

Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 25, 2018

A. Mishra
O3b Networks
M. Jethanandani

A. Saxena
Ciena Corporation
November 21, 2017

Reverse Defect Indicator for MPLS FM OAM
draft-ama-mpls-fm-rdi-00.txt

Abstract

This document describes extensions to the MPLS Fault Management Operations, Administration, and Management (MPLS FM OAM) in RFC 6427 [RFC6427] to support Remote Defect Indication (RDI) functionality. Specifically, it describes a mechanism for propagating MPLS FM OAM messages to the upstream Label Edge Router (LER) in MPLS-TP [RFC5921] bi-directional (associated and co-routed) Label Switched Paths (LSPs).

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 May 25, 2018.

Copyright Notice

Copyright (c) 2017 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. Reverse Defect Indicator (RDI)	2
3. Theory of Operations	3
3.1. RDI Operation in Associated Bidirectional LSPs	3
3.2. RDI Operation in Co-routed Bidirectional LSPs	4
4. IANA Requirements	4
5. Security Consideration	4
6. Acknowledgements	4
7. References	4
7.1. Normative References	4
7.2. Informative References	4
Authors' Addresses	5

1. Introduction

The MPLS Fault Management Operations, Administration, and Management (MPLS FM OAM) in RFC 6427 [RFC6427] describes a method to identify faults in MPLS transport networks, and a protocol to notify the upstream Label Edge Router (LER). However, in the case of MPLS-TP [RFC5921] bi-directional Label Switched Paths (LSPs), the fault must be coordinated on both, the upstream LER and the downstream LER (which is the upstream LER for the reverse path).

In typical scenario, the Bidirectional Forwarding Detection (BFD) protocol, as described in RFC 5880 [RFC5880], detects the fault signaled by MPLS FM OAM on the upstream LER and propagates the fault on the reverse LSP to the other MPLS-TP LSP LER. This allows the two MPLS-TP LERs to coordinate failover to backup LSPs.

This document proposes a mechanism to achieve MPLS FM OAM fault propagation on the MPLS-TP reverse LSP using a Reverse Defect Indicator (RDI) MPLS FM OAM message. This allows fast fault coordination between the bidirectional LSP end-points when the use of BFD is not feasible.

2. Reverse Defect Indicator (RDI)

The functionality proposed for MPLS FM OAM RDI is achieved by adding a RDI-flag in the MPLS Fault OAM message .

```

+---+---+---+---+---+---+
|Reserved |RDI|L|R|
+---+---+---+---+---+---+

```

Figure 1: RDI-Flag in MPLS FM OAM Flags

where:

L-Flag and R-Flag are as defined in RFC 6427 [RFC6427].

RDI-Flag: Reverse Defect Indication Flag. The RDI-Flag is clear in the common MPLS FM OAM messages as defined in RFC 6427 [RFC6427]. The RDI-Flag is set to indicate that the message is MPLS FM OAM RDI.

3. Theory of Operations

3.1. RDI Operation in Associated Bidirectional LSPs

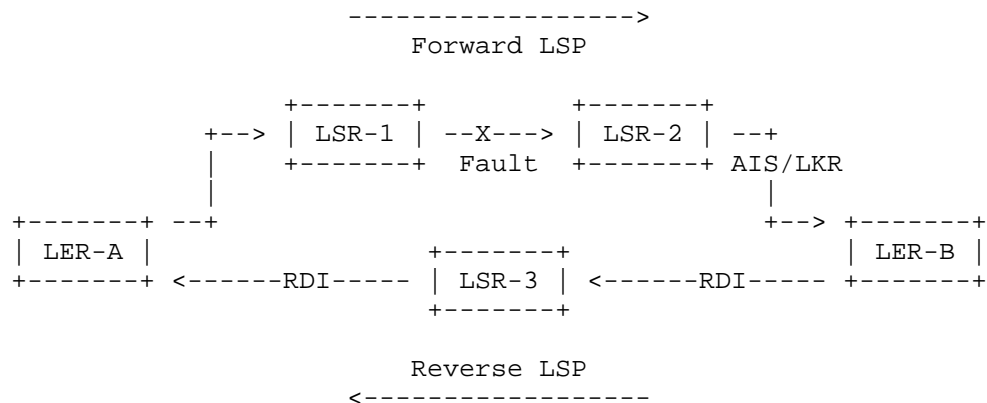


Figure 2: RDI Operation in Associated Bidirectional LSP

Figure 1 depicts an associated bidirectional LSP with:

Forward LSP (LER-A, LSR-1, LSR-2, LER-B)

Reverse LSP (LER-B, LSR-3, LER-A)

Scenario 1, Fault on LER-A: LSR-1 will detect a fault on the server sub-layer and generate AIS/LKR message on the upstream link for Forward LSP (towards LSR-2). LSR-2 will process the message and forward it, unaltered, upstream to LER-B. LER-B will process the message, set the RDI-Flag and forward it on the associated Reverse LSP. Because the RDI-Flag is set, LSR-3 does not need to process the message as the fault is not on the Reverse LSP and forwards it,

unaltered, towards LER-A. LER-A, if it receives the message (the fault may only be on the forward LSP on LER-A) processes the message and discard it (RDI-Flag set received on Reverse LSP indicates the fault is on the Forward LSP, and vice-versa). When the fault clears, LSR-1 will issue new set of AIS/LKR messages to clear the previous fault condition. This message is also propagated using the previous RDI logic to coordinate fault clear on the Reverse LSP.

Scenario 2, Link fault or LSR fault on Forward LSP: Same logic as fault on LER-A.

3.2. RDI Operation in Co-routed Bidirectional LSPs

RDI is not a required mechanism in co-routed bidirectional LSPs as MPLS LSx on either direction of the fault will generate MPLS FM OAM messages and the fault is propagated to both LERs.

4. IANA Requirements

None.

5. Security Consideration

No additional security impact because of addition of RDI-Flag in MPLS FM OAM messages.

6. Acknowledgements

7. References

7.1. Normative References

- [RFC5921] Bocci, M., Ed., Bryant, S., Ed., Frost, D., Ed., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, DOI 10.17487/RFC5921, July 2010, <<https://www.rfc-editor.org/info/rfc5921>>.
- [RFC6427] Swallow, G., Ed., Fulignoli, A., Ed., Vigoureux, M., Ed., Boutros, S., and D. Ward, "MPLS Fault Management Operations, Administration, and Maintenance (OAM)", RFC 6427, DOI 10.17487/RFC6427, November 2011, <<https://www.rfc-editor.org/info/rfc6427>>.

7.2. Informative References

- [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>>.

Authors' Addresses

Ashesh Mishra
O3b Networks
USA

Email: mishra.ashesh@outlook.com

Mahesh Jethanandani
USA

Email: mjethanandani@gmail.com

Ankur Saxena
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: ankurpsaxena@gmail.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 23, 2018

A. Farrel
Juniper Networks
S. Bryant
Huawei
J. Drake
Juniper Networks
March 22, 2018

An MPLS-Based Forwarding Plane for Service Function Chaining
draft-farrel-mpls-sfc-05

Abstract

Service Function Chaining (SFC) is the process of directing packets through a network so that they can be acted on by an ordered set of abstract service functions before being delivered to the intended destination. An architecture for SFC is defined in RFC7665.

The Network Service Header (NSH) can be inserted into packets to steer them along a specific path to realize a Service Function Chain.

Multiprotocol Label Switching (MPLS) is a widely deployed forwarding technology that uses labels placed in a packet in a label stack to identify the forwarding actions to be taken at each hop through a network. Actions may include swapping or popping the labels as well, as using the labels to determine the next hop for forwarding the packet. Labels may also be used to establish the context under which the packet is forwarded.

This document describes how Service Function Chaining can be achieved in an MPLS network by means of a logical representation of the NSH in an MPLS label stack. It does not deprecate or replace the NSH, but acknowledges that there may be a need for an interim deployment of SFC functionality in brownfield networks.

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 September 23, 2018.

Copyright Notice

Copyright (c) 2018 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. Requirements Language	4
3. Choice of Data Plane SPI/SI Representation	4
4. Basic Unit of Representation	4
5. MPLS Label Swapping	6
6. MPLS Label Stacking	8
7. Mixed Mode Forwarding	10
8. A Note on Service Function Capabilities and SFC Proxies	11
9. Control Plane Considerations	11
10. Use of the Entropy Label	12
11. Metadata	12
11.1. Indicating Metadata in User Data Packets	13
11.2. Inband Programming of Metadata	15
12. Worked Examples	18
13. Security Considerations	22
14. IANA Considerations	22
15. Acknowledgements	23
16. References	23
16.1. Normative References	23
16.2. Informative References	24
Authors' Addresses	24

1. Introduction

Service Function Chaining (SFC) is the process of directing packets through a network so that they can be acted on by an ordered set of abstract service functions before being delivered to the intended destination. An architecture for SFC is defined in [RFC7665].

When applying a particular Service Function Chain to the traffic selected by a service classifier, the traffic needs to be steered through an ordered set of Service Functions (SFs) in the network. This ordered set of SFs is termed a Service Function Path (SFP), and the traffic is passed between Service Function Forwarders (SFFs) that are responsible for delivering the packets to the SFs and for forwarding them onward to the next SFF.

In order to steer the selected traffic between SFFs and to the correct SFs the service classifier needs to attach information to each packet. This information indicates the SFP on which the packet is being forwarded and hence the SFs to which it must be delivered. The information also indicates the progress the packet has already made along the SFP.

The Network Service Header (NSH) [RFC8300] has been defined to carry the necessary information for Service Function Chaining in packets. The NSH can be inserted into packets and contains various information including a Service Path Indicator (SPI), a Service Index (SI), and a Time To Live (TTL) counter.

Multiprotocol Label Switching (MPLS) [RFC3031] is a widely deployed forwarding technology that uses labels placed in a packet in a label stack to identify the forwarding actions to be taken at each hop through a network. Actions may include swapping or popping the labels as well, as using the labels to determine the next hop for forwarding the packet. Labels may also be used to establish the context under which the packet is forwarded. In many cases, MPLS will be used as a tunneling technology to carry packets through networks between SFFs.

This document describes how Service Function Chaining can be achieved in an MPLS network by means of a logical representation of the NSH in an MPLS label stack. This approach is applicable to all forms of MPLS forwarding (where labels are looked up at each hop, and swapped or popped [RFC3031]). It does not deprecate or replace the NSH, but acknowledges that there may be a need for an interim deployment of SFC functionality in brownfield networks. The mechanisms described in this document are a compromise between the full function that can be achieved using the NSH, and the benefits of reusing the existing MPLS forwarding paradigms.

It is assumed that the reader is fully familiar with the terms and concepts introduced in [RFC7665] and [RFC8300].

Note that one of the features of the SFC architecture described in [RFC7665] is the "SFC proxy" that exists to include legacy SFs that are not able to process NSH-encapsulated packets. This issue is equally applicable to the use of MPLS-encapsulated packets that encode a logical representation of an NSH. It is discussed further in Section 8.

2. Requirements Language

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. Choice of Data Plane SPI/SI Representation

While [RFC8300] defines the NSH that can be used in a number of environments, this document provides a mechanism to handle situations in which the NSH is not ubiquitously deployed. In this case it is possible to use an alternative data plane representation of the SPI/SI by carrying the identical semantics in MPLS labels.

In order to correctly select the mechanism by which SFC information is encoded and carried between SFFs, it may be necessary to configure the capabilities and choices either within the whole Service Function Overlay Network, or on a hop by hop basis. It is a requirement that both ends of a tunnel over the underlay network (i.e., a pair of SFFs adjacent in the SFC) know that the tunnel is used for SFC and know what form of NSH representation is used. A control plane signalling approach to achieve these objectives is provided using BGP in [I-D.ietf-bess-nsh-bgp-control-plane].

Note that the encoding of the SFC information is independent of the choice of tunneling technology used between SFFs. Thus, an MPLS representation of the logical NSH (as defined in this document) may be used even if the tunnel between a pair of SFFs is not an MPLS tunnel. Conversely, MPLS tunnels may be used to carry other encodings of the logical NSH (specifically, the NSH itself).

4. Basic Unit of Representation

When an MPLS label stack is used to carry a logical NSH, a basic unit of representation is used. This unit comprises two MPLS labels as

shown below. The unit may be present one or more times in the label stack as explained in subsequent sections.

In order to convey the same information as is present in the NSH, two MPLS label stack entries are used. One carries a label to provide context within the SFC scope (the SFC Context Label), and the other carries a label to show which service function is to be actioned (the SF Label). This two-label unit is shown in Figure 1.

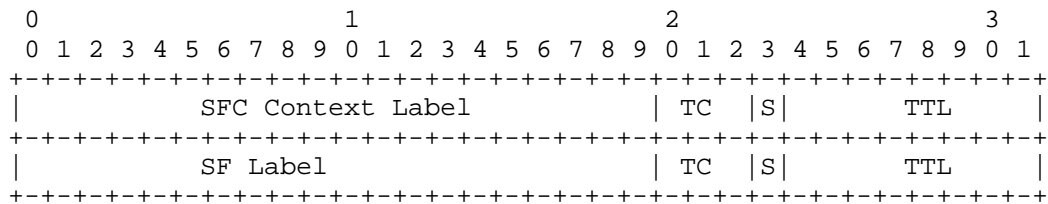


Figure 1: The Basic Unit of MPLS Label Stack for SFC

The fields of these two label stack entries are encoded as follows:

Label: The Label fields contain the values of the SFC Context Label and the SF Label encoded as 20 bit integers. The precise semantics of these label fields are dependent on whether the label stack entries are used for MPLS label swapping (see Section 5) or MPLS label stacking (see Section 6).

TC: The TC bits have no meaning. They SHOULD be set to zero in both label stack entries when a packet is sent and MUST be ignored on receipt.

S: The bottom of stack bit has its usual meaning in MPLS. It MUST be clear in the SFC Context label stack entry and MAY be set in the SF label stack entry depending on whether the label is the bottom of stack.

TTL: The TTL field in the SFC Context label stack entry SHOULD be set to 1. The TTL in SF label stack entry (called the SF TTL) is set according to its use for MPLS label swapping (see Section 5) or MPLS label stacking (see Section 6 and is used to mitigate packet loops.

The sections that follow show how this basic unit of MPLS label stack may be used for SFC in the MPLS label swapping case and in the MPLS label stacking. For simplicity, these sections do not describe the use of metadata: that is covered separately in Section 11.

5. MPLS Label Swapping

This section describes how the basic unit of MPLS label stack for SFC introduced in Section 4 is used when MPLS label swapping is in use. As can be seen from Figure 2, the top of the label stack comprises the labels necessary to deliver the packet over the MPLS tunnel between SFFs. Any MPLS encapsulation may be used (i.e., MPLS, MPLS in UDP, MPLS in GRE, and MPLS in VXLAN or GPE), thus the tunnel technology does not need to be MPLS, but that is shown here for simplicity.

An entropy label ([RFC6790]) may also be present as described in Section 10

Under these labels (or other encapsulation) comes a single instance of the basic unit of MPLS label stack for SFC. In addition to the interpretation of the fields of these label stack entries provided in Section 4 the following meanings are applied:

SPI Label: The Label field of the SFC Context label stack entry contains the value of the SPI encoded as a 20 bit integer. The semantics of the SPI is exactly as defined in [RFC8300]. Note that an SPI as defined by [RFC8300] can be encoded in 3 octets (i.e., 24 bits), but that the Label field allows for only 20 bits and reserves the values 0 through 15 as 'special purpose' labels [RFC7274]. Thus, a system using MPLS representation of the logical NSH MUST NOT assign SPI values greater than $2^{20} - 1$ or less than 16.

SI Label: The Label field of the SF label stack entry contains the value of the SI exactly as defined in [RFC8300]. Since the SI requires only 8 bits, and to avoid overlap with the 'special purpose' label range of 0 through 15 [RFC7274], the SI is carried in the top (most significant) 8 bits of the Label field with the low order 12 bits set to zero.

TC: The TC fields are as described in Section 4.

S: The S bits are as described in Section 4.

TTL: The TTL field in the SPI label stack entry SHOULD be set to 1 as stated in Section 4. The TTL in SF label stack entry is decremented once for each forwarding hop in the SFP, i.e., for each SFF transited, and so mirrors the TTL field in the NSH.

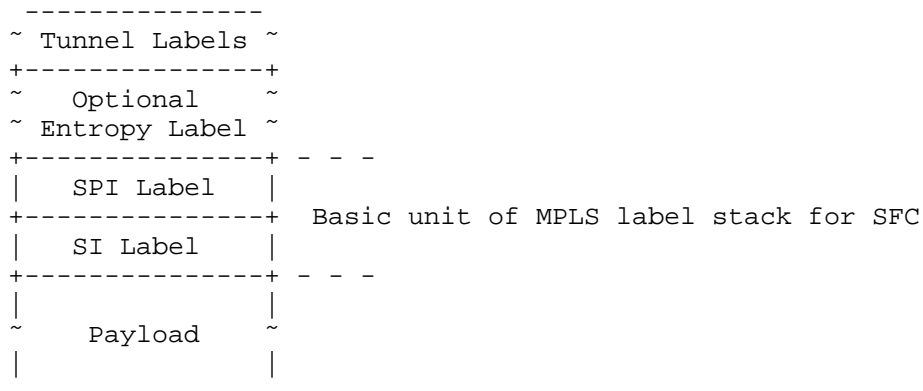


Figure 2: The MPLS SFC Label Stack

The following processing rules apply to the Label fields:

- o When a Classifier inserts a packet onto an SFP it sets the SPI Label to indicate the identity of the SFP, and sets the SI Label to indicate the first SF in the path.
- o When a component of the SFC system processes a packet it uses the SPI Label to identify the SFP and the SI Label to determine to which SFF or instance of an SF (an SFI) to deliver the packet. Under normal circumstances (with the exception of branching and reclassification - see [I-D.ietf-bess-nsh-bgp-control-plane]) the SPI Label value is preserved on all packets. The SI Label value is modified by SFFs and through reclassification to indicate the next hop along the SFP.

The following processing rules apply to the TTL field of the SF label stack entry, and are derived from section 2.2 of [RFC8300]:

- o When a Classifier places a packet onto an SFP it MUST set the TTL to a value between 1 and 255. It SHOULD set this according to the expected length of the SFP (i.e., the number of SFs on the SFP), but it MAY set it to a larger value according to local configuration. The maximum TTL value supported in an NSH is 63, and so the practical limit here may also be 63.
- o When an SFF receives a packet from any component of the SFC system (Classifier, SFI, or another SFF) it MUST discard any packets with TTL set to zero. It SHOULD log such occurrences, but MUST apply rate limiting to any such logs.

- o An SFF MUST decrement the TTL by one each time it performs a forwarding lookup.
- o If an SFF decrements the TTL to zero it MUST NOT send the packet, and MUST discard the packet. It SHOULD log such occurrences, but MUST apply rate limiting to any such logs.
- o SFIs MUST ignore the TTL, but MUST mirror it back to the SFF unmodified along with the SI (which may have been changed by local reclassification).
- o If a Classifier along the SFP makes any change to the intended path of the packet including for looping, jumping, or branching (see [I-D.ietf-bess-nsh-bgp-control-plane] it MUST NOT change the SI TTL of the packet. In particular, each component of the SFC system MUST NOT increase the SI TTL value otherwise loops may go undetected.

6. MPLS Label Stacking

This section describes how the basic unit of MPLS label stack for SFC introduced in Section 4 is used when MPLS label stacking is used to carry information about the SFP and SFs to be executed. As can be seen in Figure 3, the top of the label stack comprises the labels necessary to deliver the packet over the MPLS tunnel between SFFs. Any MPLS encapsulation may be used.

