entries for that address. The minimum valid Record Length for this Record Type is thus 18: 16 for the IP octets, 1 character, and a zero-value octet terminator.

Example: '20 01 0d b8 00 00 00 00 00 00 00 00 12 34 56 78'"somehost".

[Open issue: is an empty string (i.e., just a zero-value octet) valid?]

Record Types other than those specified earlier MUST be ignored and skipped past. More Record Types will likely be defined in the future, and MUST NOT break backwards compatibility.

Each Record Value is aligned to and padded to a 32-bit boundary. The corresponding Record Value Length reflects the actual length of the Record Value; it does not include the lengths of the Record Type field, the Record Value Length field, any padding for the Record Value, or anything after the Record Value. For Record Types with name strings, the Record Length does include the zero-value octet terminating that string. A Record Length of 0 is valid, unless indicated otherwise.

After the list of Name Resolution Records, optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
ns_dnsname	2	variable	no
ns_dnsIP4addr	3	4	no
ns_dnsIP6addr	4	16	no

Table 7: Name Resolution Block Options

#### ns\_dnsname:

The ns\_dnsname option is a UTF-8 string containing the name of the machine (DNS server) used to perform the name resolution. The string is not zero-terminated.

Example: "our\_nameserver".

#### ns\_dnsIP4addr:

The ns\_dnsIP4addr option specifies the IPv4 address of the DNS server. Note that the IP address is treated as four octets, one for each octet of the IP address; it is not a 32-bit word, and thus the endianness of the SHB does not affect this field's value.

Example: '192 168 0 1'.

ns\_dnsIP6addr:

The ns\_dnsIP6addr option specifies the IPv6 address of the DNS server.

Example: '20 01 0d b8 00 00 00 00 00 00 00 00 12 34 56 78'.

#### 4.6. Interface Statistics Block

The Interface Statistics Block (ISB) contains the capture statistics for a given interface and it is optional. The statistics are referred to the interface defined in the current Section identified by the Interface ID field. An Interface Statistics Block is normally placed at the end of the file, but no assumptions can be taken about its position - it can even appear multiple times for the same interface.

The format of the Interface Statistics Block is shown in Figure 14.

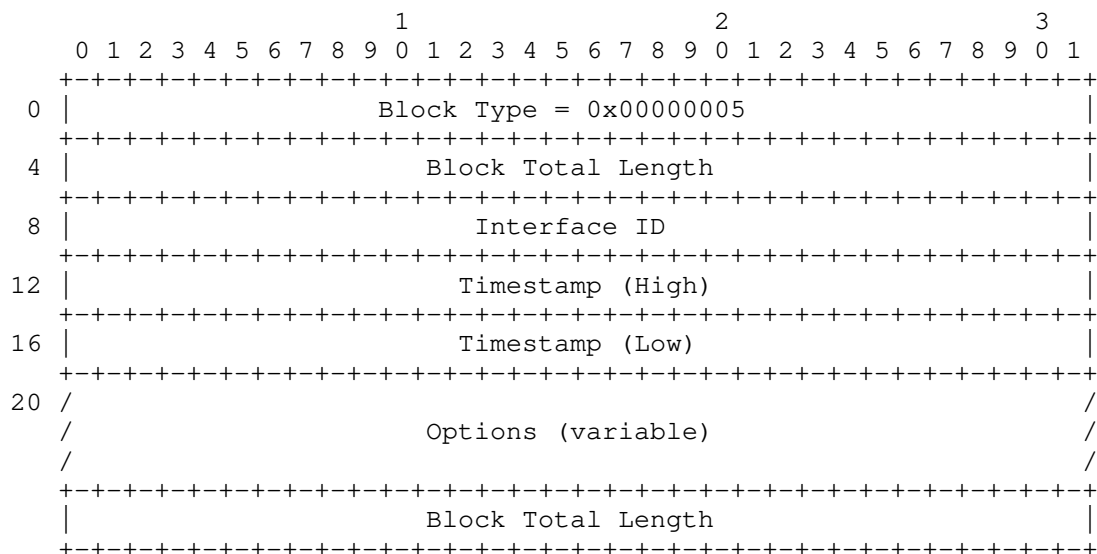


Figure 14: Interface Statistics Block Format

The fields have the following meaning:

- \* Block Type: The block type of the Interface Statistics Block is 5.
- \* Block Total Length: total size of this block, as described in Section 3.1.

- \* Interface ID: specifies the interface these statistics refers to; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by same number (see Section 4.2) of this field.
- \* Timestamp: time this statistics refers to. The format of the timestamp is the same already defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if\_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.
- \* Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

All the statistic fields are defined as options in order to deal with systems that do not have a complete set of statistics. Therefore, In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
isb_starttime	2	8	no
isb_endtime	3	8	no
isb_ifrecv	4	8	no
isb_ifdrop	5	8	no
isb_filteraccept	6	8	no
isb_osdrop	7	8	no
isb_usrdeliv	8	8	no

Table 8: Interface Statistics Block Options

#### isb\_starttime:

The isb\_starttime option specifies the time the capture started; time will be stored in two blocks of four octets each. The format of the timestamp is the same as the one defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if\_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.