An entropy label ([RFC6790]) may also be present as described in Section 10

Under these labels comes one of more instances of the basic unit of MPLS label stack for SFC. In addition to the interpretation of the fields of these label stack entries provided in Section 4 the following meanings are applied:

SFC Context Label: The Label field of the SFC Context label stack entry contains a label that delivers SFC context. This label may be used to indicate the SPI encoded as a 20 bit integer using the semantics of the SPI is exactly as defined in [RFC8300] and noting that in this case a system using MPLS representation of the logical NSH MUST NOT assign SPI values greater than $2^{20} - 1$ or less than 16. This label may also be used to convey other SFC context-specific semantics such as indicating how to interpret the SF Label or how to forward the packet to the node that offers the SF.

SF Label: The Label field of the SF label stack entry contains a value that identifies the next SFI to be actioned for the packet.

This label may be scoped globally or within the context of the preceding SFC Context Label and comes from the range $16 \dots 2^{20} - 1$.

TC: The TC fields are as described in Section 4.

S: The S bits are as described in Section 4.

TTL: The TTL fields in the SFC Context label stack entry SF label stack entry SHOULD be set to 1 as stated in Section 4, but MAY be set to larger values if the label indicated a forwarding operation towards the node that hosts the SF.

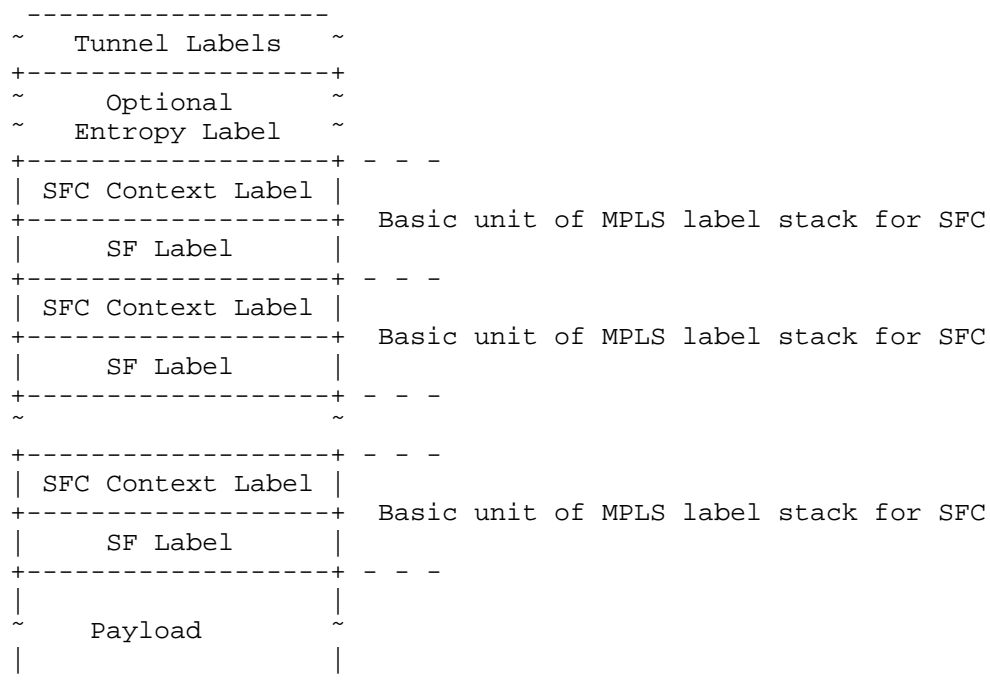


Figure 3: The MPLS SFC Label Stack for Label Stacking

The following processing rules apply to the Label fields:

- o When a Classifier inserts a packet onto an SFP it adds a stack comprising one or more instances of the basic unit of MPLS label stack for SFC. Taken together, this stack defines the SFs to be actioned and so defines the SFP that the packet will traverse.

- o When a component of the SFC system processes a packet it uses the top basic unit of label stack for SFC to determine to which SFI to next deliver the packet. When an SFF receives a packet it examines the top basic unit of MPLS label stack for SFC to determine where to send the packet next. If the next recipient is a local SFI, the SFC strips the basic unit of MPLS label stack for SFC before forwarding the packet.

7. Mixed Mode Forwarding

The previous sections describe homogeneous networks where SFC forwarding is either all label swapping or all label popping (stacking). But it is also possible that different parts of the network utilize swapping or popping. It is also worth noting that a Classifier may be content to use an SFP as installed in the network by a control plane or management plane and so would use label swapping, but that there may be a point in the SFP where a choice of SFIs can be made (perhaps for load balancing) and where, in this instance, the Classifier wishes to exert control over that choice by use of a specific entry on the label stack.

When an SFF receives a packet containing an MPLS label stack, it checks whether it is processing an {SFP, SI} label pair for label swapping or a {context label, SFI index} label pair for label stacking. It then selects the appropriate SFI to which to send the packet. When it receives the packet back from the SFI, it has four cases to consider.

- o If the current hop requires an {SFP, SI} and the next hop requires an {SFP, SI}, it sets the SI label to the SI value of the current hop, selects an instance of the SF to be executed at the next hop, and tunnels the packet to the SFF for that SFI.
- o If the current hop requires an {SFP, SI} and the next hop requires a {context label, SFI label}, it pops the {SFP, SI} from the top of the MPLS label stack and tunnels the packet to the SFF indicated by the context label.
- o If the current hop requires a {context label, SFI label}, it pops the {context label, SFI label} from the top of the MPLS label stack.
 - * If the new top of the MPLS label stack contains an {SFP, SI} label pair, it selects an SFI to use at the next hop, and tunnels the packet to SFF for that SFI.

- * If the top of the MPLS label stack contains a {context label, SFI label}, it tunnels the packet to the SFF indicated by the context label.

8. A Note on Service Function Capabilities and SFC Proxies

The concept of an "SFC Proxy" is introduced in [RFC7665]. An SFC Proxy is logically located between an SFF and an SFI that is not "SFC-aware". Such SFIs are not capable of handling the SFC encapsulation (whether that be NSH or MPLS) and need the encapsulation stripped from the packets they are to process. In many cases, legacy SFIs that were once deployed as "bumps in the wire" fit into this category until they have been upgraded to be SFC-aware.

The job of an SFC Proxy is to remove and then reimpose SFC encapsulation so that the SFF is able to process as though it was communication with an SFC-aware SFI, and so that the SFI is unaware of the SFC encapsulation. In this regard, the job of an SFC Proxy is no different when NSH encapsulation is used and when MPLS encapsulation is used as described in this document, although (of course) it is different encapsulation bytes that must be removed and reimposed.

It should be noted that the SFC Proxy is a logical function. It could be implemented as a separate physical component on the path from the SFF to SFI, but it could be coresident with the SFF or it could be a component of the SFI. This is purely an implementation choice.

Note also that the delivery of metadata (see Section 11) requires specific processing if an SFC Proxy is in use. This is also no different when NSH or the MPLS encoding defined in this document is in use, and how it is handled will depend on how (or if) each non-SFC-aware SFI can receive metadata.

9. Control Plane Considerations

In order that a packet may be forwarded along an SFP several functional elements must be executed.

- o Discovery/advertisement of SFIs.
- o Computation of SFP.
- o Programming of Classifiers.
- o Advertisement of forwarding instructions.

Various approaches may be taken. These include a fully centralized model where SFFs report to a central controller the SFIs that they support, the central controller computes the SFP and programs the Classifiers, and (if the label swapping approach is taken) the central controller installs forwarding state in the SFFs that lie on the SFP.

Alternatively, a dynamic control plane may be used such as that described in [I-D.ietf-bess-nsh-bgp-control-plane]. In this case the SFFs use the control plane to advertise the SFIs that they support, a central controller computes the SFP and programs the Classifiers, and (if the label swapping approach is taken) the central controller uses the control plane to advertise the SFPs so that SFFs that lie on the SFP can install the necessary forwarding state.

10. Use of the Entropy Label

Entropy is used in ECMP situations to ensure that packets from the same flow travel down the same path, thus avoiding jitter or re-ordering issues within a flow.

Entropy is often determined by hashing on specific fields in a packet header such as the "five-tuple" in the IP and transport headers. However, when an MPLS label stack is present, the depth of the stack could be too large for some processors to correctly determine the entropy hash. This problem is addressed by the inclusion of an Entropy Label as described in [RFC6790].

When entropy is desired for packets as they are carried in MPLS tunnels over the underlay network, it is RECOMMENDED that an Entropy Label is included in the label stack immediately after the tunnel labels and before the SFC labels as shown in Figure 2 and Figure 3.

If an Entropy Label is present in an MPLS payload, it is RECOMMENDED that the initial Classifier use that value in an Entropy Label inserted in the label stack when the packet is forwarded (on the first tunnel) to the first SFF. In this case it is not necessary to remove the Entropy Label from the payload.

11. Metadata

Metadata is defined in [RFC7665] as providing "the ability to exchange context information between classifiers and SFs, and among SFs." [RFC8300] defines how this context information can be directly encoded in fields that form part of the NSH encapsulation.

The next two sections describe how metadata is associated with user data packets, and how metadata may be exchanged between SFC nodes in

the network, when using an MPLS encoding of the logical representation of the NSH.

It should be noted that the MPLS encoding is slightly less functional than the direct use of the NSH. Both methods support metadata that is "per-SFP" or "per-packet-flow" (see [I-D.farrel-sfc-convent] for definitions of these terms), but "per-packet" metadata (where the metadata must be carried on each packet because it differs from one packet to the next even on the same flow or SFP) is only supported using the NSH and not using the mechanisms defined in this document.

11.1. Indicating Metadata in User Data Packets

Metadata is achieved in the MPLS realization of the logical NSH by the use of an SFC Metadata Label which uses the Extended Special Purpose Label construct [RFC7274]. Thus, three label stack entries are present as shown in Figure 4:

- o The Extension Label (value 15)
- o An extended special purpose label called the Metadata Label Indicator (MLI) (value TBD1 by IANA)
- o The Metadata Label (ML).

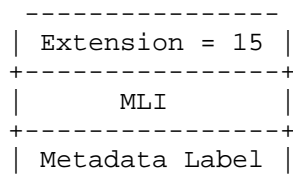


Figure 4: The MPLS SFC Metadata Label

The Metadata Label value is an index into a table of metadata that is programmed into the network using in-band or out-of-band mechanisms. Out-of-band mechanisms potentially include management plane and control plane solutions (such as [I-D.ietf-bess-nsh-bgp-control-plane]), but are out of scope for this document. The in-band mechanism is described in Section 11.2

The SFC Metadata Label (as a set of three labels as indicated in Figure 4) may be present zero, one, or more times in an MPLS SFC packet. For MPLS label swapping, the SFC Metadata Labels are placed immediately after the basic unit of MPLS label stack for SFC as shown

in Figure 5. For MPLS label stacking, the SFC Metadata Labels can be present zero, one, or more times and are placed at the bottom of the label stack as shown in Figure 6.

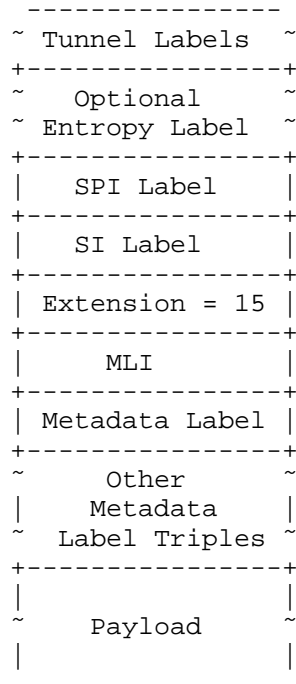


Figure 5: The MPLS SFC Label Stack for Label Swapping with Metadata Label

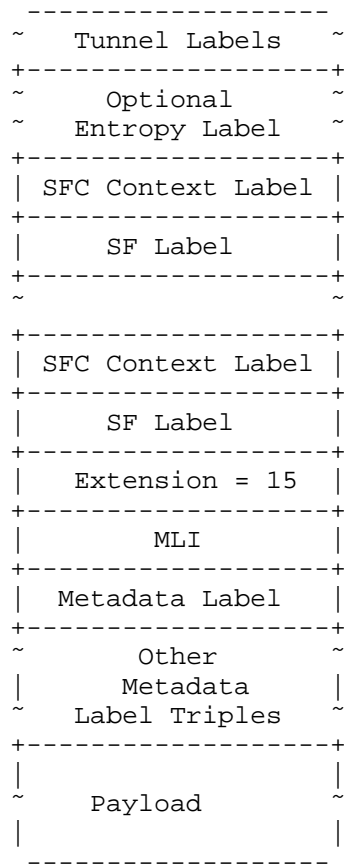


Figure 6: The MPLS SFC Label Stack for Label Stacking with Metadata Label

11.2. Inband Programming of Metadata

A mechanism for sending metadata associated with an SFP without a payload packet is described in [I-D.farrel-sfc-convent]. The same approach can be used in an MPLS network where the NSH is logically represented by an MPLS label stack.

The packet header is formed exactly as previously described in this document so that the packet will follow the SFP through the SFC network. However, instead of payload data, metadata is included after the bottom of the MPLS label stack. An Extended Special Purpose Label is used to indicate that the metadata is present. Thus, three label stack entries are present:

- o The Extension Label (value 15)
- o An extended special purpose label called the Metadata Present Indicator (MPI) (value TBD2 by IANA)
- o The Metadata Label (ML) that is associated with this metadata on this SFP and can be used to indicate the use of the metadata as described in Section 11.

The SFC Metadata Present Label, if present, is placed immediately after the last basic unit of MPLS label stack for SFC. The resultant label stacks are shown in Figure 7 for the MPLS label swapping case and Figure 8 for the MPLS label stacking case.

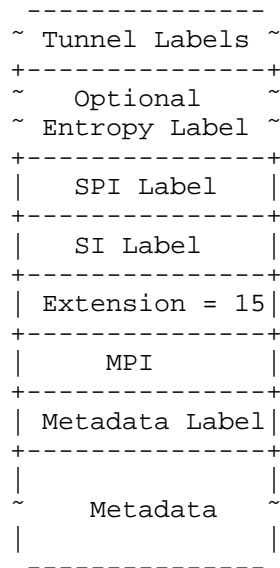


Figure 7: The MPLS SFC Label Stack for Label Swapping Carrying Metadata

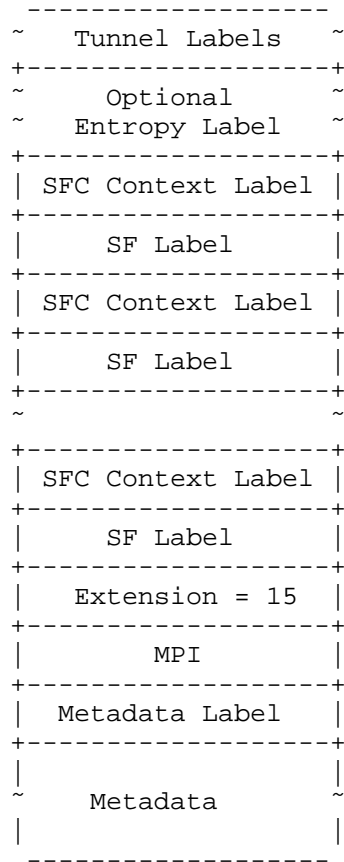


Figure 8: The MPLS SFC Label Stack for Label Stacking Carrying Metadata

In both cases the metadata is formatted as a TLV as shown in Figure 9.

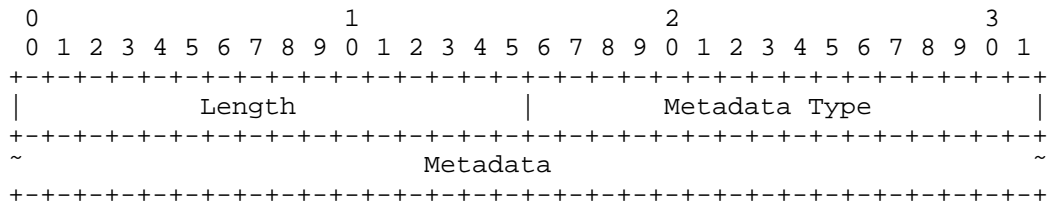


Figure 9: The Metadata TLV

The fields of this TLV are interpreted as follows:

Length: The length of the metadata carried in the Metadata field in octets not including any padding.

Metadata Type: The type of the metadata present. Values for this field are taken from the "MD Types" registry maintained by IANA and defined in [RFC8300].

Metadata: The actual metadata formatted as described in whatever document defines the metadata. This field is end-padded with zero to three octets of zeroes to take it up to a four octet boundary.

12. Worked Examples

Consider the simplistic MPLS SFC overlay network shown in Figure 10. A packet is classified for an SFP that will see it pass through two Service Functions, SFa and SFb, that are accessed through Service Function Forwarders SFFa and SFFb respectively. The packet is ultimately delivered to destination, D.

Let us assume that the SFP is computed and assigned the SPI of 239. The forwarding details of the SFP are distributed (perhaps using the mechanisms of [I-D.ietf-bess-nsh-bgp-control-plane]) so that the SFFs are programmed with the necessary forwarding instructions.

The packet progresses as follows:

- a. The Classifier assigns the packet to the SFP and imposes two label stack entries comprising a single basic unit of MPLS SFC representation:
 - * The higher label stack entry contains a label carrying the SPI value of 239.
 - * The lower label stack entry contains a label carrying the SI value of 255.

Further labels may be imposed to tunnel the packet from the Classifier to SFFa.

- b. When the packet arrives at SFFa it strips any labels associated with the tunnel that runs from the Classifier to SFFa. SFFa examines the top labels and matches the SPI/SI to identify that the packet should be forwarded to SFa. The packet is forwarded to SFa unmodified.
- c. SFa performs its designated function and returns the packet to SFFa.
- d. SFFa modifies the SI in the lower label stack entry (to 254) and uses the SPI/SI to look up the forwarding instructions. It sends the packet with two label stack entries:
 - * The higher label stack entry contains a label carrying the SPI value of 239.
 - * The lower label stack entry contains a label carrying the SI value of 254.

Further labels may be imposed to tunnel the packet from the SFFa to SFFb.

- e. When the packet arrives at SFFb it strips any labels associated with the tunnel from SFFa. SFFb examines the top labels and matches the SPI/SI to identify that the packet should be forwarded to SFb. The packet is forwarded to SFb unmodified.
- f. SFb performs its designated function and returns the packet to SFFb.
- g. SFFb modifies the SI in the lower label stack entry (to 253) and uses the SPI/SI to lookup up the forwarding instructions. It determines that it is the last SFF in the SFP so it strips the two SFC label stack entries and forwards the payload toward D using the payload protocol.

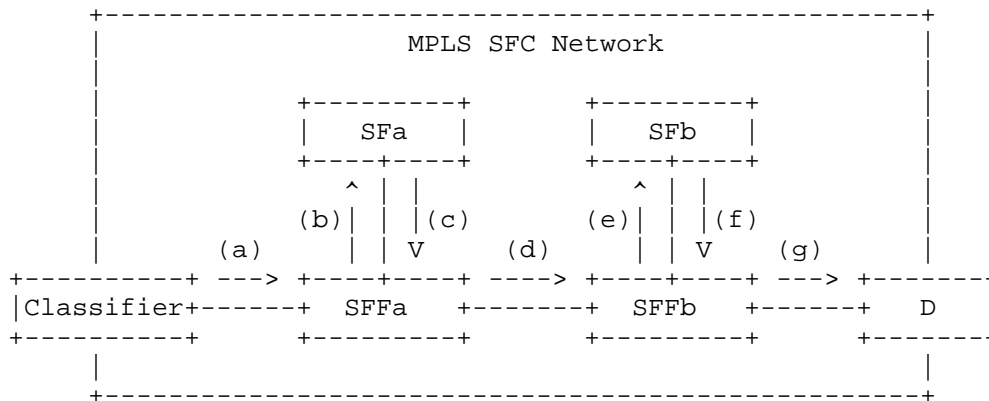


Figure 10: Service Function Chaining in an MPLS Network

Alternatively, consider the MPLS SFC overlay network shown in Figure 11. A packet is classified for an SFP that will see it pass through two Service Functions, SFx and SFy, that are accessed through Service Function Forwarders SFFx and SFFy respectively. The packet is ultimately delivered to destination, D.

Let us assume that the SFP is computed and assigned the SPI of 239. However, the forwarding state for the SFP is not distributed and installed in the network. Instead it will be attached to the individual packets using the MPLS label stack.

The packet progresses as follows:

1. The Classifier assigns the packet to the SFP and imposes two basic units of MPLS SFC representation to describe the full SFP:
 - * The top basic unit comprises two label stack entries as follows:
 - + The higher label stack entry contains a label carrying the SFC context.
 - + The lower label stack entry contains a label carrying the SF indicator for SFx.
 - * The lower basic unit comprises two label stack entries as follows:
 - + The higher label stack entry contains a label carrying the SFC context.

- + The lower label stack entry contains a label carrying the SF indicator for SFy.

Further labels may be imposed to tunnel the packet from the Classifier to SFFx.

2. When the packet arrives at SFFx it strips any labels associated with the tunnel from the Classifier. SFFx examines the top labels and matches the context/SF values to identify that the packet should be forwarded to SFx. The packet is forwarded to SFx unmodified.
3. SFx performs its designated function and returns the packet to SFFx.
4. SFFx strips the top basic unit of MPLS SFC representation revealing the next basic unit. It then uses the revealed context/SF values to determine how to route the packet to the next SFF, SFFy. It sends the packet with just one basic unit of MPLS SFC representation comprising two label stack entries:
 - * The higher label stack entry contains a label carrying the SFC context.
 - * The lower label stack entry contains a label carrying the SF indicator for SFy.

Further labels may be imposed to tunnel the packet from the SFFx to SFFy.

5. When the packet arrives at SFFy it strips any labels associated with the tunnel from SFFx. SFFy examines the top labels and matches the context/SF values to identify that the packet should be forwarded to SFy. The packet is forwarded to SFy unmodified.
6. SFy performs its designated function and returns the packet to SFFy.
7. SFFy strips the top basic unit of MPLS SFC representation revealing the payload packet. It forwards the payload toward D using the payload protocol.

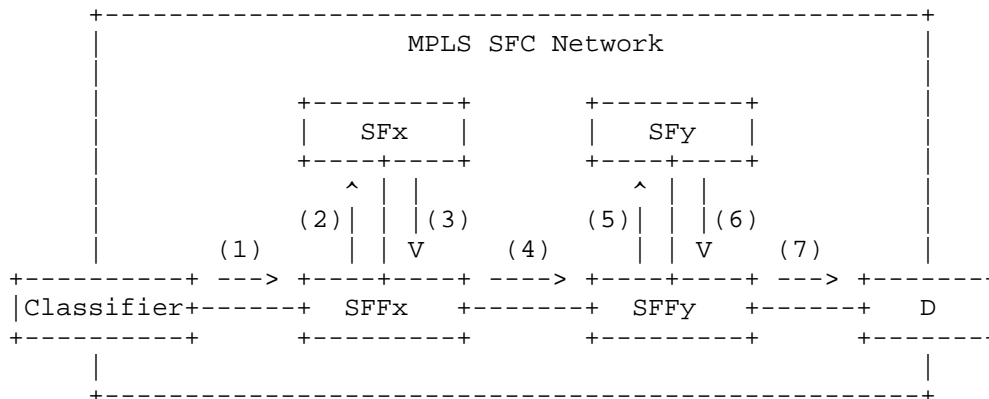


Figure 11: Service Function Chaining Using MPLS Label Stacking

13. Security Considerations

Discussion of the security properties of SFC networks can be found in [RFC7665]. Further security discussion for the NSH and its use is present in [RFC8300].

It is fundamental to the SFC design that the classifier is a trusted resource which determines the processing that the packet will be subject to, including for example the firewall. It is also fundamental to the MPLS design that packets are routed through the network using the path specified by the node imposing the labels, and that labels are swapped or popped correctly. Where an SF is not encapsulation aware the encapsulation may be stripped by an SFC proxy such that packet may exist as a native packet (perhaps IP) on the path between SFC proxy and SF, however this is an intrinsic part of the SFC design which needs to define how a packet is protected in that environment.

Additionally, where a tunnel is used to link two non-MPLS domains, the tunnel design needs to specify how the tunnel is secured.

Thus the security vulnerabilities are addressed (or should be addressed) in all the underlying technologies used by this design, which itself does not introduce any new security vulnerabilities.

14. IANA Considerations

This document requests IANA to make allocations from the "Extended Special-Purpose MPLS Label Values" subregistry of the "Special-

Purpose Multiprotocol Label Switching (MPLS) Label Values" registry as follows:

Value	Description	
TBD1	Metadata Label Indicator (MLI)	[This.I-D]
TBD2	Metadata Present Indicator (MPI)	[This.I-D]

15. Acknowledgements

This document derives ideas and text from [I-D.ietf-bess-nsh-bgp-control-plane].

The authors are grateful to all those who contributed to the discussions that led to this work: Loa Andersson, Andrew G. Malis, Alexander Vainshtein, Joel M. Halpern, Tony Przygienda, Stuart Mackie, Keyur Patel, and Jim Guichard. Loa Andersson provided helpful review comments.

Thanks to Loa Andersson, Lizhong Jin, Matthew Bocci, and Mach Chen for reviews of this text.

16. References

16.1. Normative References

- [I-D.farrel-sfc-convent]
Farrel, A. and J. Drake, "Operating the Network Service Header (NSH) with Next Protocol "None"", draft-farrel-sfc-convent-06 (work in progress), February 2018.
- [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>>.
- [RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <<https://www.rfc-editor.org/info/rfc7274>>.
- [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>>.

- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed.,
"Network Service Header (NSH)", RFC 8300,
DOI 10.17487/RFC8300, January 2018,
<<https://www.rfc-editor.org/info/rfc8300>>.

16.2. Informative References

- [I-D.ietf-bess-nsh-bgp-control-plane]
Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L.
Jalil, "BGP Control Plane for NSH SFC", draft-ietf-bess-
nsh-bgp-control-plane-03 (work in progress), March 2018.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol
Label Switching Architecture", RFC 3031,
DOI 10.17487/RFC3031, January 2001,
<<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and
L. Yong, "The Use of Entropy Labels in MPLS Forwarding",
RFC 6790, DOI 10.17487/RFC6790, November 2012,
<<https://www.rfc-editor.org/info/rfc6790>>.
- [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>>.

Authors' Addresses

Adrian Farrel
Juniper Networks

Email: afarrel@juniper.net

Stewart Bryant
Huawei

Email: stewart.bryant@gmail.com

John Drake
Juniper Networks

Email: jdrake@juniper.net

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2020

K. Raza, Ed.
R. Asati
Cisco Systems

X. Liu
Volta Networks

S. Esale
Juniper Networks

X. Chen
Huawei Technologies

H. Shah
Ciena Corporation

March 20, 2020

YANG Data Model for MPLS LDP
draft-ietf-mpls-ldp-yang-09

Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). The model also serves as the base model to define Multipoint LDP (mLDP) model.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

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 September 21, 2020.

Copyright Notice

Copyright (c) 2020 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
1.1. Base and Extended	3
2. Specification of Requirements	4
3. Overview	4
4. The Complete Tree	7
5. Configuration	16
5.1. Configuration Hierarchy	19
5.1.1. Global parameters	20
5.1.2. Capabilities parameters	20
5.1.3. Per-Address-Family parameters	20
5.1.4. Hello Discovery parameters	20
5.1.5. Peer parameters	21
5.1.6. Forwarding parameters	21
6. Operational State	22
6.1. Adjacency state	22
6.2. Peer state	23
6.3. Bindings state	24
6.4. Capabilities state	26
7. Notifications	27
8. Action	27
9. YANG Specification	27
9.1. Base	27
9.2. Extended	59
10. Security Considerations	80
10.1. YANG model	80
10.1.1. Writable nodes	81
10.1.2. Readable nodes	81
10.1.3. RPC operations	82
10.1.4. Notifications	83
11. IANA Considerations	83
12. Acknowledgments	83
13. Contributors	84
14. Normative References	84
15. Informative References	87
Appendix A. Data Tree Example	88
Authors' Addresses	92

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] [RFC7950] is a modular language that represents data structures in an XML tree format, and is used as a data modelling language for the NETCONF.

This document introduces a YANG data model for MPLS Label Distribution Protocol (LDP) [RFC5036]. This model also covers LDP IPv6 [RFC7552] and LDP capabilities [RFC5561] specifications.

The data model is defined for the following constructs that are used for managing the protocol:

- * Configuration
- * Operational State
- * Executables (Actions)
- * Notifications

This document is organized to define the data model for each of the above constructs in the sequence as listed above.

1.1. Base and Extended

The configuration and state items are divided into the following two broad categories:

- * Base
- * Extended

The "base" category contains the basic and fundamental features that are covered in LDP base specification [RFC5036] and constitute the minimum requirements for a typical base LDP deployment. Whereas, the "extended" category contains other non-base features. All the items in a base category are mandatory and hence no "if-feature" is allowed under the "base" category. The base and extended categories are defined in their own modules as described later.

The example of base feature includes the configuration of LDP lsr-id, enabling LDP interfaces, setting password for LDP session etc., whereas the examples of extended feature include inbound/outbound label policies, igp sync [RFC5443], downstream-on-demand etc. It is

worth highlighting that LDP IPv6 [RFC7552] is also categorized as an extended feature.

While "base" model support will suffice for small deployments, it is expected that large deployments will require both the "base" and "extended" models support from the vendors.

2. Specification of Requirements

In this document, the word "IP" is used to refer to both IPv4 and IPv6, unless otherwise explicitly stated. For example, "IP address family" should be read as "IPv4 and/or IPv6 address family".

3. Overview

This document defines two new modules for LDP YANG support:

- * "ietf-mpls-ldp" module that specifies the base LDP features and augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol defined in [RFC8349]. We define new identity 'mpls-ldp' for LDP and the model allows only a single instance of 'mpls-ldp'.
- * "ietf-mpls-ldp-extended" module that specifies the extended LDP features and augments the base LDP module.

It is to be noted that mLDP YANG model [I-D.ietf-mpls-mldp-yang] augments LDP base and extended modules to specify the mLDP specific base and extended features.

There are four types of containers in our module(s):

- * Read-Write parameters for configuration (Section 5)
- * Read-only parameters for operational state (Section 6)
- * Notifications for events (Section 7)
- * RPCs for executing commands to perform some action (Section 8)

The modules in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

Following diagram depicts high level LDP YANG tree organization and hierarchy:

```

+-- rw routing
  +-- rw control-plane-protocols
    +-- rw control-plane-protocol
      +-- rw mpls-ldp
        +-- rw ...
          +-- rw ... // base
          |   +-- rw ...
          |   +-- ro ...
          |   +--
          +-- ro ...
          |   +-- ro ...
          |   +-- ro ...
          |   +--
          +-- rw ldp-ext: .... // extended
          |   +-- rw ...
          |   +-- ro ...
          |   +--
          +-- ro ...
          |   +-- ro ...
          |   +-- ro ...

```

rpcs:

```

+-- x mpls-ldp-some_action
+-- x . . . . .

```

notifications:

```

+--- n mpls-ldp-some_event
+--- n ...

```

Figure 1: LDP YANG tree organization

Before going into data model details, it is important to take note of the following points:

- * This model aims to address only the core LDP parameters as per RFC specification, as well as well-known and widely deployed manageability controls (such as label filtering policies to apply filtering rules on the assignment, advertisement, and acceptance for label bindings). Any vendor specific feature should be defined in a vendor-specific augmentation of this model.
- * Multi-topology LDP [RFC7307] is beyond the scope of this document.

- * This model does not cover any applications running on top of LDP, nor does it cover any OAM procedures for LDP.
- * This model is a VPN Routing and Forwarding (VRF)-centric model. It is important to note that [RFC4364] defines VRF tables and default forwarding tables as different, however from a YANG modelling perspective this introduces unnecessary complications, hence we are treating the default forwarding table as just another VRF.
- * A "network-instance", as defined in [RFC8529], refers to a VRF instance (both default and non-default) within the scope of this model.
- * This model supports two address-families, namely "ipv4" and "ipv6".
- * This model assumes platform-wide label space (i.e. label space Id of zero). However, when Upstream Label assignment [RFC6389] is in use, an upstream assigned label is looked up in a Context-Specific label space as defined in [RFC5331].
- * The label and peer policies (including filters) are defined using prefix-set and neighbor-set respectively as defined in routing-policy model [I-D.ietf-rtgwg-policy-model].
- * This model uses the terms LDP "neighbor"/"adjacency", "session", and "peer" with the following semantics:
 - Neighbor/Adjacency: An LDP enabled LSR that is discovered through LDP discovery mechanisms.
 - Session: An LDP neighbor with whom a TCP connection has been established.
 - Peer: An LDP session which has successfully progressed beyond its initialization phase and is either already exchanging the bindings or is ready to do so.

It is to be noted that LDP Graceful Restart (GR) mechanisms defined in [RFC3478] allow keeping the exchanged bindings for some time after a session goes down with a peer. We call such a state belonging to a "stale" peer -- i.e. keeping peer bindings from a peer with whom currently there is either no connection established or connection is established but GR session is in recovery state. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A simplified graphical tree representation of base and extended LDP YANG data model is presented in Figure 2. The meaning of the symbols in these tree diagrams is defined in [RFC8340].

The actual YANG specification for base and extended modules is captured in Section 9.

While presenting the YANG tree view and actual specification, this document assumes readers' familiarity with the concepts of YANG modeling, its presentation and its compilation.

4. The Complete Tree

Following is a complete tree representation of configuration, state, notification, and RPC items under LDP base and extended modules.

```

module: ietf-mpls-ldp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw mpls-ldp
        +--rw global
          +--rw capability
            +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
            |   +--rw ldp-ext:enabled?    boolean
            +--rw ldp-ext:typed-wildcard-fec
            |   {capability-typed-wildcard-fec}?
            |   +--rw ldp-ext:enabled?    boolean
            +--rw ldp-ext:upstream-label-assignment
            |   {capability-upstream-label-assignment}?
            |   +--rw ldp-ext:enabled?    boolean
          +--rw graceful-restart
            +--rw enabled?                boolean
            +--rw reconnect-time?         uint16
            +--rw recovery-time?          uint16
            +--rw forwarding-holdtime?    uint16
            +--rw ldp-ext:helper-enabled? boolean
            |   {graceful-restart-helper-mode}?
          +--rw lsr-id?
            |   rt-types:router-id
          +--rw address-families
            +--rw ipv4!
            |   +--rw enabled?                boolean
            |   +--ro label-distribution-control-mode? enumeration
            |   +--ro bindings
            |   |   +--ro address* [address]
            |   |   |   +--ro address          inet:ipv4-address
            |   |   |   +--ro advertisement-type? advertised-received
            |   |   +--ro peer
  
```

```

|         +---ro lsr-id?                leafref
|         +---ro label-space-id?       leafref
+---ro fec-label* [fec]
|         +---ro fec        inet:ipv4-prefix
|         +---ro peer*
|             [lsr-id label-space-id advertisement-type]
|         +---ro lsr-id                leafref
|         +---ro label-space-id        leafref
|         +---ro advertisement-type
|             | advertised-received
|         +---ro label?
|             | rt-types:mpls-label
|         +---ro used-in-forwarding?    boolean
+---rw ldp-ext:label-policy
|   +---rw ldp-ext:advertise
|       | +---rw ldp-ext:egress-explicit-null
|       | | +---rw ldp-ext:enabled?    boolean
|       | +---rw ldp-ext:prefix-list?
|       |     prefix-list-ref
|   +---rw ldp-ext:accept
|       | +---rw ldp-ext:prefix-list?    prefix-list-ref
|   +---rw ldp-ext:assign
|       | {policy-label-assignment-config}?
|       | +---rw ldp-ext:independent-mode
|       | | +---rw ldp-ext:prefix-list?    prefix-list-ref
|       | +---rw ldp-ext:ordered-mode
|       |     {policy-ordered-label-config}?
|       | +---rw ldp-ext:egress-prefix-list?
|       |     prefix-list-ref
+---rw ldp-ext:transport-address?
|   inet:ipv4-address
+---rw ldp-ext:ipv6!
|   +---rw ldp-ext:enabled?
|       | boolean
+---rw ldp-ext:label-policy
|   +---rw ldp-ext:advertise
|       | +---rw ldp-ext:egress-explicit-null
|       | | +---rw ldp-ext:enabled?    boolean
|       | +---rw ldp-ext:prefix-list?
|       |     prefix-list-ref
|   +---rw ldp-ext:accept
|       | +---rw ldp-ext:prefix-list?    prefix-list-ref
|   +---rw ldp-ext:assign
|       | {policy-label-assignment-config}?
|       | +---rw ldp-ext:independent-mode
|       | | +---rw ldp-ext:prefix-list?    prefix-list-ref
|       | +---rw ldp-ext:ordered-mode
|       |     {policy-ordered-label-config}?

```

```

|         +---rw ldp-ext:egress-prefix-list?
|             prefix-list-ref
+---rw ldp-ext:transport-address
|     inet:ipv6-address
+---ro ldp-ext:label-distribution-control-mode?
|     enumeration
+---ro ldp-ext:bindings
|   +---ro ldp-ext:address* [address]
|       |   +---ro ldp-ext:address
|       |       |   inet:ipv6-address
|       |   +---ro ldp-ext:advertisement-type?
|       |       |   advertised-received
|       |   +---ro ldp-ext:peer
|       |       |   +---ro ldp-ext:lsr-id?           leafref
|       |       |   +---ro ldp-ext:label-space-id?  leafref
|       +---ro ldp-ext:fec-label* [fec]
|           +---ro ldp-ext:fec      inet:ipv6-prefix
|       +---ro ldp-ext:peer*
|           [lsr-id label-space-id advertisement-type]
|           +---ro ldp-ext:lsr-id           leafref
|           +---ro ldp-ext:label-space-id    leafref
|           +---ro ldp-ext:advertisement-type
|               |   advertised-received
|           +---ro ldp-ext:label?
|               |   rt-types:mpls-label
|           +---ro ldp-ext:used-in-forwarding?  boolean
+---rw ldp-ext:forwarding-nexthop
|   {forwarding-nexthop-config}?
+---rw ldp-ext:interfaces
|   +---rw ldp-ext:interface* [name]
|       |   +---rw ldp-ext:name           if:interface-ref
|       |   +---rw ldp-ext:address-family* [afi]
|       |       |   +---rw ldp-ext:afi       identityref
|       |       |   +---rw ldp-ext:ldp-disable?  boolean
+---rw ldp-ext:igp-synchronization-delay?  uint16
+---rw discovery
|   +---rw interfaces
|       |   +---rw hello-holdtime?    uint16
|       |   +---rw hello-interval?    uint16
|       +---rw interface* [name]
|           |   +---rw name
|           |       |   if:interface-ref
|           +---ro next-hello?           uint16
|       +---rw address-families
|           |   +---rw ipv4!
|           |       |   +---rw enabled?           boolean
|           |       |   +---ro hello-adjacencies
|           |       |   |   +---ro hello-adjacency* [adjacent-address]

```

```

+--ro adjacent-address
|   inet:ipv4-address
+--ro flag*                               identityref
+--ro hello-holdtime
|   +--ro adjacent?      uint16
|   +--ro negotiated?    uint16
|   +--ro remaining?     uint16
+--ro next-hello?      uint16
+--ro statistics
|   +--ro discontinuity-time
|   |   yang:date-and-time
|   +--ro hello-received?
|   |   yang:counter64
|   +--ro hello-dropped?
|   |   yang:counter64
+--ro peer
|   +--ro lsr-id?        leafref
|   +--ro label-space-id? leafref
+--rw ldp-ext:transport-address? union
+--rw ldp-ext:ipv6!
+--rw ldp-ext:enabled?          boolean
+--ro ldp-ext:hello-adjacencies
|   +--ro ldp-ext:hello-adjacency*
|   |   [adjacent-address]
|   +--ro ldp-ext:adjacent-address
|   |   inet:ipv6-address
|   +--ro ldp-ext:flag*
|   |   identityref
|   +--ro ldp-ext:hello-holdtime
|   |   +--ro ldp-ext:adjacent?      uint16
|   |   +--ro ldp-ext:negotiated?    uint16
|   |   +--ro ldp-ext:remaining?     uint16
|   +--ro ldp-ext:next-hello?      uint16
|   +--ro ldp-ext:statistics
|   |   +--ro ldp-ext:discontinuity-time
|   |   |   yang:date-and-time
|   |   +--ro ldp-ext:hello-received?
|   |   |   yang:counter64
|   |   +--ro ldp-ext:hello-dropped?
|   |   |   yang:counter64
|   +--ro ldp-ext:peer
|   |   +--ro ldp-ext:lsr-id?        leafref
|   |   +--ro ldp-ext:label-space-id? leafref
+--rw ldp-ext:transport-address? union
+--rw ldp-ext:hello-holdtime?      uint16
|   {per-interface-timer-config}?
+--rw ldp-ext:hello-interval?      uint16
|   {per-interface-timer-config}?