Example: '96 c3 04 00 73 89 6a 65', in Little Endian, decodes to 2012-06-29 06:17:00.834163 UTC.

**isb\_endtime:**

The `isb_endtime` option specifies the time the capture ended; time will be stored in two blocks of four octets each. The format of the timestamp is the same as the one defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if\_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.

Example: '97 c3 04 00 aa 47 ca 64', in Little Endian, decodes to 2012-06-29 07:28:25.298858 UTC.

**isb\_ifrecv:**

The `isb_ifrecv` option specifies the 64-bit unsigned integer number of packets received from the physical interface starting from the beginning of the capture.

Example: the decimal number 100.

**isb\_ifdrop:**

The `isb_ifdrop` option specifies the 64-bit unsigned integer number of packets dropped by the interface due to lack of resources starting from the beginning of the capture.

Example: '0'.

**isb\_filteraccept:**

The `isb_filteraccept` option specifies the 64-bit unsigned integer number of packets accepted by filter starting from the beginning of the capture.

Example: the decimal number 100.

**isb\_osdrop:**

The `isb_osdrop` option specifies the 64-bit unsigned integer number of packets dropped by the operating system starting from the beginning of the capture.

Example: '0'.

**isb\_usrdeliv:**

The `isb_usrdeliv` option specifies the 64-bit unsigned integer number of packets delivered to the user starting from the beginning of the capture. The value contained in this field can be different from the value `'isb_filteraccept - isb_osdrop'` because some packets could still be in the OS buffers when the capture ended.

Example: `'0'`.

All the fields that refer to packet counters are 64-bit values, represented with the octet order of the current section. Special care must be taken in accessing these fields: since all the blocks are aligned to a 32-bit boundary, such fields are not guaranteed to be aligned on a 64-bit boundary.

#### 4.7. Decryption Secrets Block

A Decryption Secrets Block (DSB) stores (session) secrets that enable decryption of packets within the capture file. The format of these secrets is defined by the Secrets Type.

Multiple DSBs can exist in a pcapng file, but they SHOULD be written before packet blocks that require those secrets. Tools MAY limit decryption to secrets that appear before packet blocks.

The structure of a Decryption Secrets Block is shown in Figure 15.

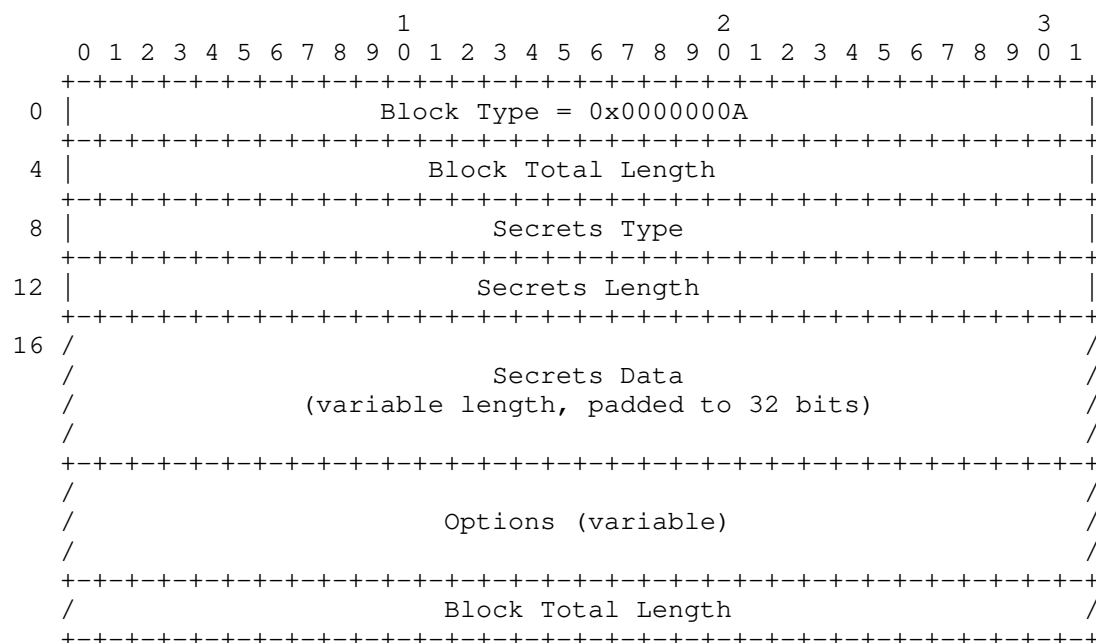


Figure 15: Decryption Secrets Block Format

The Decryption Secrets Block has the following fields.