```



```

|         +---rw ldp-ext:igp-synchronization-delay?   uint16
|         {per-interface-timer-config}?
+---rw targeted
|   +---rw hello-holdtime?      uint16
|   +---rw hello-interval?     uint16
|   +---rw hello-accept
|   |   +---rw enabled?          boolean
|   |   +---rw ldp-ext:neighbor-list? neighbor-list-ref
|   |   {policy-targeted-discovery-config}?
+---rw address-families
|   +---rw ipv4!
|   |   +---ro hello-adjacencies
|   |   |   +---ro hello-adjacency*
|   |   |   |   [local-address adjacent-address]
|   |   |   |   +---ro local-address      inet:ipv4-address
|   |   |   |   +---ro adjacent-address   inet:ipv4-address
|   |   |   |   +---ro flag*              identityref
|   |   |   |   +---ro hello-holdtime
|   |   |   |   |   +---ro adjacent?      uint16
|   |   |   |   |   +---ro negotiated?    uint16
|   |   |   |   |   +---ro remaining?     uint16
|   |   |   |   +---ro next-hello?        uint16
|   |   |   |   +---ro statistics
|   |   |   |   |   +---ro discontinuity-time
|   |   |   |   |   |   yang:date-and-time
|   |   |   |   |   +---ro hello-received?
|   |   |   |   |   |   yang:counter64
|   |   |   |   |   +---ro hello-dropped?
|   |   |   |   |   |   yang:counter64
|   |   |   |   +---ro peer
|   |   |   |   |   +---ro lsr-id?          leafref
|   |   |   |   |   +---ro label-space-id? leafref
|   |   +---rw target* [adjacent-address]
|   |   |   +---rw adjacent-address   inet:ipv4-address
|   |   |   +---rw enabled?          boolean
|   |   |   +---rw local-address?     inet:ipv4-address
+---rw ldp-ext:ipv6!
|   +---ro ldp-ext:hello-adjacencies
|   |   +---ro ldp-ext:hello-adjacency*
|   |   |   [local-address adjacent-address]
|   |   |   +---ro ldp-ext:local-address
|   |   |   |   inet:ipv6-address
|   |   |   +---ro ldp-ext:adjacent-address
|   |   |   |   inet:ipv6-address
|   |   |   +---ro ldp-ext:flag*
|   |   |   |   identityref
|   |   |   +---ro ldp-ext:hello-holdtime
|   |   |   |   +---ro ldp-ext:adjacent?   uint16

```

```

|
|
|      +---ro ldp-ext:negotiated?      uint16
|      +---ro ldp-ext:remaining?      uint16
|      +---ro ldp-ext:next-hello?      uint16
|      +---ro ldp-ext:statistics
|      |      +---ro ldp-ext:discontinuity-time
|      |      |      yang:date-and-time
|      |      +---ro ldp-ext:hello-received?
|      |      |      yang:counter64
|      |      +---ro ldp-ext:hello-dropped?
|      |      |      yang:counter64
|      +---ro ldp-ext:peer
|      |      +---ro ldp-ext:lsr-id?      leafref
|      |      +---ro ldp-ext:label-space-id?      leafref
+---rw ldp-ext:target* [adjacent-address]
|      +---rw ldp-ext:adjacent-address
|      |      inet:ipv6-address
+---rw ldp-ext:enabled?      boolean
+---rw ldp-ext:local-address?
|      inet:ipv6-address
+---rw peers
+---rw authentication
|      +---rw (authentication-type)?
|      |      +---: (password)
|      |      |      +---rw key?      string
|      |      |      +---rw crypto-algorithm?      identityref
|      |      +---: (ldp-ext:key-chain) {key-chain}?
|      |      |      +---rw ldp-ext:key-chain?      key-chain:key-chain-ref
+---rw session-ka-holdtime?      uint16
+---rw session-ka-interval?      uint16
+---rw peer* [lsr-id label-space-id]
|      +---rw lsr-id      rt-types:router-id
|      +---rw label-space-id      uint16
|      +---rw authentication
|      |      +---rw (authentication-type)?
|      |      |      +---: (password)
|      |      |      |      +---rw key?      string
|      |      |      |      +---rw crypto-algorithm?      identityref
|      |      |      +---: (ldp-ext:key-chain) {key-chain}?
|      |      |      |      +---rw ldp-ext:key-chain?
|      |      |      |      |      key-chain:key-chain-ref
+---rw address-families
|      +---rw ipv4!
|      |      +---ro hello-adjacencies
|      |      |      +---ro hello-adjacency*
|      |      |      |      [local-address adjacent-address]
|      |      |      |      +---ro local-address      inet:ipv4-address
|      |      |      |      +---ro adjacent-address      inet:ipv4-address
|      |      |      +---ro flag*      identityref

```

```

+--ro hello-holdtime
|   +--ro adjacent?      uint16
|   +--ro negotiated?    uint16
|   +--ro remaining?     uint16
+--ro next-hello?        uint16
+--ro statistics
|   +--ro discontinuity-time
|       |   yang:date-and-time
+--ro hello-received?
|       |   yang:counter64
+--ro hello-dropped?
|       |   yang:counter64
+--ro interface?        if:interface-ref
+--rw ldp-ext:label-policy
+--rw ldp-ext:advertise
|   +--rw ldp-ext:prefix-list?    prefix-list-ref
+--rw ldp-ext:accept
|   +--rw ldp-ext:prefix-list?    prefix-list-ref
+--rw ldp-ext:ipv6!
+--ro ldp-ext:hello-adjacencies
|   +--ro ldp-ext:hello-adjacency*
|       |   [local-address adjacent-address]
+--ro ldp-ext:local-address
|   |   inet:ipv6-address
+--ro ldp-ext:adjacent-address
|   |   inet:ipv6-address
+--ro ldp-ext:flag*
|   |   identityref
+--ro ldp-ext:hello-holdtime
|   +--ro ldp-ext:adjacent?      uint16
|   +--ro ldp-ext:negotiated?    uint16
|   +--ro ldp-ext:remaining?     uint16
+--ro ldp-ext:next-hello?        uint16
+--ro ldp-ext:statistics
|   +--ro ldp-ext:discontinuity-time
|       |   yang:date-and-time
+--ro ldp-ext:hello-received?
|       |   yang:counter64
+--ro ldp-ext:hello-dropped?
|       |   yang:counter64
+--ro ldp-ext:interface?
|       |   if:interface-ref
+--rw ldp-ext:label-policy
+--rw ldp-ext:advertise
|   +--rw ldp-ext:prefix-list?    prefix-list-ref
+--rw ldp-ext:accept
|   +--rw ldp-ext:prefix-list?    prefix-list-ref
+--ro label-advertisement-mode

```

```

|   +--ro local?          label-adv-mode
|   +--ro peer?           label-adv-mode
|   +--ro negotiated?     label-adv-mode
+--ro next-keep-alive?    uint16
+--ro received-peer-state
|   +--ro graceful-restart
|   |   +--ro enabled?      boolean
|   |   +--ro reconnect-time?  uint16
|   |   +--ro recovery-time?  uint16
|   +--ro capability
|   |   +--ro end-of-lib
|   |   |   +--ro enabled?  boolean
|   |   +--ro typed-wildcard-fec
|   |   |   +--ro enabled?  boolean
|   |   +--ro upstream-label-assignment
|   |   |   +--ro enabled?  boolean
+--ro session-holdtime
|   +--ro peer?          uint16
|   +--ro negotiated?    uint16
|   +--ro remaining?     uint16
+--ro session-state?     enumeration
+--ro tcp-connection
|   +--ro local-address?  inet:ip-address
|   +--ro local-port?     inet:port-number
|   +--ro remote-address? inet:ip-address
|   +--ro remote-port?    inet:port-number
+--ro up-time?
|   rt-types:timeticks64
+--ro statistics
|   +--ro discontinuity-time  yang:date-and-time
|   +--ro received
|   |   +--ro total-octets?  yang:counter64
|   |   +--ro total-messages? yang:counter64
|   |   +--ro address?       yang:counter64
|   |   +--ro address-withdraw? yang:counter64
|   |   +--ro initialization? yang:counter64
|   |   +--ro keepalive?     yang:counter64
|   |   +--ro label-abort-request? yang:counter64
|   |   +--ro label-mapping? yang:counter64
|   |   +--ro label-release? yang:counter64
|   |   +--ro label-request? yang:counter64
|   |   +--ro label-withdraw? yang:counter64
|   |   +--ro notification?  yang:counter64
|   +--ro sent
|   |   +--ro total-octets?  yang:counter64
|   |   +--ro total-messages? yang:counter64
|   |   +--ro address?       yang:counter64
|   |   +--ro address-withdraw? yang:counter64

```

```

| | | +--ro initialization?          yang:counter64
| | | +--ro keepalive?             yang:counter64
| | | +--ro label-abort-request?   yang:counter64
| | | +--ro label-mapping?        yang:counter64
| | | +--ro label-release?        yang:counter64
| | | +--ro label-request?        yang:counter64
| | | +--ro label-withdraw?       yang:counter64
| | | +--ro notification?         yang:counter64
| | +--ro total-addresses?        uint32
| | +--ro total-labels?           uint32
| | +--ro total-fec-label-bindings? uint32
+--rw ldp-ext:admin-down?         boolean
|   {per-peer-admin-down}?
+--rw ldp-ext:graceful-restart
|   {per-peer-graceful-restart-config}?
+--rw ldp-ext:enabled?            boolean
+--rw ldp-ext:reconnect-time?     uint16
+--rw ldp-ext:recovery-time?      uint16
+--rw ldp-ext:session-ka-holdtime? uint16
|   {per-peer-session-attributes-config}?
+--rw ldp-ext:session-ka-interval? uint16
|   {per-peer-session-attributes-config}?
+--rw ldp-ext:session-downstream-on-demand
|   {session-downstream-on-demand-config}?
+--rw ldp-ext:enabled?            boolean
+--rw ldp-ext:peer-list?          peer-list-ref
+--rw ldp-ext:dual-stack-transport-preference
|   {peers-dual-stack-transport-preference}?
+--rw ldp-ext:max-wait?           uint16
+--rw ldp-ext:prefer-ipv4!
|   +--rw ldp-ext:peer-list?      peer-list-ref

rpccs:
+---x mpls-ldp-clear-peer
|   +---w input
|   |   +---w protocol-name?      leafref
|   |   +---w lsr-id?             leafref
|   |   +---w label-space-id?     leafref
+---x mpls-ldp-clear-hello-adjacency
|   +---w input
|   |   +---w hello-adjacency
|   |   |   +---w protocol-name?  leafref
|   |   |   +---w (hello-adjacency-type)?
|   |   |   |   +--:(targeted)
|   |   |   |   |   +---w targeted!
|   |   |   |   |   +---w target-address?  inet:ip-address
|   |   |   +--:(link)
|   |   +---w link!

```

```

|               +---w next-hop-interface?  leafref
|               +---w next-hop-address?    inet:ip-address
+---x mpls-ldp-clear-peer-statistics
  +---w input
    +---w protocol-name?  leafref
    +---w lsr-id?         leafref
    +---w label-space-id? leafref

notifications:
+---n mpls-ldp-peer-event
  +--ro event-type?  oper-status-event-type
  +--ro peer
    +--ro protocol-name?  leafref
    +--ro lsr-id?         leafref
    +--ro label-space-id? leafref
+---n mpls-ldp-hello-adjacency-event
  +--ro event-type?      oper-status-event-type
  +--ro protocol-name?   leafref
  +--ro (hello-adjacency-type)?
    +--:(targeted)
      +--ro targeted
        +--ro target-address?  inet:ip-address
    +--:(link)
      +--ro link
        +--ro next-hop-interface?  if:interface-ref
        +--ro next-hop-address?    inet:ip-address
+---n mpls-ldp-fec-event
  +--ro event-type?      oper-status-event-type
  +--ro protocol-name?   leafref
  +--ro fec?             inet:ip-prefix

```

Figure 2: Complete Tree

5. Configuration

This specification defines the configuration parameters for base LDP as specified in [RFC5036] and LDP IPv6 [RFC7552]. Moreover, it incorporates provisions to enable LDP Capabilities [RFC5561], and defines some of the most significant and commonly used capabilities such as Typed Wildcard FEC [RFC5918], End-of-LIB [RFC5919], and LDP Upstream Label Assignment [RFC6389].

This model augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol that is defined in [RFC8349] and follows NMDA as mentioned earlier.

Following is the high-level configuration organization for base LDP module:

```

augment /rt:routing/rt:control-plane-protocols:
  /rt:control-plane-protocol:
    +-- mpls-ldp
      +-- global
        +-- ...
        +-- ...
        +-- address-families
          +-- ipv4
            +-- . . .
            +-- . . .
        +-- capability
          +-- ...
          +-- ...
      +-- discovery
        +-- interfaces
          +-- ...
          +-- ...
          +-- interface* [interface]
            +-- ...
            +-- address-families
              +-- ipv4
                +-- ...
                +-- ...
        +-- targeted
          +-- ...
          +-- address-families
            +-- ipv4
              +- target* [adjacent-address]
                +- ...
                +- ...
      +-- peers
        +-- ...
        +-- ...
        +-- peer* [lsr-id label-space-id]
          +-- ...
          +-- ...

```

Figure 3: Base Configuration organization

Following is the high-level configuration organization for extended LDP:

```

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protoc
ol
    +-- mpls-ldp
    +-- global
    |   +-- ...
    |   +-- ...
    |   +-- address-families
    |   |   +-- ipv4
    |   |   |   +-- . . .
    |   |   |   +-- . . .
    |   |   |   +-- label-policy
    |   |   |   +-- ...
    |   |   |   +-- ...
    |   |   +-- ipv6
    |   |   |   +-- . . .
    |   |   |   +-- . . .
    |   |   |   +-- label-policy
    |   |   |   +-- ...
    |   |   |   +-- ...
    |   +-- capability
    |   |   +-- ...
    |   |   +-- ...
    |   +-- discovery
    |   |   +-- interfaces
    |   |   |   +-- ...
    |   |   |   +-- ...
    |   |   |   +-- interface* [interface]
    |   |   |   |   +-- ...
    |   |   |   |   +-- address-families
    |   |   |   |   |   +-- ipv4
    |   |   |   |   |   |   +-- ...
    |   |   |   |   |   |   +-- ...
    |   |   |   |   +-- ipv6
    |   |   |   |   |   +-- ...
    |   |   |   |   +-- ...
    |   |   +-- targetted
    |   |   |   +-- ...
    |   |   |   +-- address-families
    |   |   |   |   +-- ipv6
    |   |   |   |   |   +-- target* [adjacent-address]
    |   |   |   |   |   +-- ...
    |   |   |   |   +-- ...
    +-- forwarding-nextthop
    |   +-- ...
    |   +-- ...
    +-- peers
    |   +-- ...
    |   +-- ...
    +-- peer*

```



```

+-- ...
+-- ...
+-- label-policy
|   +-- ..
+-- address-families
    +-- ipv4
    |   +-- ...
    +-- ipv6
        +-- ...

```

Figure 4: Extended Configuration organization

Given the configuration hierarchy, the model allows inheritance such that an item in a child tree is able to derive value from a similar or related item in one of the parents. For instance, hello holdtime can be configured per-VRF or per-VRF-interface, thus allowing inheritance as well flexibility to override with a different value at any child level.

5.1. Configuration Hierarchy

LDP module resides under a network-instance and the scope of any LDP configuration defined under this tree is per network-instance (per-VRF). This configuration is further divided into sub categories as follows.

- * Global parameters
- * Per-address-family parameters
- * LDP Capabilities parameters
- * Hello Discovery parameters
 - interfaces
 - o Global
 - o Per-interface: Global
 - o Per-interface: Per-address-family
 - targeted
 - o Global

- o Per-address-family: Per-target
- * Peer parameters
 - Global
 - Per-peer: Global
 - Per-peer: Per-address-family

- * Forwarding parameters

Following subsections briefly explain these configuration areas.

5.1.1. Global parameters

There are configuration items that are available directly under a VRF instance and do not fall under any other sub tree. Example of such a parameter is LDP LSR Id that is typically configured per VRF. To keep legacy LDP features and applications working in an LDP IPv4 networks with this model, this document recommends an operator to pick a routable IPv4 unicast address (within a routing domain) as an LSR Id.

5.1.2. Capabilities parameters

This container falls under the global tree and holds the LDP capabilities that are to be enabled for certain features. By default, an LDP capability is disabled unless explicitly enabled. These capabilities are typically used to negotiate with LDP peer(s) the support/non-support related to a feature and its parameters. The scope of a capability enabled under this container applies to all LDP peers in the given VRF instance. There is also a peer level capability container that is provided to override a capability that is enabled/specified at VRF level.

5.1.3. Per-Address-Family parameters

Any LDP configuration parameter related to IP address family (AF) whose scope is VRF wide is configured under this tree. The examples of per-AF parameters include enabling LDP for an address family, prefix-list based label policies, and LDP transport address.

5.1.4. Hello Discovery parameters

This container is used to hold LDP configuration related to Hello and discovery process for both basic (link) and extended (targeted) discovery.

The "interfaces" is a container to configure parameters related to VRF interfaces. There are parameters that apply to all interfaces (such as hello timers), as well as parameters that can be configured per-interface. Hence, an interface list is defined under "interfaces" container. The model defines parameters to configure per-interface non AF related items, as well as per-interface per-AF items. The example of the former is interface hello timers, and example of the latter is enabling hellos for a given AF under an interface.

The "targeted" container under a VRF instance allows to configure LDP targeted discovery related parameters. Within this container, the "target" list provides a means to configure multiple target addresses to perform extended discovery to a specific destination target, as well as to fine-tune the per-target parameters.

5.1.5. Peer parameters

This container is used to hold LDP configuration related to LDP sessions and peers under a VRF instance. This container allows to configure parameters that either apply on VRF's all peers or a subset (peer-list) of VRF peers. The example of such parameters include authentication password, session KA timers etc. Moreover, the model also allows per-peer parameter tuning by specifying a "peer" list under the "peers" container. A peer is uniquely identified by its LSR Id.

Like per-interface parameters, some per-peer parameters are AF-agnostic (i.e. either non AF related or apply to both IP address families), and some that belong to an AF. The example of the former is per-peer session password configuration, whereas the example of the latter is prefix-list based label policies (inbound and outbound) that apply to a given peer.

5.1.6. Forwarding parameters

This container is used to hold configuration used to control LDP forwarding behavior under a VRF instance. One example of a configuration under this container is when a user wishes to enable neighbor discovery on an interface but wishes to disable use of the same interface as forwarding nexthop. This example configuration makes sense only when there are more than one LDP enabled interfaces towards the neighbor.

6. Operational State

Operational state of LDP can be queried and obtained from read-only state containers that fall under the same tree (/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol) as the configuration.

Following are main areas for which LDP operational state is defined:

- * Neighbor Adjacencies
- * Peer
- * Bindings (FEC-label and address)
- * Capabilities

6.1. Adjacency state

Neighbor adjacencies are per address-family hello adjacencies that are formed with neighbors as result of LDP basic or extended discovery. In terms of organization, there is a source of discovery (e.g. interface or target address) along with its associated parameters and one or more discovered neighbors along with neighbor discovery related parameters. For the basic discovery, there could be more than one discovered neighbor for a given source (interface), whereas there is at most one discovered neighbor for an extended discovery source (local-address and target-address). It is also to be noted that the reason for a targeted neighbor adjacency could be either an active source (locally configured targeted) or passive source (to allow any incoming extended/targeted hellos). A neighbor/adjacency record also contains session-state that helps highlight whether a given adjacency has progressed to subsequent session level or to eventual peer level.

Following captures high level tree hierarchy for neighbor adjacency state. The tree is shown for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw discovery
    +--rw interfaces
      +--rw interface* [interface]
        +--rw address-families
          +--rw ipv4
            +--ro hello-adjacencies
              +--ro hello-adjacencies* [adjacent-address]
                +--ro adjacent-address
                  . . . .
            +--rw targeted
              +--rw address-families
                +--rw ipv4
                  +--ro hello-adjacencies
                    +--ro hello-adjacencies*
                      | [local-address adjacent-address]
                    +--ro local-address
                      +--ro adjacent-address
                        . . . .
                  . . . .
          . . . .
    . . . .

```

Figure 5: Adjacency state

6.2. Peer state

Peer related state is presented under peers tree. This is one of the core state that provides info on the session related parameters (mode, authentication, KA timeout etc.), TCP connection info, hello adjacencies for the peer, statistics related to messages and bindings, and capabilities exchange info.

Following captures high level tree hierarchy for peer state. The peer's hello adjacencies tree is shown for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw peers
    +--rw peer* [lsr-id label-space-id]
      +--rw lsr-id
      +--rw label-space-id
      +--ro label-advertisement-mode
      +--ro session-state
      +--ro tcp-connection
      +--ro session-holdtime?
      +--ro up-time
      +-- . . . .
      +--ro address-families
        +--ro ipv4
          +--ro hello-adjacencies
            +--ro hello-adjacencies*
              [local-address adjacent-address]
              . . . .
              . . . .
      +--ro received-peer-state
        +--ro . . . .
        +--ro capability
          +--ro . . . .
      +--ro statistics
        +-- . . . .
        +-- received
          +-- ...
        +-- sent
          +-- ...

```

Figure 6: Peer state

6.3. Bindings state

Binding state provides information on LDP FEC-label bindings as well as address binding for both inbound (received) as well as outbound (advertised) direction. FEC-label bindings are presented as a FEC-centric view, and address bindings are presented as an address-centric view:

```

FEC-Label bindings:
  FEC 203.0.113.1/32:
    advertised: local-label 16000
      peer 192.0.2.1:0
      peer 192.0.2.2:0
      peer 192.0.2.3:0
    received:
      peer 192.0.2.1:0, label 16002, used-in-forwarding=Yes
      peer 192.0.2.2:0, label 17002, used-in-forwarding=No
  FEC 203.0.113.2/32:
    . . . .
  FEC 198.51.100.0/24:
    . . . .
  FEC 2001:db8:0:2::
    . . . .
  FEC 2001:db8:0:3::
    . . . .

Address bindings:
  Addr 192.0.2.10:
    advertised
  Addr 2001:db8:0:10::
    advertised

  Addr 192.0.2.1:
    received, peer 192.0.2.1:0
  Addr 192.0.2.2:
    received, peer 192.0.2.2:0
  Addr 192.0.2.3:
    received, peer 192.0.2.3:0
  Addr 2001:db8:0:2::
    received, peer 192.0.2.2:0
  Addr 2001:db8:0:3::
    received, peer 192.0.2.3:0

```

Figure 7: Example Bindings

Note that all local addresses are advertised to all peers and hence no need to provide per-peer information for local address advertisement. Furthermore, note that it is easy to derive a peer-centric view for the bindings from the information already provided in this model.

Following captures high level tree hierarchy for bindings state. The tree shown below is for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw global
    +--rw address-families
      +--rw ipv4
        +--ro bindings
          +--ro address* [address]
            +--ro address (ipv4-address or ipv6-address)
            +--ro advertisement-type? advertised-received
            +--ro peer? leafref
          +--ro fec-label* [fec]
            +--ro fec (ipv4-prefix or ipv6-prefix)
            +--ro peer* [peer advertisement-type]
              +--ro peer leafref
              +--ro advertisement-type? advertised-received
              +--ro label? mpls:mpls-label
              +--ro used-in-forwarding? boolean

```

Figure 8: Bindings state

6.4. Capabilities state

LDP capabilities state comprise two types of information - global information (such as timer etc.), and per-peer information.

Following captures high level tree hierarchy for LDP capabilities state.

```

+--rw mpls-ldp!
  +--rw peers
    +--rw peer* [lsr-id label-space-id]
      +--rw lsr-id yang:dotted-quad
      +--rw label-space-id
      +--ro received-peer-state
        +--ro capability
          +--ro . . . .
          +--ro . . . .

```

Figure 9: Capabilities state

7. Notifications

This model defines a list of notifications to inform client of important events detected during the protocol operation. These events include events related to changes in the operational state of an LDP peer, hello adjacency, and FEC etc. It is to be noted that an LDP FEC is treated as operational (up) as long as it has at least 1 NHLFE (Next Hop Label Forwarding Entry) with outgoing label.

A simplified graphical representation of the data model for LDP notifications is shown in Figure 2.

8. Action

This model defines a list of rpcs that allow performing an action or executing a command on the protocol. For example, it allows to clear (reset) LDP peers, hello-adjacencies, and statistics. The model makes an effort to provide different level of control so that a user is able to either clear all, or clear all for a given type, or clear a specific entity.

A simplified graphical representation of the data model for LDP actions is shown in Figure 2.

9. YANG Specification

Following sections specify the actual YANG (module) specification for LDP constructs defined earlier in the document.

9.1. Base

This YANG module imports types defined in [RFC6991], [RFC8349], [RFC8294], [RFC8343], and [RFC8344].

```
<CODE BEGINS> file "ietf-mpls-ldp@2020-02-25.yang"
```

```
// RFC Editor: replace the above date 2020-02-25 with the date of
// publication and remove this note.
```

```
module ietf-mpls-ldp {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
  prefix "ldp";

  import ietf-inet-types {
```

```
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
}

import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
}

import ietf-routing {
    prefix "rt";
    reference
        "RFC 8349: A YANG Data Model for Routing Management (NMDA
        version)";
}

import ietf-routing-types {
    prefix "rt-types";
    reference
        "RFC 8294: Common YANG Data Types for the Routing Area";
}

import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-ip {
    prefix "ip";
    reference "RFC 7277: A YANG Data Model for IP Management";
}

import ietf-key-chain {
    prefix "key-chain";
    reference "RFC 8177: YANG Data Model for Key Chains";
}

organization
    "IETF MPLS Working Group";
contact
    "WG Web:  <http://tools.ietf.org/wg/mpls/>
    WG List:  <mailto:mpls@ietf.org>

    Editor:   Kamran Raza
              <mailto:skraza@cisco.com>

    Editor:   Rajiv Asati
              <mailto:rajiva@cisco.com>
```

Editor: Xufeng Liu
<mailto:xufeng.liu.ietf@gmail.com>

Editor: Santosh Esale
<mailto:sesale@juniper.net>

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>;

description

"This YANG module defines the essential components for the management of Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). It is also the base model to be augmented for Multipoint LDP (mLDP).

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 (<http://trustee.ietf.org/license-info>).

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

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
revision 2020-02-25 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Typedefs
 */
typedef advertised-received {
  type enumeration {
```

```
    enum advertised {
      description "Advertised information.";
    }
    enum received {
      description "Received information.";
    }
  }
  description
    "Received or advertised.";
}

typedef downstream-upstream {
  type enumeration {
    enum downstream {
      description "Downstream information.";
    }
    enum upstream {
      description "Upstream information.";
    }
  }
  description
    "Downstream or upstream.";
}

typedef label-adv-mode {
  type enumeration {
    enum downstream-unsolicited {
      description "Downstream Unsolicited.";
    }
    enum downstream-on-demand {
      description "Downstream on Demand.";
    }
  }
  description
    "Label Advertisement Mode.";
}

typedef oper-status-event-type {
  type enumeration {
    enum up {
      value 1;
      description
        "Operational status changed to up.";
    }
    enum down {
      value 2;
      description
        "Operational status changed to down.";
    }
  }
}
```

```
    }
  }
  description "Operational status event type for notifications.";
}

/*
 * Identities
 */
identity mpls-ldp {
  base rt:control-plane-protocol;
  description
    "LDP protocol.";
  reference
    "RFC 5036: LDP Specification";
}

identity adjacency-flag-base {
  description "Base type for adjacency flags.";
}

identity adjacency-flag-active {
  base adjacency-flag-base;
  description
    "This adjacency is configured and actively created.";
}

identity adjacency-flag-passive {
  base adjacency-flag-base;
  description
    "This adjacency is not configured and passively accepted.";
}

/*
 * Groupings
 */
grouping adjacency-state-attributes {
  description
    "The operational state attributes of an LDP Hello adjacency,
    which can used for basic and extended discoveris, in IPv4 and
    IPv6 address families.";

  leaf-list flag {
    type identityref {
      base adjacency-flag-base;
    }
    description
      "On or more flags to indicate whether the adjacency is
```

```
        actively created, passively accepted, or both.";
    }
    container hello-holdtime {
        description
            "Containing Hello holdtime state information.";
        leaf adjacent {
            type uint16;
            units seconds;
            description
                "The holdtime value learned from the adjacent LSR.";
        }
        leaf negotiated {
            type uint16;
            units seconds;
            description
                "The holdtime negotiated between this LSR and the adjacent
                LSR.";
        }
        leaf remaining {
            type uint16;
            units seconds;
            description
                "The time remaining until the holdtime timer expires.";
        }
    }

    leaf next-hello {
        type uint16;
        units seconds;
        description
            "The time when the next Hello message will be sent.";
    }

    container statistics {
        description
            "Statistics objects.";

        leaf discontinuity-time {
            type yang:date-and-time;
            mandatory true;
            description
                "The time on the most recent occasion at which any one or
                more of this interface's counters suffered a
                discontinuity.  If no such discontinuities have occurred
                since the last re-initialization of the local management
                subsystem, then this node contains the time the local
                management subsystem re-initialized itself.";
        }
    }
}
```

```
    leaf hello-received {
      type yang:counter64;
      description
        "The number of Hello messages received.";
    }
    leaf hello-dropped {
      type yang:counter64;
      description
        "The number of Hello messages dropped.";
    }
  } // statistics
} // adjacency-state-attributes

grouping basic-discovery-timers {
  description
    "The timer attributes for basic discovery, used in the
    per-interface setting and in the all-interface setting.";

  leaf hello-holdtime {
    type uint16 {
      range 15..3600;
    }
    units seconds;
    description
      "The time interval for which a LDP link Hello adjacency
      is maintained in the absence of link Hello messages from
      the LDP neighbor.
      This leaf may be configured at the per-interface level or
      the global level, with precedence given to the value at the
      per-interface level. If the leaf is not configured at
      either level, the default value at the global level is
      used.";
  }
  leaf hello-interval {
    type uint16 {
      range 5..1200;
    }
    units seconds;
    description
      "The interval between consecutive LDP link Hello messages
      used in basic LDP discovery.
      This leaf may be configured at the per-interface level or
      the global level, with precedence given to the value at the
      per-interface level. If the leaf is not configured at
      either level, the default value at the global level is
      used.";
  }
} // basic-discovery-timers
```

```
grouping binding-address-state-attributes {
  description
    "Operational state attributes of an address binding, used in
    IPv4 and IPv6 address families.";

  leaf advertisement-type {
    type advertised-received;
    description
      "Received or advertised.";
  }
  container peer {
    when "../advertisement-type = 'received'" {
      description
        "Applicable for received address.";
    }
    description
      "LDP peer from which this address is received.";
    uses ldp-peer-ref-from-binding;
  }
} // binding-address-state-attributes

grouping binding-label-state-attributes {
  description
    "Operational state attributes for a FEC-label binding, used in
    IPv4 and IPv6 address families.";

  list peer {
    key "lsr-id label-space-id advertisement-type";
    description
      "List of advertised and received peers.";
    uses ldp-peer-ref-from-binding {
      description
        "The LDP peer from which this binding is received, or to
        which this binding is advertised.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
    leaf advertisement-type {
      type advertised-received;
      description
        "Received or advertised.";
    }
    leaf label {
      type rt-types:mpls-label;
      description
        "Advertised (outbound) or received (inbound)
        label.";
    }
  }
}
```



```
    leaf used-in-forwarding {
        type boolean;
        description
            "'true' if the label is used in forwarding.";
    }
} // peer
} // binding-label-state-attributes

grouping graceful-restart-attributes-per-peer {
    description
        "Per peer graceful restart attributes.
        On the local side, these attributes are configuration and
        operational state data. On the peer side, these attributes
        are operational state data received from the peer.";

    container graceful-restart {
        description
            "Attributes for graceful restart.";
        leaf enabled {
            type boolean;
            description
                "Enable or disable graceful restart.
                This leaf may be configured at the per-peer level or the
                global level, with precedence given to the value at the
                per-peer level. If the leaf is not configured at either
                level, the default value at the global level is used.";
        }
        leaf reconnect-time {
            type uint16 {
                range 10..1800;
            }
            units seconds;
            description
                "Specifies the time interval that the remote LDP peer
                must wait for the local LDP peer to reconnect after the
                remote peer detects the LDP communication failure.
                This leaf may be configured at the per-peer level or the
                global level, with precedence given to the value at the
                per-peer level. If the leaf is not configured at either
                level, the default value at the global level is used.";
        }
        leaf recovery-time {
            type uint16 {
                range 30..3600;
            }
            units seconds;
            description
                "Specifies the time interval, in seconds, that the remote
```

```
        LDP peer preserves its MPLS forwarding state after
        receiving the Initialization message from the restarted
        local LDP peer.
        This leaf may be configured at the per-peer level or the
        global level, with precedence given to the value at the
        per-peer level. If the leaf is not configured at either
        level, the default value at the global level is used.";
    }
} // graceful-restart
} // graceful-restart-attributes-per-peer

grouping ldp-interface-ref {
  description
    "Defining a reference to LDP interface.";

  leaf name {
    type if:interface-ref;
    must "(/if:interfaces/if:interface[if:name=current()]/ip:ipv4)"
      + " or "
      + "(/if:interfaces/if:interface[if:name=current()]/ip:ipv6)"
    {
      description "Interface is IPv4 or IPv6.";
    }
    description
      "The name of an LDP interface.";
  }
}

grouping ldp-peer-ref-absolute {
  description
    "An absolute reference to an LDP peer, by the LDP ID, which
    consists of the LSR ID and the Label Space ID.";

  leaf protocol-name {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rt:name";
    }
    description
      "The name of the LDP protocol instance.";
  }
  leaf lsr-id {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol"
        + "[rt:name=current()]/../protocol-name/"
        + "ldp:mpls-ldp/ldp:peers/ldp:peer/ldp:lsr-id";
    }
  }
}
```

```
        description
            "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol"
                + "[rt:name=current()/../protocol-name]/"
                + "ldp:mpls-ldp/ldp:peers/"
                + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
                + "ldp:label-space-id";
        }
        description
            "The Label Space ID of the peer, as a portion of the peer
            LDP ID.";
    }
} // ldp-peer-ref-absolute

grouping ldp-peer-ref-from-binding {
    description
        "A relative reference to an LDP peer, by the LDP ID, which
        consists of the LSR ID and the Label Space ID.";

    leaf lsr-id {
        type leafref {
            path "../..../..../..../ldp:peers/ldp:peer/ldp:lsr-id";
        }
        description
            "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
        type leafref {
            path "../..../..../..../ldp:peers/"
                + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
                + "ldp:label-space-id";
        }
        description
            "The Label Space ID of the peer, as a portion of the peer
            LDP ID.";
    }
} // ldp-peer-ref-from-binding

grouping ldp-peer-ref-from-interface {
    description
        "A relative reference to an LDP peer, by the LDP ID, which
        consists of the LSR ID and the Label Space ID.";

    container peer {
```

```

description
  "Reference to an LDP peer, by the LDP ID, which consists of
  the LSR ID and the Label Space ID.";
leaf lsr-id {
  type leafref {
    path "../../../../../ldp:peers/ldp:peer/"
      + "ldp:lsr-id";
  }
  description
    "The LSR ID of the peer, as a portion of the peer LDP ID.";
}
leaf label-space-id {
  type leafref {
    path "../../../../../ldp:peers/"
      + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
      + "ldp:label-space-id";
  }
  description
    "The Label Space ID of the peer, as a portion of the peer
    LDP ID.";
}
} // peer
} // ldp-peer-ref-from-interface

grouping ldp-peer-ref-from-target {
  description
    "A relative reference to an LDP peer, by the LDP ID, which
    consists of the LSR ID and the Label Space ID.";

  container peer {
    description
      "Reference to an LDP peer, by the LDP ID, which consists of
      the LSR ID and the Label Space ID.";
    leaf lsr-id {
      type leafref {
        path "../../../../../ldp:peers/ldp:peer/"
          + "ldp:lsr-id";
      }
      description
        "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
      type leafref {
        path "../../../../../ldp:peers/"
          + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
          + "ldp:label-space-id";
      }
      description