- \* **Block Type:** The block type of the Decryption Secrets Block is 10.
- \* **Block Total Length:** total size of this block, as described in Section 3.1.
- \* **Secrets Type (32 bits):** an unsigned integer identifier that describes the format of the following Secrets field. Requests for new Secrets Type codes should be made by creating a pull request to update this document as described in Section 10.1.
- \* **Secrets Length (32 bits):** an unsigned integer that indicates the size of the following Secrets field, without any padding octets.
- \* **Secrets Data:** binary data containing secrets, padded to a 32 bit boundary.
- \* **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present. No DSB-specific options are currently defined.

The following is a list of Secrets Types.



**0x544c534b:**

TLS Key Log. This format is described at NSS Key Log Format ([https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key\\_Log\\_Format](https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format)). Every line MUST be properly terminated with either carriage return and linefeed ('\r\n') or linefeed ('\n'). Tools MUST be able to handle both line endings.

**0x57474b4c:**

WireGuard Key Log. Every line consists of the key type, equals sign ('='), and the base64-encoded 32-byte key with optional spaces before and in between. The key type is one of LOCAL\_STATIC\_PRIVATE\_KEY, REMOTE\_STATIC\_PUBLIC\_KEY, LOCAL\_EPHEMERAL\_PRIVATE\_KEY, or PRESHARED\_KEY. This matches the output of extract-handshakes.sh (<https://git.zx2c4.com/WireGuard/tree/contrib/examples/extract-handshakes/README>), which is part of the WireGuard (<https://www.wireguard.com/>) project. A PRESHARED\_KEY line is linked to a session matched by a previous LOCAL\_EPHEMERAL\_PRIVATE\_KEY line. Every line MUST be properly terminated with either carriage return and linefeed ('\r\n') or linefeed ('\n'). Tools MUST be able to handle both line endings.

Warning: LOCAL\_STATIC\_PRIVATE\_KEY and potentially PRESHARED\_KEY are long-term secrets, users SHOULD only store non-production keys, or ensure proper protection of the pcapng file.

**0x5a4e574b:**

ZigBee NWK Key and ZigBee PANID for that network. Network Key as described in the ZigBee Specification (<https://zigbeealliance.org/>) 05-3473-21 (R21) section 4.2.2. The NWK Key is a 16 octet binary AES-128 key used to secure NWK Level frames within a single PAN. The NWK key is immediately followed by the 2 octet (16 bit) network PANID in little endian format. If and when the NWK Key changes a new DSB will contain the new NWK Key.

**0x5a415053:**

ZigBee APS Key. Application Support Link Key as described in the ZigBee Specification (<https://zigbeealliance.org/>) 05-3473-21 (R21) section 4.4. Each 16 octet binary AES-128 key secures frames exchanged between a pair of network nodes. The APS Key is immediately followed by the 2 octet (16 bit) network PANID in little endian format. The PANID is followed by the 2 octet (16 bit) short addresses, in little endian format, of the nodes to which the APS Key applies. The numerically lower short address shall come first. There is a

APS Key DSB for each node pair for which the Link Key is known. As new links are formed, new DSBs contain the new Keys. If the APS Key changes for an existing link, it is contained in a new DSB with the new APS Key.

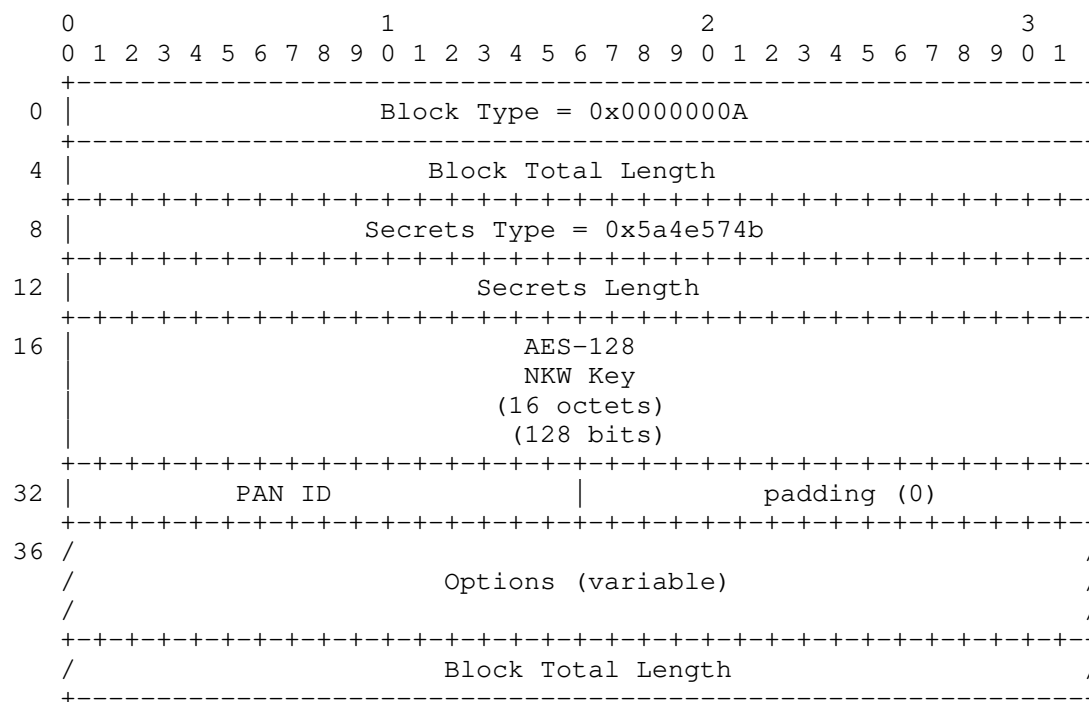


Figure 16: ZigBee NWK Key Data Format

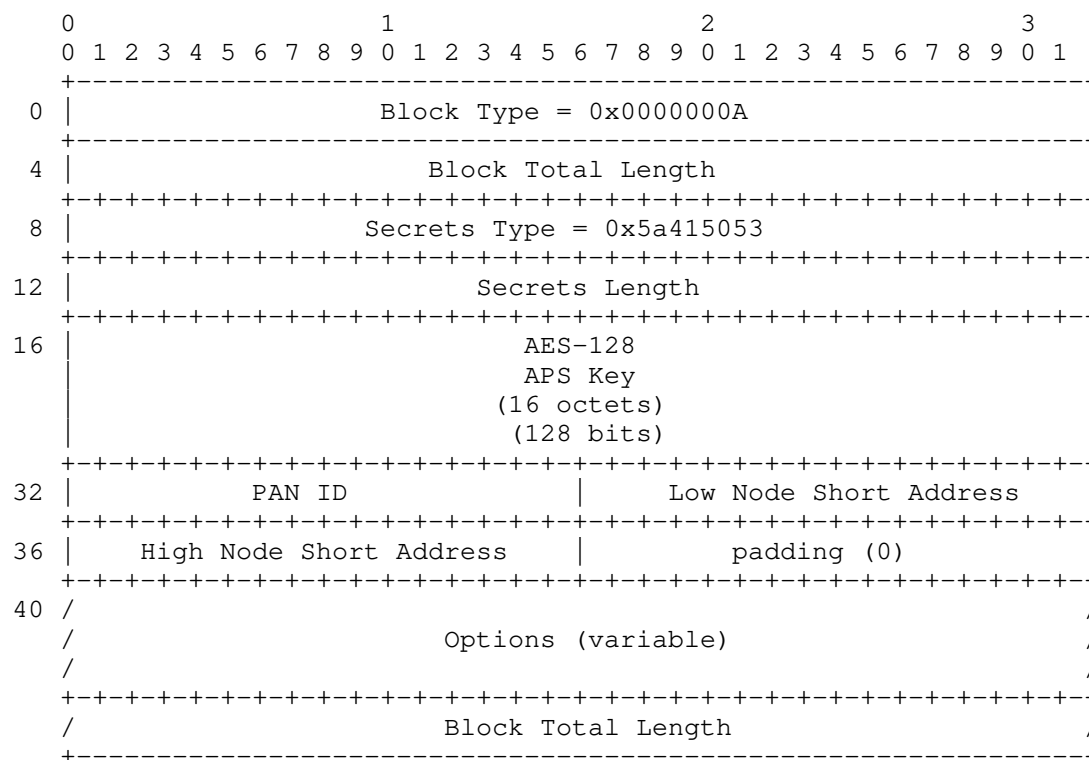


Figure 17: ZigBee APS Key Data Format

#### 4.8. Custom Block

A Custom Block (CB) is the container for storing custom data that is not part of another block; for storing custom data as part of another block, see Section 3.5.1. The Custom Block is optional, can be repeated any number of times, and can appear before or after any other block except the first Section Header Block which must come first in the file. Different Custom Blocks, of different type codes and/or different Private Enterprise Numbers, may be used in the same pcapng file. The format of a Custom Block is shown in Figure 18.