```

```
        "The Label Space ID of the peer, as a portion of the peer
        LDP ID.";
    }
} // peer
} // ldp-peer-ref-from-target

grouping peer-attributes {
    description
        "Peer configuration attributes, used in the per-peer setting
        can in the all-peer setting.";

    leaf session-ka-holdtime {
        type uint16 {
            range 45..3600;
        }
        units seconds;
        description
            "The time interval after which an inactive LDP session
            terminates and the corresponding TCP session closes.
            Inactivity is defined as not receiving LDP packets from the
            peer.
            This leaf may be configured at the per-peer level or the
            global level, with precedence given to the value at the
            per-peer level. If the leaf is not configured at either
            level, the default value at the global level is used.";
    }
    leaf session-ka-interval {
        type uint16 {
            range 15..1200;
        }
        units seconds;
        description
            "The interval between successive transmissions of keepalive
            packets. Keepalive packets are only sent in the absence of
            other LDP packets transmitted over the LDP session.
            This leaf may be configured at the per-peer level or the
            global level, with precedence given to the value at the
            per-peer level. If the leaf is not configured at either
            level, the default value at the global level is used.";
    }
} // peer-attributes

grouping peer-authentication {
    description
        "Peer authentication container, used in the per-peer setting
        can in the all-peer setting.";

    container authentication {
```

```
description
  "Containing authentication information.";
choice authentication-type {
  description
    "Choice of authentication.";
  case password {
    leaf key {
      type string;
      description
        "This leaf specifies the authentication key. The length
        of the key may be dependent on the cryptographic
        algorithm.";
    }
    leaf crypto-algorithm {
      type identityref {
        base key-chain:crypto-algorithm;
      }
      description
        "Cryptographic algorithm associated with key.";
    }
  }
}
} // peer-authentication

grouping peer-state-derived {
  description
    "The peer state information derived from the LDP protocol
    operations.";

  container label-advertisement-mode {
    config false;
    description "Label advertisement mode state.";
    leaf local {
      type label-adv-mode;
      description
        "Local Label Advertisement Mode.";
    }
    leaf peer {
      type label-adv-mode;
      description
        "Peer Label Advertisement Mode.";
    }
    leaf negotiated {
      type label-adv-mode;
      description
        "Negotiated Label Advertisement Mode.";
    }
  }
}
```

```
    }
    leaf next-keep-alive {
        type uint16;
        units seconds;
        config false;
        description
            "Time duration from now until sending the next KeepAlive
            message.";
    }

    container received-peer-state {
        config false;
        description
            "Operational state information learned from the peer.";

        uses graceful-restart-attributes-per-peer;

        container capability {
            description "Peer capability information.";
            container end-of-lib {
                description
                    "Peer's end-of-lib capability.";
                leaf enabled {
                    type boolean;
                    description
                        "'true' if peer's end-of-lib capability is enabled.";
                }
            }
        }
        container typed-wildcard-fec {
            description
                "Peer's typed-wildcard-fec capability.";
            leaf enabled {
                type boolean;
                description
                    "'true' if peer's typed-wildcard-fec capability is
                    enabled.";
            }
        }
        container upstream-label-assignment {
            description
                "Peer's upstream label assignment capability.";
            leaf enabled {
                type boolean;
                description
                    "'true' if peer's upstream label assignment is
                    enabled.";
            }
        }
    }
}
```

```
    } // capability
  } // received-peer-state

  container session-holdtime {
    config false;
    description "Session holdtime state.";
    leaf peer {
      type uint16;
      units seconds;
      description "Peer holdtime.";
    }
    leaf negotiated {
      type uint16;
      units seconds;
      description "Negotiated holdtime.";
    }
    leaf remaining {
      type uint16;
      units seconds;
      description "Remaining holdtime.";
    }
  } // session-holdtime

  leaf session-state {
    type enumeration {
      enum non-existent {
        description "NON EXISTENT state. Transport disconnected.";
      }
      enum initialized {
        description "INITIALIZED state.";
      }
      enum openrec {
        description "OPENREC state.";
      }
      enum opensent {
        description "OPENSENT state.";
      }
      enum operational {
        description "OPERATIONAL state.";
      }
    }
    config false;
    description
      "Representing the operational status of the LDP session.";
    reference
      "RFC5036, Sec. 2.5.4.";
  }
}
```



```
container tcp-connection {
  config false;
  description "TCP connection state.";
  leaf local-address {
    type inet:ip-address;
    description "Local address.";
  }
  leaf local-port {
    type inet:port-number;
    description "Local port number.";
  }
  leaf remote-address {
    type inet:ip-address;
    description "Remote address.";
  }
  leaf remote-port {
    type inet:port-number;
    description "Remote port number.";
  }
} // tcp-connection

leaf up-time {
  type rt-types:timeticks64;
  config false;
  description
    "The number of time ticks (hundredths of a second) since the
    the state of the session with the peer changed to
    OPERATIONAL.";
}

container statistics {
  config false;
  description
    "Statistics objects.";

  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
    description
      "The time on the most recent occasion at which any one or
      more of this interface's counters suffered a
      discontinuity.  If no such discontinuities have occurred
      since the last re-initialization of the local management
      subsystem, then this node contains the time the local
      management subsystem re-initialized itself.";
  }

  container received {
```

```
        description "Inbound statistics.";
        uses statistics-peer-received-sent;
    }
    container sent {
        description "Outbound statistics.";
        uses statistics-peer-received-sent;
    }

    leaf total-addresses {
        type uint32;
        description
            "The number of learned addresses.";
    }
    leaf total-labels {
        type uint32;
        description
            "The number of learned labels.";
    }
    leaf total-fec-label-bindings {
        type uint32;
        description
            "The number of learned label-address bindings.";
    }
} // statistics
} // peer-state-derived

grouping statistics-peer-received-sent {
    description
        "Inbound and outbound statistic counters.";
    leaf total-octets {
        type yang:counter64;
        description
            "The total number of octets sent or received.";
    }
    leaf total-messages {
        type yang:counter64;
        description
            "The number of messages sent or received.";
    }
    leaf address {
        type yang:counter64;
        description
            "The number of address messages sent or received.";
    }
    leaf address-withdraw {
        type yang:counter64;
        description
            "The number of address-withdraw messages sent or received.";
```

```
    }
    leaf initialization {
        type yang:counter64;
        description
            "The number of initialization messages sent or received.";
    }
    leaf keepalive {
        type yang:counter64;
        description
            "The number of keepalive messages sent or received.";
    }
    leaf label-abort-request {
        type yang:counter64;
        description
            "The number of label-abort-request messages sent or
            received.";
    }
    leaf label-mapping {
        type yang:counter64;
        description
            "The number of label-mapping messages sent or received.";
    }
    leaf label-release {
        type yang:counter64;
        description
            "The number of label-release messages sent or received.";
    }
    leaf label-request {
        type yang:counter64;
        description
            "The number of label-request messages sent or received.";
    }
    leaf label-withdraw {
        type yang:counter64;
        description
            "The number of label-withdraw messages sent or received.";
    }
    leaf notification {
        type yang:counter64;
        description
            "The number of notification messages sent or received.";
    }
} // statistics-peer-received-sent

/*
 * Configuration data and operational state data nodes
 */
```

```
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol" {
    when "derived-from-or-self(rt:type, 'ldp:mpls-ldp')" {
      description
        "This augmentation is only valid for a control-plane
        protocol instance of LDP (type 'mpls-ldp').";
    }
    description
      "LDP augmentation to routing control-plane protocol
      configuration and state.";

    container mpls-ldp {
      must "not (../../rt:control-plane-protocol"
        + "[derived-from-or-self(rt:type, 'ldp:mpls-ldp')]"
        + "[rt:name!=current()/../../rt:name])"
      {
        description "Only one LDP instance is allowed.";
      }

      description
        "Containing configuration and operational data for the LDP
        protocol.";

      container global {
        description
          "Global attributes for LDP.";

        container capability {
          description
            "Containing the LDP capability data. The container is
            used for augmentations.";
          reference
            "RFC5036: Sec. 1.5.";
        }

        container graceful-restart {
          description
            "Attributes for graceful restart.";
          leaf enabled {
            type boolean;
            default false;
            description
              "Enable or disable graceful restart.";
          }
          leaf reconnect-time {
            type uint16 {
              range 10..1800;
            }
          }
        }
      }
    }
  }
```

```
    units seconds;
    default 120;
    description
        "Specifies the time interval that the remote LDP peer
        must wait for the local LDP peer to reconnect after
        the remote peer detects the LDP communication
        failure.";
}
leaf recovery-time {
    type uint16 {
        range 30..3600;
    }
    units seconds;
    default 120;
    description
        "Specifies the time interval, in seconds, that the
        remote LDP peer preserves its MPLS forwarding state
        after receiving the Initialization message from the
        restarted local LDP peer.";
}
leaf forwarding-holdtime {
    type uint16 {
        range 30..3600;
    }
    units seconds;
    default 180;
    description
        "Specifies the time interval, in seconds, before the
        termination of the recovery phase.";
}
} // graceful-restart

leaf lsr-id {
    type rt-types:router-id;
    description
        "Specify the value to act as the LDP LSR ID.
        If this attribute is not specified, LDP uses the router
        ID as determined by the system.";
}

container address-families {
    description
        "Per address family configuration and operational state.
        The address family can be either IPv4 or IPv6.";
    container ipv4 {
        presence
            "Present if IPv4 is enabled, unless the 'enabled'
            leaf is set to 'false'";
    }
}
```

```
description
  "Containing data related to the IPv4 address family.";

leaf enabled {
  type boolean;
  default true;
  description
    "'false' to disable the address family.";
}

leaf label-distribution-control-mode {
  type enumeration {
    enum independent {
      description
        "Independent label distribution control.";
    }
    enum ordered {
      description
        "Ordered label distribution control.";
    }
  }
  config false;
  description
    "Label distribution control mode.";
  reference
    "RFC5036: LDP Specification. Sec 2.6.";
}

// ipv4 bindings
container bindings {
  config false;
  description
    "LDP address and label binding information.";
  list address {
    key "address";
    description
      "List of address bindings learned by LDP.";
    leaf address {
      type inet:ipv4-address;
      description
        "The IPv4 address learned from an Address
        message received from or advertised to a peer.";
    }
    uses binding-address-state-attributes;
  }

  list fec-label {
    key "fec";
```

```
    description
      "List of FEC-label bindings learned by LDP.";
    leaf fec {
      type inet:ipv4-prefix;
      description
        "The prefix FEC value in the FEC-label binding,
        learned in a Label Mapping message received from
        or advertised to a peer.";
    }
    uses binding-label-state-attributes;
  }
} // bindings
} // ipv4
} // address-families
} // global

container discovery {
  description
    "Neighbor discovery configuration and operational state.";

  container interfaces {
    description
      "A list of interfaces for LDP Basic Discovery.";
    reference
      "RFC5036: LDP Specification. Sec 2.4.1.";

    uses basic-discovery-timers {
      refine "hello-holdtime" {
        default 15;
      }
      refine "hello-interval" {
        default 5;
      }
    }
  }

  list interface {
    key "name";
    description
      "List of LDP interfaces used for LDP Basic Discovery.";
    uses ldp-interface-ref;
    leaf next-hello {
      type uint16;
      units seconds;
      config false;
      description "Time to send the next Hello message.";
    }

    container address-families {
```

```
description
  "Container for address families.";
container ipv4 {
  presence
    "Present if IPv4 is enabled, unless the 'enabled'
    leaf is set to 'false'";
  description
    "IPv4 address family.";

  leaf enabled {
    type boolean;
    default true;
    description
      "Set to false to disable the address family on
      the interface.";
  }

  container hello-adjacencies {
    config false;
    description
      "Containing a list of Hello adjacencies.";

    list hello-adjacency {
      key "adjacent-address";
      config false;
      description "List of Hello adjacencies.";

      leaf adjacent-address {
        type inet:ipv4-address;
        description
          "Neighbor address of the Hello adjacency.";
      }

      uses adjacency-state-attributes;
      uses ldp-peer-ref-from-interface;
    }
  } // ipv4
} // address-families
} // interface
} // interfaces

container targeted
{
  description
    "A list of targeted neighbors for extended discovery.";

  leaf hello-holdtime {
```



```
    type uint16 {
      range 15..3600;
    }
    units seconds;
    default 45;
    description
      "The time interval for which LDP targeted Hello
      adjacency is maintained in the absence of targeted
      Hello messages from an LDP neighbor.";
  }
  leaf hello-interval {
    type uint16 {
      range 5..3600;
    }
    units seconds;
    default 15;
    description
      "The interval between consecutive LDP targeted Hello
      messages used in extended LDP discovery.";
  }

  container hello-accept {
    description
      "LDP policy to control the acceptance of extended
      neighbor discovery Hello messages.";

    leaf enabled {
      type boolean;
      default false;
      description
        "'true' to accept; 'false' to deny.";
    }
  }

  container address-families {
    description
      "Container for address families.";
    container ipv4 {
      presence
        "Present if IPv4 is enabled.";
      description
        "IPv4 address family.";

      container hello-adjacencies {
        config false;
        description
          "Containing a list of Hello adjacencies.";
      }
    }
  }
}
```

```
list hello-adjacency {
  key "local-address adjacent-address";
  description "List of Hello adjacencies.";

  leaf local-address {
    type inet:ipv4-address;
    description
      "Local address of the Hello adjacency.";
  }
  leaf adjacent-address {
    type inet:ipv4-address;
    description
      "Neighbor address of the Hello adjacency.";
  }

  uses adjacency-state-attributes;
  uses ldp-peer-ref-from-target;
}

list target {
  key "adjacent-address";
  description
    "Targeted discovery params.";

  leaf adjacent-address {
    type inet:ipv4-address;
    description
      "Configures a remote LDP neighbor for the
       extended LDP discovery.";
  }

  leaf enabled {
    type boolean;
    default true;
    description
      "'true' to enable the target.";
  }

  leaf local-address {
    type inet:ipv4-address;
    description
      "The local address used as the source address to
       send targeted Hello messages.
       If the value is not specified, the
       transport-address is used as the source
       address.";
  }
} // target
```

```
        } // ipv4
    } // address-families
} // targeted
} // discovery

container peers {
    description
        "Peers configuration attributes.";

    uses peer-authentication;
    uses peer-attributes {
        refine session-ka-holdtime {
            default 180;
        }
        refine session-ka-interval {
            default 60;
        }
    }

    list peer {
        key "lsr-id label-space-id";
        description
            "List of peers.";

        leaf lsr-id {
            type rt-types:router-id;
            description
                "The LSR ID of the peer, to identify the globally
                unique LSR. This is the first four octets of the LDP
                ID. This leaf is used together with the leaf
                'label-space-id' to form the LDP ID.";
            reference
                "RFC5036. Sec 2.2.2.";
        }
        leaf label-space-id {
            type uint16;
            description
                "The Label Space ID of the peer, to identify a specific
                label space within the LSR. This is the last two
                octets of the LDP ID. This leaf is used together with
                the leaf 'lsr-id' to form the LDP ID.";
            reference
                "RFC5036. Sec 2.2.2.";
        }
    }

    uses peer-authentication;

    container address-families {
```

```
description
  "Per-vrf per-af params.";
container ipv4 {
  presence
    "Present if IPv4 is enabled.";
  description
    "IPv4 address family.";

  container hello-adjacencies {
    config false;
    description
      "Containing a list of Hello adjacencies.";

    list hello-adjacency {
      key "local-address adjacent-address";
      description "List of Hello adjacencies.";

      leaf local-address {
        type inet:ipv4-address;
        description
          "Local address of the Hello adjacency.";
      }
      leaf adjacent-address {
        type inet:ipv4-address;
        description
          "Neighbor address of the Hello adjacency.";
      }
    }

    uses adjacency-state-attributes;

    leaf interface {
      type if:interface-ref;
      description "Interface for this adjacency.";
    }
  }
} // ipv4
} // address-families

uses peer-state-derived;
} // list peer
} // peers
} // container mpls-ldp
}

/*
 * RPCs
 */
```

```
rpc mpls-ldp-clear-peer {
  description
    "Clears the session to the peer.";
  input {
    uses ldp-peer-ref-absolute {
      description
        "The LDP peer to be cleared. If this is not provided
        then all peers are cleared.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
  }
}

rpc mpls-ldp-clear-hello-adjacency {
  description
    "Clears the hello adjacency";
  input {
    container hello-adjacency {
      description
        "Link adjacency or targetttted adjacency. If this is not
        provided then all Hello adjacencies are cleared";
      leaf protocol-name {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/rt:name";
        }
        description
          "The name of the LDP protocol instance.";
      }
      choice hello-adjacency-type {
        description "Adjacency type.";
        case targeted {
          container targeted {
            presence "Present to clear targeted adjacencies.";
            description
              "Clear targeted adjacencies.";
            leaf target-address {
              type inet:ip-address;
              description
                "The target address. If this is not provided then
                all targeted adjacencies are cleared";
            }
          }
        }
        case link {
          container link {
            presence "Present to clear link adjacencies.";
          }
        }
      }
    }
  }
}
```

```

description
  "Clear link adjacencies.";
leaf next-hop-interface {
  type leafref {
    path "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/mpls-ldp/discovery/"
      + "interfaces/interface/name";
  }
  description
    "Interface connecting to next-hop. If this is not
    provided then all link adjacencies are cleared.";
}
leaf next-hop-address {
  type inet:ip-address;
  must "../next-hop-interface" {
    description
      "Applicable when interface is specified.";
  }
  description
    "IP address of next-hop. If this is not provided
    then adjacencies to all next-hops on the given
    interface are cleared.";
}
}
} // hello-adjacency-type
} // hello-adjacency
} // input
} // mpls-ldp-clear-hello-adjacency

rpc mpls-ldp-clear-peer-statistics {
  description
    "Clears protocol statistics (e.g. sent and received
    counters).";
  input {
    uses ldp-peer-ref-absolute {
      description
        "The LDP peer whose statistics are to be cleared.
        If this is not provided then all peers' statistics are
        cleared.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
  }
}

/*
 * Notifications

```

```
*/
notification mpls-ldp-peer-event {

  description
    "Notification event for a change of LDP peer operational
    status.";
  leaf event-type {
    type oper-status-event-type;
    description "Event type.";
  }
  container peer {
    description
      "Reference to an LDP peer, by the LDP ID, which consists of
      the LSR ID and the Label Space ID.";
    uses ldp-peer-ref-absolute;
  }
}

notification mpls-ldp-hello-adjacency-event {
  description
    "Notification event for a change of LDP adjacency operational
    status.";
  leaf event-type {
    type oper-status-event-type;
    description "Event type.";
  }
  leaf protocol-name {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rt:name";
    }
    description
      "The name of the LDP protocol instance.";
  }
  choice hello-adjacency-type {
    description
      "Interface or targeted adjacency.";
    case targeted {
      container targeted {
        description
          "Targeted adjacency through LDP extended discovery.";
        leaf target-address {
          type inet:ip-address;
          description
            "The target adjacent address learned.";
        }
      }
    }
  }
}
```

```
    case link {
      container link {
        description
          "Link adjacency through LDP basic discovery.";
        leaf next-hop-interface {
          type if:interface-ref;
          description
            "The interface connecting to the adjacent next hop.";
        }
        leaf next-hop-address {
          type inet:ip-address;
          must "../next-hop-interface" {
            description
              "Applicable when interface is specified.";
          }
          description
            "IP address of the next hop. This can be IPv4 or IPv6
            address.";
        }
      }
    } // hello-adjacency-type
  } // mpls-ldp-hello-adjacency-event

notification mpls-ldp-fec-event {
  description
    "Notification event for a change of FEC status.";
  leaf event-type {
    type oper-status-event-type;
    description "Event type.";
  }
  leaf protocol-name {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rt:name";
    }
    description
      "The name of the LDP protocol instance.";
  }
  leaf fec {
    type inet:ip-prefix;
    description
      "The address prefix element of the FEC whose status
      has changed.";
  }
}
}
```


<CODE ENDS>

Figure 10: LDP base module

9.2. Extended

This YANG module imports types defined in [RFC6991], [RFC8349], [RFC8177], and [RFC8343].

```
<CODE BEGINS> file "ietf-mpls-ldp-extended@2020-02-25.yang"

// RFC Editor: replace the above date 2020-02-25 with the date of
// publication and remove this note.

module ietf-mpls-ldp-extended {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended";
  prefix "ldp-ext";

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management (NMDA
      version)";
  }

  import ietf-key-chain {
    prefix "key-chain";
    reference "RFC 8177: YANG Data Model for Key Chains";
  }

  import ietf-mpls-ldp {
    prefix "ldp";
    reference "RFC XXXX: YANG Data Model for MPLS LDP";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  }
}
```

```
import ietf-interfaces {
  prefix "if";
  reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-routing-policy {
  prefix rt-pol;
  reference
    "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
    Policy Management";
}

organization
  "IETF MPLS Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/mppls/>
  WG List:  <mailto:mppls@ietf.org>

  Editor:   Kamran Raza
            <mailto:skraza@cisco.com>

  Editor:   Rajiv Asati
            <mailto:rajiva@cisco.com>

  Editor:   Xufeng Liu
            <mailto:xufeng.liu.ietf@gmail.com>

  Editor:   Santosh Esale
            <mailto:sesale@juniper.net>

  Editor:   Xia Chen
            <mailto:jescia.chenxia@huawei.com>

  Editor:   Himanshu Shah
            <mailto:hshah@ciena.com>";

description
  "This YANG module defines the extended components for the
  management of Multi-Protocol Label Switching (MPLS) Label
  Distribution Protocol (LDP). It is also the model to
  be augmented for extended Multipoint LDP (mLDP).

  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
(<http://trustee.ietf.org/license-info>).

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

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
revision 2020-02-25 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";

    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Features
 */
feature capability-end-of-lib {
  description
    "This feature indicates that the system allows to configure
    LDP end-of-lib capability.";
}

feature capability-typed-wildcard-fec {
  description
    "This feature indicates that the system allows to configure
    LDP typed-wildcard-fec capability.";
}

feature capability-upstream-label-assignment {
  description
    "This feature indicates that the system allows to configure
    LDP upstream label assignment capability.";
}

feature forwarding-nexthop-config {
  description
    "This feature indicates that the system allows to configure
    forwarding nexthop on interfaces.";
}

feature graceful-restart-helper-mode {
```

```
    description
      "This feature indicates that the system supports graceful
      restart helper mode. We call an LSR to be operating in GR
      helper mode when it advertises 0 as its FT Reconnect Timeout
      in the FT Session TLV.
      Please refer RFC3478 section 2 for details.";
  }

  feature key-chain {
    description
      "This feature indicates that the system supports keychain for
      authentication.";
  }

  feature peers-dual-stack-transport-preference {
    description
      "This feature indicates that the system allows to configure
      the transport connection preference in a dual-stack setup
      for peers.";
  }

  feature per-interface-timer-config {
    description
      "This feature indicates that the system allows to configure
      interface Hello timers at the per-interface level.";
  }

  feature per-peer-admin-down {
    description
      "This feature indicates that the system allows to
      administratively disable a peer.";
  }

  feature per-peer-graceful-restart-config {
    description
      "This feature indicates that the system allows to configure
      graceful restart at the per-peer level.";
  }

  feature per-peer-session-attributes-config {
    description
      "This feature indicates that the system allows to configure
      session attributes at the per-peer level.";
  }

  feature policy-label-assignment-config {
    description
      "This feature indicates that the system allows to configure
```

```
        policies to assign labels according to certain prefixes.";
    }

    feature policy-ordered-label-config {
        description
            "This feature indicates that the system allows to configure
            ordered label policies.";
    }

    feature policy-targeted-discovery-config {
        description
            "This feature indicates that the system allows to configure
            policies to control the acceptance of targeted neighbor
            discovery Hello messages.";
    }

    feature session-downstream-on-demand-config {
        description
            "This feature indicates that the system allows to configure
            session downstream-on-demand";
    }

    /*
     * Typedefs
     */
    typedef neighbor-list-ref {
        type leafref {
            path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
        }
        description
            "A type for a reference to a neighbor address list.
            The string value is the name identifier for uniquely
            identifying the referenced address list, which contains a list
            of addresses that a routing policy can applied.";
        reference
            "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
            Policy Management";
    }

    typedef prefix-list-ref {
        type leafref {
            path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                + "rt-pol:prefix-sets/rt-pol:prefix-set/rt-pol:name";
        }
        description
            "A type for a reference to a prefix list.
            The string value is the name identifier for uniquely
```

```
        identifying the referenced prefix set, which contains a list
        of prefixes that a routing policy can applied.";
    reference
        "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
        Policy Management";
}

typedef peer-list-ref {
    type leafref {
        path "/rt-pol:routing-policy/rt-pol:defined-sets/"
            + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
    }
    description
        "A type for a reference to a peer address list.
        The string value is the name identifier for uniquely
        identifying the referenced address list, which contains a list
        of addresses that a routing policy can applied.";
    reference
        "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
        Policy Management";
}

/*
 * Identities
 */

/*
 * Groupings
 */
grouping address-family-ipv4-augment {
    description "Augmentation to address family IPv4.";

    uses policy-container;

    leaf transport-address {
        type inet:ipv4-address;
        description
            "The transport address advertised in LDP Hello messages.
            If this value is not specified, the LDP LSR ID is used as
            the transport address.";
        reference
            "RFC5036. Sec. 3.5.2.";
    }
}

grouping authentication-keychain-augment {
    description "Augmentation to authentication to add keychain.";
```

```
    leaf key-chain {
      type key-chain:key-chain-ref;
      description
        "key-chain name.
        If not specified, no key chain is used.";
    }
  }

  grouping capability-augment {
    description "Augmentation to capability.";

    container end-of-lib {
      if-feature capability-end-of-lib;
      description
        "Configure end-of-lib capability.";
      leaf enabled {
        type boolean;
        default false;
        description
          "'true' to enable end-of-lib capability.";
      }
    }

    container typed-wildcard-fec {
      if-feature capability-typed-wildcard-fec;
      description
        "Configure typed-wildcard-fec capability.";
      leaf enabled {
        type boolean;
        default false;
        description
          "'true' to enable typed-wildcard-fec capability.";
      }
    }

    container upstream-label-assignment {
      if-feature capability-upstream-label-assignment;
      description
        "Configure upstream label assignment capability.";
      leaf enabled {
        type boolean;
        default false;
        description
          "'true' to enable upstream label assignment.";
      }
    }
  } // capability-augment

  grouping global-augment {
    description "Augmentation to global attributes.";
```

```
leaf igp-synchronization-delay {
  type uint16 {
    range "0 | 3..300";
  }
  units seconds;
  default 0;
  description
    "Sets the interval that the LDP waits before notifying the
    Interior Gateway Protocol (IGP) that label exchange is
    completed so that IGP can start advertising the normal
    metric for the link.
    If the value is not specified, there is no delay.";
}

grouping global-forwarding-nexthop-augment {
  description
    "Augmentation to global forwarding nexthop interfaces.";

  container forwarding-nexthop {
    if-feature forwarding-nexthop-config;
    description
      "Configuration for forwarding nexthop.";

    container interfaces {
      description
        "Containing a list of interfaces on which forwarding can be
        disabled.";

      list interface {
        key "name";
        description
          "List of LDP interfaces on which forwarding can be
          disabled.";
        uses ldp:ldp-interface-ref;
        list address-family {
          key "afi";
          description
            "Per-vrf per-af params.";
          leaf afi {
            type identityref {
              base rt:address-family;
            }
            description
              "Address family type value.";
          }
          leaf ldp-disable {
            type boolean;
          }
        }
      }
    }
  }
}
```



```
        default false;
        description
            "'true' to disable LDP forwarding on the interface.";
    }
} // interface
} // interfaces
} // forwarding-nexthop
} // global-forwarding-nexthop-augment

grouping graceful-restart-augment {
    description "Augmentation to graceful restart.";

    leaf helper-enabled {
        if-feature graceful-restart-helper-mode;
        type boolean;
        default false;
        description
            "Enable or disable graceful restart helper mode.";
    }
}

grouping interface-address-family-ipv4-augment {
    description "Augmentation to interface address family IPv4.";

    leaf transport-address {
        type union {
            type enumeration {
                enum "use-global-transport-address" {
                    description
                        "Use the transport address set at the global level
                        common for all interfaces for this address family.";
                }
                enum "use-interface-address" {
                    description
                        "Use interface address as the transport address.";
                }
            }
            type inet:ipv4-address;
        }
        default "use-global-transport-address";
        description
            "IP address to be advertised as the LDP transport address.";
    }
}

grouping interface-address-family-ipv6-augment {
    description "Augmentation to interface address family IPv6.";
```

```
leaf transport-address {
  type union {
    type enumeration {
      enum "use-global-transport-address" {
        description
          "Use the transport address set at the global level
           common for all interfaces for this address family.";
      }
      enum "use-interface-address" {
        description
          "Use interface address as the transport address.";
      }
    }
    type inet:ipv6-address;
  }
  default "use-global-transport-address";
  description
    "IP address to be advertised as the LDP transport address.";
}

grouping interface-augment {
  description "Augmentation to interface.";

  uses ldp:basic-discovery-timers {
    if-feature per-interface-timer-config;
  }
  leaf igp-synchronization-delay {
    if-feature per-interface-timer-config;
    type uint16 {
      range "0 | 3..300";
    }
    units seconds;
    description
      "Sets the interval that the LDP waits before notifying the
       Interior Gateway Protocol (IGP) that label exchange is
       completed so that IGP can start advertising the normal
       metric for the link.
       This leaf may be configured at the per-interface level or
       the global level, with precedence given to the value at the
       per-interface level. If the leaf is not configured at
       either level, the default value at the global level is
       used.";
  }
}

grouping peer-af-policy-container {
  description
```

```
    "LDP policy attribute container under peer address-family.";
  container label-policy {
    description
      "Label policy attributes.";
    container advertise {
      description
        "Label advertising policies.";
      leaf prefix-list {
        type prefix-list-ref;
        description
          "Applies the prefix list to filter outgoing label
           advertisements.
           If the value is not specified, no prefix filter
           is applied.";
      }
    }
    container accept {
      description
        "Label advertisement acceptance policies.";
      leaf prefix-list {
        type prefix-list-ref;
        description
          "Applies the prefix list to filter incoming label
           advertisements.
           If the value is not specified, no prefix filter
           is applied.";
      }
    }
  }
} // peer-af-policy-container

grouping peer-augment {
  description "Augmentation to each peer list entry.";

  leaf admin-down {
    if-feature per-peer-admin-down;
    type boolean;
    default false;
    description
      "'true' to disable the peer.";
  }

  uses ldp:graceful-restart-attributes-per-peer {
    if-feature per-peer-graceful-restart-config;
  }

  uses ldp:peer-attributes {
    if-feature per-peer-session-attributes-config;
  }
}
```

```
    }  
  }  
  
  grouping peers-augment {  
    description "Augmentation to peers container.";  
  
    container session-downstream-on-demand {  
      if-feature session-downstream-on-demand-config;  
      description  
        "Session downstream-on-demand attributes.";  
      leaf enabled {  
        type boolean;  
        default false;  
        description  
          "'true' if session downstream-on-demand is enabled.";  
      }  
      leaf peer-list {  
        type peer-list-ref;  
        description  
          "The name of a peer ACL, to be applied to the  
          downstream-on-demand sessions.  
          If this value is not specified, no filter is applied to  
          any downstream-on-demand sessions.";  
      }  
    }  
  }  
  
  container dual-stack-transport-preference {  
    if-feature peers-dual-stack-transport-preference;  
    description  
      "The settings of peers to establish TCP connection in a  
      dual-stack setup.";  
    leaf max-wait {  
      type uint16 {  
        range "0..60";  
      }  
      default 30;  
      description  
        "The maximum wait time in seconds for preferred transport  
        connection establishment. 0 indicates no preference.";  
    }  
  }  
  
  container prefer-ipv4 {  
    presence  
      "Present if IPv4 is preferred for transport connection  
      establishment, subject to the 'peer-list' in this  
      container.";  
    description  
      "Uses IPv4 as the preferred address family for transport  
      connection establishment, subject to the 'peer-list' in  
      this container."
```

```
        If this container is not present, as a default, IPv6 is
        the preferred address family for transport connection
        establishment.";
    leaf peer-list {
        type peer-list-ref;
        description
            "The name of a peer ACL, to be applied to the IPv4
            transport connections.
            If this value is not specified, no filter is applied,
            and the IPv4 is preferred for all peers.";
    }
}
} // peers-augment

grouping policy-container {
    description
        "LDP policy attributes.";
    container label-policy {
        description
            "Label policy attributes.";
    }
    container advertise {
        description
            "Label advertising policies.";
    }
    container egress-explicit-null {
        description
            "Enables an egress router to advertise an
            explicit null label (value 0) in place of an
            implicit null label (value 3) to the
            penultimate hop router.";
        leaf enabled {
            type boolean;
            default false;
            description
                "'true' to enable explicit null.";
        }
    }
    leaf prefix-list {
        type prefix-list-ref;
        description
            "Applies the prefix list to filter outgoing label
            advertisements.
            If the value is not specified, no prefix filter
            is applied.";
    }
}
container accept {
    description
```

```
        "Label advertisement acceptance policies.";
    leaf prefix-list {
        type prefix-list-ref;
        description
            "Applies the prefix list to filter incoming label
             advertisements.
             If the value is not specified, no prefix filter
             is applied.";
    }
}
container assign {
    if-feature policy-label-assignment-config;
    description
        "Label assignment policies";
    container independent-mode {
        description
            "Independent label policy attributes.";
        leaf prefix-list {
            type prefix-list-ref;
            description
                "Assign labels according to certain prefixes.
                 If the value is not specified, no prefix filter
                 is applied (labels are assigned to all learned
                 routes).";
        }
    }
}
container ordered-mode {
    if-feature policy-ordered-label-config;
    description
        "Ordered label policy attributes.";
    leaf egress-prefix-list {
        type prefix-list-ref;
        description
            "Assign labels according to certain prefixes for
             egress LSR.";
    }
}
} // assign
} // label-policy
} // policy-container

/*
 * Configuration and state data nodes
 */
// Forwarding nexthop augmentation to the global tree
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global" {
```

```
    description "Forwarding nexthop augmentation.";
    uses global-forwarding-nexthop-augment;
}

// global/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
+ "ldp:address-families" {
    description "Global IPv6 augmentation.";

    container ipv6 {
        presence
            "Present if IPv6 is enabled, unless the 'enabled'
            leaf is set to 'false'";
        description
            "Containing data related to the IPv6 address family.";

        leaf enabled {
            type boolean;
            default true;
            description
                "'false' to disable the address family.";
        }

        uses policy-container;

        leaf transport-address {
            type inet:ipv6-address;
            mandatory true;
            description
                "The transport address advertised in LDP Hello messages.";
        }

        leaf label-distribution-control-mode {
            type enumeration {
                enum independent {
                    description
                        "Independent label distribution control.";
                }
                enum ordered {
                    description
                        "Ordered label distribution control.";
                }
            }
            config false;
            description
                "Label distribution control mode.";
            reference
```

```
    "RFC5036: LDP Specification. Sec 2.6.";
  }

  // ipv6 bindings
  container bindings {
    config false;
    description
      "LDP address and label binding information.";
    list address {
      key "address";
      description
        "List of address bindings learned by LDP.";
      leaf address {
        type inet:ipv6-address;
        description
          "The IPv6 address learned from an Address
            message received from or advertised to a peer.";
      }
      uses ldp:binding-address-state-attributes;
    }

    list fec-label {
      key "fec";
      description
        "List of FEC-label bindings learned by LDP.";
      leaf fec {
        type inet:ipv6-prefix;
        description
          "The prefix FEC value in the FEC-label binding,
            learned in a Label Mapping message received from
            or advertised to a peer.";
      }
      uses ldp:binding-label-state-attributes;
    }
  } // bindings
} // ipv6

// discovery/interfaces/interface/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:interfaces/ldp:interface/"
+ "ldp:address-families" {
  description "Interface IPv6 augmentation.";

  container ipv6 {
    presence
      "Present if IPv6 is enabled, unless the 'enabled'
```



```
        leaf is set to 'false'";
    description
        "IPv6 address family.";

    leaf enabled {
        type boolean;
        default true;
        description
            "'false' to disable the address family on the interface.";
    }

    container hello-adjacencies {
        config false;
        description
            "Containing a list of Hello adjacencies.";

        list hello-adjacency {
            key "adjacent-address";
            config false;
            description "List of Hello adjacencies.";

            leaf adjacent-address {
                type inet:ipv6-address;
                description
                    "Neighbor address of the Hello adjacency.";
            }

            uses ldp:adjacency-state-attributes;
            uses ldp:ldp-peer-ref-from-interface;
        }
    }
} // ipv6
}

// discovery/targeted/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:targeted/ldp:address-families" {
    description "Targeted discovery IPv6 augmentation.";

    container ipv6 {
        presence
            "Present if IPv6 is enabled.";
        description
            "IPv6 address family.";

        container hello-adjacencies {
            config false;
        }
    }
}
```

```
description
  "Containing a list of Hello adjacencies.";

list hello-adjacency {
  key "local-address adjacent-address";
  config false;
  description "List of Hello adjacencies.";

  leaf local-address {
    type inet:ipv6-address;
    description
      "Local address of the Hello adjacency.";
  }
  leaf adjacent-address {
    type inet:ipv6-address;
    description
      "Neighbor address of the Hello adjacency.";
  }

  uses ldp:adjacency-state-attributes;
  uses ldp:ldp-peer-ref-from-target;
}

list target {
  key "adjacent-address";
  description
    "Targeted discovery params.";

  leaf adjacent-address {
    type inet:ipv6-address;
    description
      "Configures a remote LDP neighbor for the
       extended LDP discovery.";
  }
  leaf enabled {
    type boolean;
    default true;
    description
      "'true' to enable the target.";
  }
  leaf local-address {
    type inet:ipv6-address;
    description
      "The local address used as the source address to send
       targeted Hello messages.
       If the value is not specified, the transport-address
       is used as the source address.";
  }
}
```

```
    }
  } // target
} // ipv6
}

// /peers/peer/state/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/"
+ "ldp:peer/ldp:address-families" {
  description "Peer state IPv6 augmentation.";

  container ipv6 {
    presence
      "Present if IPv6 is enabled.";
    description
      "IPv6 address family.";

    container hello-adjacencies {
      config false;
      description
        "Containing a list of Hello adjacencies.";

      list hello-adjacency {
        key "local-address adjacent-address";
        description "List of Hello adjacencies.";

        leaf local-address {
          type inet:ipv6-address;
          description
            "Local address of the Hello adjacency.";
        }
        leaf adjacent-address {
          type inet:ipv6-address;
          description
            "Neighbor address of the Hello adjacency.";
        }
      }

      uses ldp:adjacency-state-attributes;

      leaf interface {
        type if:interface-ref;
        description "Interface for this adjacency.";
      }
    }
  }
} // ipv6
}
```

```
/*
 * Configuration data and operational state data nodes
 */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global" {
  description "Graceful restart augmentation.";
  uses global-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
+ "ldp:capability" {
  description "Capability augmentation.";
  uses capability-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
+ "ldp:graceful-restart" {
  description "Graceful restart augmentation.";
  uses graceful-restart-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
+ "ldp:address-families/ldp:ipv4" {
  description "Address family IPv4 augmentation.";
  uses address-family-ipv4-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:interfaces/ldp:interface" {
  description "Interface augmentation.";
  uses interface-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:interfaces/ldp:interface/ldp:address-families/"
+ "ldp:ipv4" {
  description "Interface address family IPv4 augmentation.";
  uses interface-address-family-ipv4-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:interfaces/ldp:interface/ldp:address-families/"
```

```
+ "ldp-ext:ipv6" {
  description "Interface address family IPv6 augmentation.";
  uses interface-address-family-ipv6-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:targeted/ldp:hello-accept" {
  description "Targeted discovery augmentation.";
  leaf neighbor-list {
    if-feature policy-targeted-discovery-config;
    type neighbor-list-ref;
    description
      "The name of a neighbor ACL, to accept Hello messages from
       LDP peers as permitted by the neighbor-list policy.
       If this value is not specified, targeted Hello messages from
       any source are accepted.";
  }
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers" {
  description "Peers augmentation.";
  uses peers-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/"
+ "ldp:authentication/ldp:authentication-type" {
  if-feature key-chain;
  description "Peers authentication augmentation.";
  case key-chain {
    uses authentication-keychain-augment;
  }
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/ldp:peer" {
  description "Peer list entry augmentation.";
  uses peer-augment;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/ldp:peer/"
+ "ldp:authentication/ldp:authentication-type" {
  if-feature key-chain;
  description "Peer list entry authentication augmentation.";
  case key-chain {
```

```
        uses authentication-keychain-augment;
    }
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/ldp:peer/"
+ "ldp:address-families/ldp:ipv4" {
    description
        "Peer list entry IPv4 augmentation.";
    uses peer-af-policy-container;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/ldp:peer/"
+ "ldp:address-families/ldp-ext:ipv6" {
    description
        "Peer list entry IPv6 augmentation.";
    uses peer-af-policy-container;
}
}

<CODE ENDS>
```

Figure 11: LDP extended module

10. Security Considerations

This specification inherits the security considerations captured in [RFC5920] and the LDP protocol specification documents, namely base LDP [RFC5036], LDP IPv6 [RFC7552], LDP Capabilities [RFC5561], Typed Wildcard FEC [RFC5918], LDP End-of-LIB [RFC5919], and LDP Upstream Label Assignment [RFC6389].