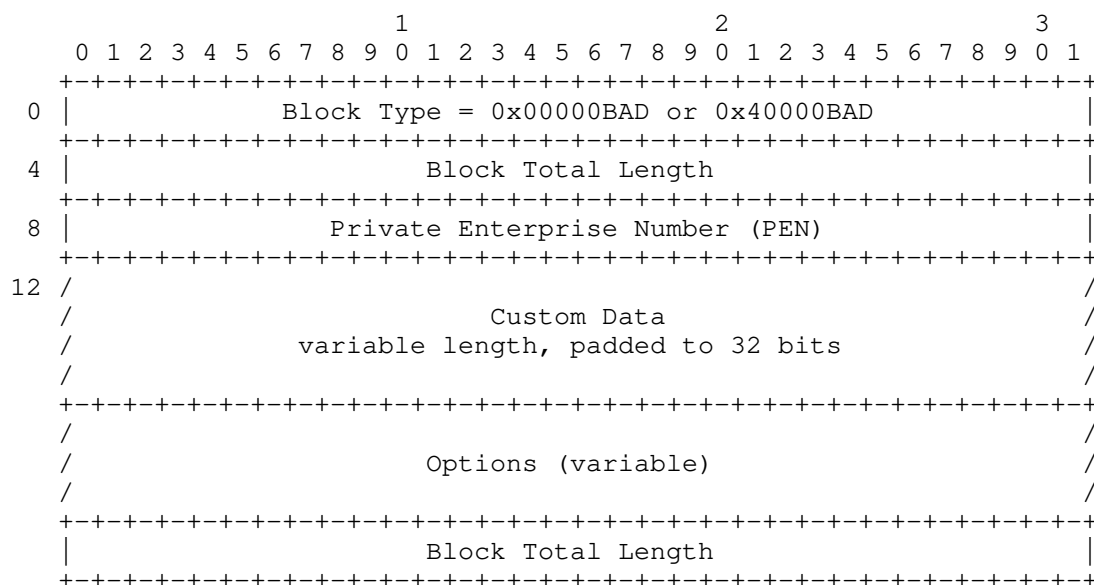


Figure 18: Custom Block Format

The Custom Block uses the type code 0x00000BAD (2989 in decimal) for a custom block that pcapng re-writers can copy into new files, and the type code 0x40000BAD (1073744813 in decimal) for one that should not be copied. See Section 5.2 for details.

The Custom Block has the following fields:

- \* Block Type: The block type of the Custom Block is 0x00000BAD or 0x40000BAD, as described previously.
- \* Block Total Length: total size of this block, as described in Section 3.1.
- \* Private Enterprise Number (32 bits): An IANA-assigned Private Enterprise Number identifying the organization which defined the Custom Block. See Section 5.1 for details. The PEN MUST be encoded using the same endianness as the Section Header Block it is within the scope of.
- \* Custom Data: the custom data, padded to a 32 bit boundary.
- \* Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present. Note that custom options for the Custom Block still use the custom option format and type code, as described in Section 3.5.1.

## 5. Vendor-Specific Custom Extensions

This section uses the term "vendor" to describe an organization which extends the pcapng file with custom, proprietary blocks or options. It should be noted, however, that the "vendor" is just an abstract entity that agrees on a custom extension format: for example it may be a manufacturer, industry association, an individual user, or collective group of users.

### 5.1. Supported Use-Cases

There are two different supported use-cases for vendor-specific custom extensions: local and portable. Local use means the custom data is only expected to be usable on the same machine, and the same application, which encoded it into the file. This limitation is due to the lack of a common registry for the local use number codes (the block or option type code numbers with the Most Significant Bit set). Since two different vendors may choose the same number, one vendor's application reading the other vendor's file would result in decoding failure. Therefore, vendors SHOULD instead use the portable method, as described next.

The portable use-case supports vendor-specific custom extensions in pcapng files which can be shared across systems, organizations, etc. To avoid number space collisions, an IANA-registered Private Enterprise Number (PEN) is encoded into the Custom Block or Custom Option, using the PEN that belongs to the vendor defining the extension. Anyone can register a new PEN with IANA, for free, by filling out the online request form at [http://pen.iana.org/pen/](http://pen.iana.org/pen/PenApplication.page) PenApplication.page (<http://pen.iana.org/pen/PenApplication.page>).

## 5.2. Controlling Copy Behavior

Both Custom Blocks and Custom Options support two different codes to distinguish their "copy" behavior: a code for when the block or option can be safely copied into a new pcapng file by a pcapng manipulating application, and a code for when it should not be copied. A common reason for not copying a Custom Block or Custom Option is because it depends on other blocks or options in some way that would invalidate the custom data if the other blocks/options were removed or re-ordered. For example, if a Custom Block's data includes an Interface ID number in its Custom Data portion, then it cannot be safely copied by a pcapng application that merges pcapng files, because the merging application might re-order or remove one or more of the Interface Description Blocks, and thereby change the Interface IDs that the Custom Block depends upon. The same issue arises if a Custom Block or Custom Option depends on the presence of, or specific ordering of, other standard-based or custom-defined blocks or options.

Note that the copy semantics is not related to privacy - there is no guarantee that a pcapng anonymizer will remove a Custom Block or Custom Option, even if the appropriate code is used requesting it not be copied; and the original pcapng file can be shared anyway. If the Custom Data portion of the Custom Block or Custom Option contains sensitive information, then it should be encrypted in some fashion.

## 5.3. Strings vs. Octets

For the Custom Options, there are two Custom Data formats supported: a UTF-8 string and a binary data payload. The rationale for this separation is that a pcapng display application which does not understand the specific PEN's Custom Option can still display the data as a string if it's a string type code, rather than as hex-ascii of the octets.

## 5.4. Endianness Issues

Implementers writing Custom Blocks or binary data Custom Options should be aware that a pcapng file can be re-written by machines using a different endianness than the original file, which means all known fields of the pcapng file will change endianness in the new file. Since the Custom Data payload of the Custom Block or the binary data Custom Option might be an arbitrary sequence of unknown octets to such machines, they cannot convert multi-octet values inside the Custom Data, or in the Options section of a Custom Block, into the appropriate endianness.

For example, a little-endian machine can create a new pcapng file and add some binary data Custom Options to some non-Custom Block(s) in the file. This file can then be sent to a big-endian host, which will convert the Option Code, Option Length, and PEN fields of the options to big-endian format if it re-writes the file. However, if the software reading the file does not understand the contents of all of the Custom Options, it will leave the Custom Data payload of the options alone (as little-endian format). If this file then gets sent to a little-endian machine, then, when that little-endian machine reads the file, it will, if the software reading the file understands the contents of all the Custom Options, it will detect that the file format is big-endian, and swap the endianness while it parses the file - but that will cause the Custom Data payload to be incorrect since it was already in little-endian format.

In addition, a little-endian machine can create a pcapng file and write some binary data Custom Blocks, containing options, to the file. The file can then be sent to a big-endian host, which, if the software reading the file does not understand the contents of the Custom Blocks, will leave the Custom Data and Options alone (as little-endian format). If this file then gets sent to a little-endian machine, then, when that little-endian machine reads the file, it will, if the software reading the file understands the contents of all the Custom Blocks, it will detect that the file format is big-endian, and swap the endianness while it parses the file - but that will cause the Custom Data payload, the Option Code and Option Length values in the Options, and the PEN in any Custom Options to be incorrect since they were already in little-endian format.

Therefore, the vendor should either encode the Custom Data of their Custom Blocks and Custom Options, the Option Code and Option Length fields of options in Custom Blocks, and the PEN field of Custom Options in Custom Blocks in a consistent manner, such as always in big-endian or always in little-endian format, regardless of the host platform's endianness, or should encode some flag in the Custom Data payload to indicate in which endianness the rest of the payload is written.

The PEN field of a Custom Block, or of a Custom Option not contained in a Custom Block, MUST be converted by code that reads pcapng files, so this is not an issue for that field, except for Custom Options in Custom Blocks. This is also not an issue for the Custom Data payload of UTF-8 string Custom Options.

#### 6. Recommended File Name Extension: .pcapng

The recommended file name extension for the "PCAP Next Generation Capture File Format" specified in this document is ".pcapng".

On Windows and macOS, files are distinguished by an extension to their filename. Such an extension is technically not actually required, as applications should be able to automatically detect the pcapng file format through the "magic bytes" at the beginning of the file, as some other UN\*X desktop environments do. However, using name extensions makes it easier to work with files (e.g. visually distinguish file formats) so it is recommended - though not required - to use .pcapng as the name extension for files following this specification.

Please note: To avoid confusion (such as the current usage of .cap for a plethora of different capture file formats) file name extensions other than .pcapng should be avoided.

## 7. Conclusions

The file format proposed in this document should be very versatile and satisfy a wide range of applications. In the simplest case, it can contain a raw capture of the network data, made of a series of Simple Packet Blocks. In the most complex case, it can be used as a repository for heterogeneous information. In every case, the file remains easy to parse and an application can always skip the data it is not interested in; at the same time, different applications can share the file, and each of them can benefit of the information produced by the others. Two or more files can be concatenated obtaining another valid file.

## 8. Implementations

Some known implementations that read or write the pcapng file format are listed on the pcapng GitHub wiki (<https://github.com/pcapng/pcapng/wiki/Implementations>).

## 9. Security Considerations

TBD.

## 10. IANA Considerations

TBD.

[Open issue: decide whether the block types, option types, NRB Record types, etc. should be IANA registries. And if so, what the IANA policy for each should be (see RFC 5226)]



### 10.1. Standardized Block Type Codes

Every Block is uniquely identified by a 32-bit integer value, stored in the Block Header.

As pointed out in Section 3.1, Block Type codes whose Most Significant Bit (bit 31) is set to 1 are reserved for local use by the application.

All the remaining Block Type codes (0x00000000 to 0x7FFFFFFF) are standardized by this document. Requests for new Block Type codes, Option Type codes, and Secrets Type codes should be made by creating a pull request to update this document at [github.com/pcapng/pcapng](https://github.com/pcapng/pcapng) (<https://github.com/pcapng/pcapng>). The pull request should add a description of the new block, option, or secret type to Section 4. The pull request description should contain a clear request for a new type code assignment.

The following is a list of the Standardized Block Type Codes:

Block Type Code	Description
0x00000000	Reserved ???
0x00000001	Interface Description Block (Section 4.2)
0x00000002	Packet Block (Appendix A)
0x00000003	Simple Packet Block (Section 4.4)
0x00000004	Name Resolution Block (Section 4.5)
0x00000005	Interface Statistics Block (Section 4.6)
0x00000006	Enhanced Packet Block (Section 4.3)
0x00000007	IRIG Timestamp Block (requested by Gianluca Varenni < <a href="mailto:gianluca.varenni@cacetech.com">gianluca.varenni@cacetech.com</a> >, CACE Technologies LLC); code also used for Socket Aggregation Event Block ( <a href="https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt">https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt</a> )
0x00000008	ARINC 429 ( <a href="https://en.wikipedia.org/wiki/ARINC_429">https://en.wikipedia.org/wiki/ARINC_429</a> ) in AFDX Encapsulation Information Block (requested by Gianluca Varenni < <a href="mailto:gianluca.varenni@cacetech.com">gianluca.varenni@cacetech.com</a> >, CACE

	Technologies LLC)
0x00000009	[systemd Journal Export Block] [I-D.richardson-opsawg-pcapng-extras]
0x0000000A	Decryption Secrets Block (Section 4.7)
0x00000101	Hone Project ( <a href="https://github.com/HoneProject">https://github.com/HoneProject</a> ) Machine Info Block ( <a href="https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format">https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format</a> ) (see also Google version ( <a href="https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt">https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt</a> ))
0x00000102	Hone Project ( <a href="https://github.com/HoneProject">https://github.com/HoneProject</a> ) Connection Event Block ( <a href="https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format">https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format</a> ) (see also Google version ( <a href="https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt">https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt</a> ))
0x00000201	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Machine Info Block
0x00000202	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 1
0x00000203	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) FD List Block
0x00000204	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Event Block
0x00000205	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Interface List Block
0x00000206	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) User List Block
0x00000207	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 2
0x00000208	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Event Block with flags
0x00000209	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> )

	Process Info Block, version 3
0x00000210	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 4
0x00000211	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 5
0x00000212	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 6
0x00000213	Sysdig ( <a href="https://github.com/draios/sysdig">https://github.com/draios/sysdig</a> ) Process Info Block, version 7
0x00000BAD	Custom Block that rewriters can copy into new files (Section 4.8)
0x40000BAD	Custom Block that rewriters should not copy into new files (Section 4.8)
0x0A0D0D0A	Section Header Block (Section 4.1)
0x0A0D0A00-0x0A0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0A-0xFF0A0D0A	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0D-0xFF0A0D0D	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x0D0D0A00-0x0D0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the FTP protocol in text mode.
0x80000000-0xFFFFFFFF	Reserved for local use.

Table 9: Standardized Block Type Codes

[Open issue: reserve 0x40000000-0x7FFFFFFF for do-not-copy-bit range of base types?]

## 11. Contributors

Loris Degioanni and Gianluca Varenni were coauthoring this document before it was submitted to the IETF.

## 12. Acknowledgments

The authors wish to thank Anders Broman, Ulf Lamping, Richard Sharpe and many others for their invaluable comments.

## 13. References

### 13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 13.2. Informative References

- [I-D.richardson-opsawg-pcapng-extras]  
"\*\*\* BROKEN REFERENCE \*\*\*".
- [LINKTYPES]  
The Tcpdump Group, "the tcpdump.org link-layer header types registry", <<http://www.tcpdump.org/linktypes.html>>.

## Appendix A. Packet Block (obsolete!)

The Packet Block is obsolete, and MUST NOT be used in new files. Use the Enhanced Packet Block or Simple Packet Block instead. This section is for historical reference only.

A Packet Block was a container for storing packets coming from the network.

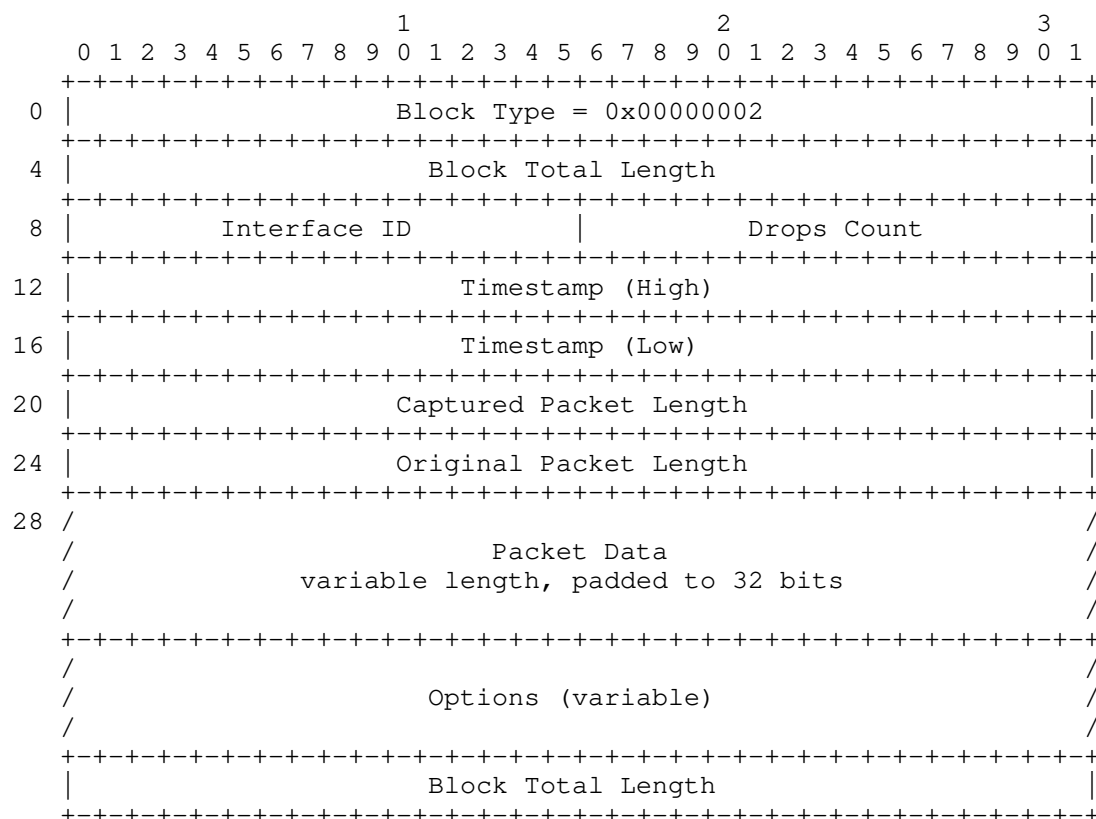


Figure 19: Packet Block Format

The Packet Block has the following fields:

- \* Block Type: The block type of the Packet Block is 2.
- \* Block Total Length: total size of this block, as described in Section 3.1.
- \* Interface ID: specifies the interface this packet comes from; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by the same number (see Section 4.2) of this field. The interface ID MUST be valid, which means that an matching interface description block MUST exist.

- \* **Drops Count:** a local drop counter. It specifies the number of packets lost (by the interface and the operating system) between this packet and the preceding one. The value xFFFF (in hexadecimal) is reserved for those systems in which this information is not available.
- \* **Timestamp (High) and Timestamp (Low):** timestamp of the packet. The format of the timestamp is the same as was already defined for the Enhanced Packet Block (Section 4.3).
- \* **Captured Packet Length:** number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 10). The value of this field does not include the padding octets added at the end of the Packet Data field to align the Packet Data field to a 32-bit boundary.
- \* **Original Packet Length:** actual length of the packet when it was transmitted on the network. It can be different from Captured Packet Length if the packet has been truncated by the capture process.
- \* **Packet Data:** the data coming from the network, including link-layer headers. The actual length of this field is Captured Packet Length plus the padding to a 32-bit boundary. The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].
- \* **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options were valid within this block:

Name	Code	Length	Multiple allowed?
pack_flags	2	4	no
pack_hash	3	variable	yes

Table 10: Packet Block Options

pack\_flags:

The `pack_flags` option is the same as the `epb_flags` of the enhanced packet block.

Example: `'0'`.

`pack_hash`:

The `pack_hash` option is the same as the `epb_hash` of the enhanced packet block.

Examples: `'02 EC 1D 87 97'`, `'03 45 6E C2 17 7C 10 1E 3C 2E 99 6E C2 9A 3D 50 8E'`.

#### Authors' Addresses

Michael Tuexen (editor)  
Muenster University of Applied Sciences  
Stegerwaldstrasse 39  
48565 Steinfurt  
Germany

Email: [tuexen@fh-muenster.de](mailto:tuexen@fh-muenster.de)

Fulvio Risso  
Politecnico di Torino  
Corso Duca degli Abruzzi, 24  
10129 Torino  
Italy

Email: [fulvio.risso@polito.it](mailto:fulvio.risso@polito.it)

Jasper Bongertz  
Airbus Defence and Space CyberSecurity  
Kanzlei 63c  
40667 Meerbusch  
Germany

Email: [jasper@packet-foo.com](mailto:jasper@packet-foo.com)

Gerald Combs  
Wireshark Foundation  
339 Madson Pl  
Davis, CA 95618  
United States of America

Email: [gerald@wireshark.org](mailto:gerald@wireshark.org)

Guy Harris

Email: [gharris@sonic.net](mailto:gharris@sonic.net)

Eelco Chaudron  
Red Hat  
De Entree 238  
1101 EE Amsterdam  
Netherlands

Email: [eelco@redhat.com](mailto:eelco@redhat.com)

Michael C. Richardson  
Sandelman Software Works

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)  
URI: <http://www.sandelman.ca/>