10.1. YANG model

The YANG modules 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.

10.1.1. Writable nodes

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.

For LDP, the ability to modify MPLS LDP configuration may allow the entire MPLS LDP domain to be compromised including forming LDP adjacencies and/or peer sessions with unauthorized routers to mount a massive Denial-of-Service (DoS) attack. In particular, following are the subtrees and data nodes that are sensitive and vulnerable:

- * /mpls-ldp/discovery/interfaces/interface: Adding LDP on any unprotected interface could allow an LDP hello adjacency to be formed with an unauthorized and malicious neighbor. Once an hello adjacency is formed, a peer session could progress with this neighbor.
- * /mpls-ldp/discovery/targeted/hello-accept: Allowing acceptance of targeted-hellos could open LDP to DoS attacks related to incoming targeted hellos from malicious sources.
- * /mpls-ldp/peers/authentication: Allowing a peer session establishment is typically controlled via LDP authentication where a proper and secure authentication password/key management is warranted.
- * /mpls-ldp/peers/peer/authentication: Same as above.

10.1.2. Readable nodes

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:

The exposure of LDP databases (such as hello adjacencies, peers, address bindings, and fec-label bindings) beyond the scope of the LDP admin domain may be undesirable. The relevant subtrees and data nodes are as follows:

- * /mpls-ldp/global/address-families/ipv4/bindings/address
- * /mpls-ldp/global/address-families/ipv6/bindings/address
- * /mpls-ldp/global/address-families/ipv4/bindings/fec-label
- * /mpls-ldp/global/address-families/ipv6/bindings/fec-label
- * /mpls-ldp/discovery/interfaces/interface/address-families/ipv4/hello-adjacencies
- * /mpls-ldp/discovery/interfaces/interface/address-families/ipv6/hello-adjacencies
- * /mpls-ldp/discovery/targeted/address-families/ipv4/hello-adjacencies
- * /mpls-ldp/discovery/targeted/address-families/ipv6/hello-adjacencies
- * /mpls-ldp/peers

The configuration for LDP peer authentication is supported via the specification of key-chain [RFC8040], or via direct specification of a key associated with a crypto algorithm (such as MD5). The relevant subtrees and data nodes are as follows:

- * /mpls-ldp/peers/authentication
- * /mpls-ldp/peers/peer/authentication

The actual authentication key data (whether locally specified or part of a key-chain) is sensitive and needs to be kept secret from unauthorized parties. For key-chain based authentication, this model inherits the security considerations of [RFC8040] (that includes the considerations with respect to the local storage and handling of authentication keys). A similar procedure for storage and access to direct key is warranted.

10.1.3. RPC operations

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations otherwise control plane flaps, network outages, and DoS attacks are possible. The RPC operations are:

- * mpls-ldp-clear-peer

* mpls-ldp-clear-hello-adjacency

10.1.4. Notifications

The model describes several notifications. The implementations must rate-limit the generation of these notifications to avoid creating significant notification load and possible side effects on the system stability.

11. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

URI	Registrant	XML
urn:ietf:params:xml:ns:yang:ietf-mpls-ldp	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended	The IESG	N/A

Table 1: URIs

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

Name	Namespace	Prefix	Reference
ietf-mpls-ldp	urn:ietf:params:xml:ns:yang: :ietf-mpls-ldp	ldp	This document
ietf-mpls-ldp-extended	urn:ietf:params:xml:ns:yang: :ietf-mpls-ldp-extended	ldp- ext	This document

Table 2: YANG Modules

-- RFC Editor: Replace "this document" with the document RFC number at time of publication, and remove this note.

12. Acknowledgments

The authors would like to acknowledge Eddie Chami, Nagendra Kumar, Mannan Venkatesan, and Pavan Beeram for their contribution to this document.

We also acknowledge Ladislav Lhotka, Jan Lindblad, Tom Petch, Yingzhen Qu, and Benjamin Kaduk for their detailed review of the model during WG and IESG.

13. Contributors

Danial Johari
Cisco Systems
Email: dajohari@cisco.com

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Jeff Tantsura
Apstra
Email: jefftant.ietf@gmail.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com

Reshad Rahman
Cisco Systems
Email: rrahman@cisco.com

Stephane Litkowski
Cisco Systems
Email: slitkows@cisco.com

14. Normative References

- [I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", Work in Progress, Internet-Draft, draft-ietf-rtgwg-policy-model-09, 4 March 2020, <<https://tools.ietf.org/html/draft-ietf-rtgwg-policy-model-09>>.
- [RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <<https://www.rfc-editor.org/info/rfc3478>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,

- DOI 10.17487/RFC3688, January 2004,
<<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
"LDP Specification", RFC 5036, DOI 10.17487/RFC5036,
October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream
Label Assignment and Context-Specific Label Space",
RFC 5331, DOI 10.17487/RFC5331, August 2008,
<<https://www.rfc-editor.org/info/rfc5331>>.
- [RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP
Synchronization", RFC 5443, DOI 10.17487/RFC5443, March
2009, <<https://www.rfc-editor.org/info/rfc5443>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL.
Le Roux, "LDP Capabilities", RFC 5561,
DOI 10.17487/RFC5561, July 2009,
<<https://www.rfc-editor.org/info/rfc5561>>.
- [RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution
Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class
(FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010,
<<https://www.rfc-editor.org/info/rfc5918>>.
- [RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas,
"Signaling LDP Label Advertisement Completion", RFC 5919,
DOI 10.17487/RFC5919, August 2010,
<<https://www.rfc-editor.org/info/rfc5919>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS
Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010,
<<https://www.rfc-editor.org/info/rfc5920>>.
- [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>>.

- [RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <<https://www.rfc-editor.org/info/rfc6389>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<https://www.rfc-editor.org/info/rfc7277>>.
- [RFC7552] Asati, R., Pignataro, C., Raza, K., Manral, V., and R. Papneja, "Updates to LDP for IPv6", RFC 7552, DOI 10.17487/RFC7552, June 2015, <<https://www.rfc-editor.org/info/rfc7552>>.
- [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>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <<https://www.rfc-editor.org/info/rfc8177>>.
- [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>>.
- [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>>.

- [RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <<https://www.rfc-editor.org/info/rfc8344>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [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>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.

15. Informative References

- [I-D.ietf-mpls-mldp-yang]
Raza, K., Liu, X., Esale, S., Andersson, L., Tantsura, J., and S. Krishnaswamy, "YANG Data Model for MPLS mLDP", Work in Progress, Internet-Draft, draft-ietf-mpls-mldp-yang-06, 31 May 2019, <<https://tools.ietf.org/html/draft-ietf-mpls-mldp-yang-06>>.
- [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>>.
- [RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <<https://www.rfc-editor.org/info/rfc7307>>.
- [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>>.
- [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. Data Tree Example

This section contains an example of an instance data tree in the JSON encoding [RFC7951], containing both configuration and state data.

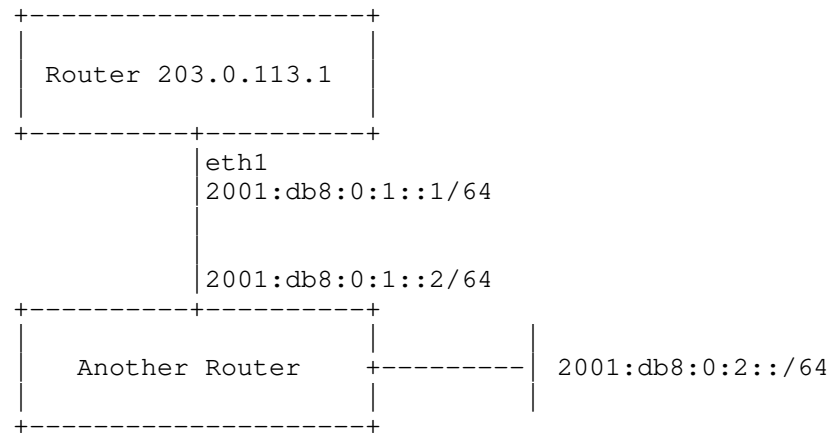


Figure 12: Example topology

The configuration instance data tree for Router 203.0.113.1 in the above figure could be as follows:

```

{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with LDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "ietf-ip:ipv6": {
          "address": [
            {
              "ip": "2001:db8:0:1::1",
              "prefix-length": 64
            }
          ]
        },
        "forwarding": true
      }
    ]
  },
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "control-plane-protocols": {

```

```

"control-plane-protocol": [
  {
    "type": "ietf-mpls-ldp:mpls-ldp",
    "name": "ldp-1",
    "ietf-mpls-ldp:mpls-ldp": {
      "global": {
        "address-families": {
          "ietf-mpls-ldp-extended:ipv6": {
            "enabled": true,
            "transport-address": "2001:db8:0:1::1"
          }
        }
      },
      "discovery": {
        "interfaces": {
          "interface": [
            {
              "name": "eth1",
              "address-families": {
                "ietf-mpls-ldp-extended:ipv6": {
                  "enabled": true
                }
              }
            }
          ]
        }
      }
    }
  }
]
}

```

Figure 13: Example Configuration data in JSON

The corresponding operational state data for Router 203.0.113.1 could be as follows:

```

{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with LDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "phys-address": "00:00:5e:00:53:01",

```

```
    "oper-status": "up",
    "statistics": {
      "discontinuity-time": "2018-09-10T15:16:27-05:00"
    },
    "ietf-ip:ipv6": {
      "forwarding": true,
      "mtu": 1500,
      "address": [
        {
          "ip": "2001:db8:0:1::1",
          "prefix-length": 64,
          "origin": "static",
          "status": "preferred"
        },
        {
          "ip": "fe80::200:5eff:fe00:5301",
          "prefix-length": 64,
          "origin": "link-layer",
          "status": "preferred"
        }
      ],
      "neighbor": [
        {
          "ip": "2001:db8:0:1::2",
          "link-layer-address": "00:00:5e:00:53:02",
          "origin": "dynamic",
          "is-router": [null],
          "state": "reachable"
        },
        {
          "ip": "fe80::200:5eff:fe00:5302",
          "link-layer-address": "00:00:5e:00:53:02",
          "origin": "dynamic",
          "is-router": [null],
          "state": "reachable"
        }
      ]
    }
  ],
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "interfaces": {
      "interface": [
        "eth1"
      ]
    }
  },
}
```



```

"control-plane-protocols": {
  "control-plane-protocol": [
    {
      "type": "ietf-mpls-ldp:mpls-ldp",
      "name": "ldp-1",
      "ietf-mpls-ldp:mpls-ldp": {
        "global": {
          "address-families": {
            "ietf-mpls-ldp-extended:ipv6": {
              "enabled": true,
              "transport-address": "2001:db8:0:1::1"
            }
          }
        },
        "discovery": {
          "interfaces": {
            "interface": [
              {
                "name": "eth1",
                "address-families": {
                  "ietf-mpls-ldp-extended:ipv6": {
                    "enabled": true,
                    "hello-adjacencies": {
                      "hello-adjacency": [
                        {
                          "adjacent-address":
                            "fe80::200:5eff:fe00:5302",
                          "flag": ["adjacency-flag-active"],
                          "hello-holdtime": {
                            "adjacent": 15,
                            "negotiated": 15,
                            "remaining": 9
                          },
                          "next-hello": 3,
                          "statistics": {
                            "discontinuity-time":
                              "2018-09-10T15:16:27-05:00"
                          },
                          "peer": {
                            "lsr-id": "203.0.113.2",
                            "label-space-id": 0
                          }
                        }
                      ]
                    }
                  }
                }
              ]
            }
          }
        }
      }
    }
  ]
}

```

```
]
}
},
"peers": {
  "peer": [
    {
      "lsr-id": "203.0.113.2",
      "label-space-id": 0,
      "label-advertisement-mode": {
        "local": "downstream-unsolicited",
        "peer": "downstream-unsolicited",
        "negotiated": "downstream-unsolicited"
      },
      "next-keep-alive": 5,
      "session-holdtime": {
        "peer": 180,
        "negotiated": 180,
        "remaining": 78
      },
      "session-state": "operational",
      "tcp-connection": {
        "local-address": "fe80::200:5eff:fe00:5301",
        "local-port": 646,
        "remote-address": "fe80::200:5eff:fe00:5302",
        "remote-port": 646
      },
      "up-time": 3438100,
      "statistics": {
        "discontinuity-time": "2018-09-10T15:16:27-05:00"
      }
    ]
  }
}
}
```

Figure 14: Example Operational data in JSON

Authors' Addresses

Kamran Raza (editor)
Cisco Systems

Canada
Email: skraza@cisco.com

Rajiv Asati
Cisco Systems
United States of America
Email: rajiva@cisco.com

Xufeng Liu
Volta Networks
United States of America
Email: xufeng.liu.ietf@gmail.com

Santosh Esale
Juniper Networks
United States of America
Email: sesale@juniper.net

Xia Chen
Huawei Technologies
China
Email: jescia.chenxia@huawei.com

Himanshu Shah
Ciena Corporation
United States of America
Email: hshah@ciena.com

MPLS Working Group
Internet-Draft
Intended status: Informational
Expires: September 8, 2019

L. Andersson
Bronze Dragon Consulting
S. Bryant
A. Malis
Huawei Technologies
N. Leymann
Deutsche Telekom
G. Swallow
Independent
March 7, 2019

Deprecating MD5 for LDP
draft-nslag-mpls-deprecate-md5-04

Abstract

When the MPLS Label Distribution Protocol (LDP) was specified circa 1999, there were very strong requirements that LDP should use a cryptographic hash function to sign LDP protocol messages. MD5 was widely used at that time, and was the obvious choices.

However, even when this decision was being taken there were concerns as to whether MD5 was a strong enough signing option. This discussion was briefly reflected in section 5.1 of RFC 5036 [RFC5036] (and also in RFC 3036 [RFC3036]).

Over time it has been shown that MD5 can be compromised. Thus, there is a concern shared in the security community and the working groups responsible for the development of the LDP protocol that LDP is no longer adequately secured.

This document deprecates MD5 as the signing method for LDP messages. The document also selects a future method to secure LDP messages - the choice is TCP-AO. In addition, we specify that the TBD cryptographic mechanism is to be the default TCP-AO security method.

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 September 8, 2019.

Copyright Notice

Copyright (c) 2019 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
1.1. Requirement Language	3
2. Background	3
2.1. LDP in RFC 5036	3
2.2. MD5 in BGP	3
2.3. Prior Art	4
3. Securing LDP	4
4. Security Considerations	5
5. IANA Considerations	5
6. Acknowledgements	5
7. References	5
7.1. Normative References	5
7.2. Informative References	6
Authors' Addresses	6

1. Introduction

RFC 3036 was published in January 2001 as a Proposed Standard, and it was replaced by RFC 5035, which is a Draft Standard, in October 2007. Two decades after LDP was originally specified there is a concern shared by the security community and the IETF working groups that develop the LDP protocol that LDP is no longer adequately secured.

LDP currently uses MD5 to cryptographically sign its messages for security security purposes. However, MD5 is a hash function that is no longer considered adequate to meet current security requirements.

1.1. Requirement Language

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. Background

2.1. LDP in RFC 5036

In Section 5.1 "Spoofing" of RFC 5036 [RFC5036], in list item 2 "Session communication carried by TCP" the following statements are made:

LDP specifies use of the TCP MD5 Signature Option to provide for the authenticity and integrity of session messages.

RFC 2385 [RFC2385] asserts that MD5 authentication is now considered by some to be too weak for this application. It also points out that a similar TCP option with a stronger hashing algorithm (it cites SHA-1 as an example) could be deployed. To our knowledge, no such TCP option has been defined and deployed. However, we note that LDP can use whatever TCP message digest techniques are available, and when one stronger than MD5 is specified and implemented, upgrading LDP to use it would be relatively straightforward.

2.2. MD5 in BGP

There has been a similar discussion among working groups developing the BGP protocol. BGP has already replaced MD5 with TCP-AO. This was specified in RFC 7454 [RFC7454].

To secure LDP the same approach will be followed, TCP-AO will be used for LDP also.

As far as we are able to ascertain, there is currently no recommended, mandatory to implement, cryptographic function specified. We are concerned that without such a mandatory function, implementations will simply fall back to MD5 and nothing will really be changed. The MPLS working group will need the expertise of the

security community to specify a viable security function that is suitable for wide scale deployment on existing network platforms.

2.3. Prior Art

RFC 6952 [RFC6952] discusses a set of routing protocols that all are using TCP for transport of protocol messages, according to guidelines set forth in Section 4.2 of "Keying and Authentication for Routing Protocols Design Guidelines", RFC 6518 [RFC6518].

RFC 6952 takes a much broader approach than this document, it discusses several protocols and also securing the LDP session initialization. This document has a narrower scope, securing LDP session messages only. LDP in initialization mode is addressed in RFC 7349 [RFC7349].

RFC 6952 and this document, basically suggest the same thing, move to TCP-AO and deploy a strong cryptographic algorithm.

All the protocols discussed in RFC 6952 should adopt the approach to securing protocol messages over TCP.

3. Securing LDP

Implementations conforming to this RFC MUST implement TCP-AO to secure the TCP sessions carrying LDP in addition to the currently required TCP MD5 Signature Option.

A TBD cryptographic mechanism must be implemented and provided to TCP-AO to secure LDP messages.

The TBD mechanism is the preferred option, and MD5 SHOULD only be used when TBD is unavailable.

Note: The authors are not experts on this part of the stack, but it seems that TCP security negotiation is still work in progress. If we are wrong, then we need to include a requirement that such negotiation is also required. In the absence of a negotiation protocol, however, we need to leave this as a configuration process until such time as the negotiation protocol work is complete. On completion of a suitable negotiation protocol we need to issue a further update requiring its use.

Cryptographic mechanisms do not have an indefinite lifetime, the IETF hence anticipates updating default cryptographic mechanisms over time.

The TBD default security function will need to be chosen such that it can reasonably be implemented on a typical router route processor, and which will provide adequate security without significantly degrading the convergence time of a Label Switching Router (LSR).

Without a function that does not significantly impact router convergence we simply close one vulnerability and open another.

Note: As experts on the LDP protocol, but not on security mechanisms, we need to ask the security area for a review of our proposed approach, and help correcting any misunderstanding of the security issues or our misunderstanding of the existing security mechanisms. We also need a recommendation on a suitable security function (TBD in the above text).

4. Security Considerations

This document is entirely about LDP operational security. It describes best practices that one should adopt to secure LDP messages and the TCP based LDP sessions between LSRs.

This document does not aim to describe existing LDP implementations, their potential vulnerabilities, or ways they handle errors. It does not detail how protection could be enforced against attack techniques using crafted packets.

5. IANA Considerations

There are no requests for IANA actions in this document.

Note to the RFC Editor - this section can be removed before publication.

6. Acknowledgements

-

-

7. References

7.1. 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>>.

- [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <<https://www.rfc-editor.org/info/rfc2385>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [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>>.

7.2. Informative References

- [RFC3036] Andersson, L., Doolan, P., Feldman, N., Fredette, A., and B. Thomas, "LDP Specification", RFC 3036, DOI 10.17487/RFC3036, January 2001, <<https://www.rfc-editor.org/info/rfc3036>>.
- [RFC6518] Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", RFC 6518, DOI 10.17487/RFC6518, February 2012, <<https://www.rfc-editor.org/info/rfc6518>>.
- [RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <<https://www.rfc-editor.org/info/rfc6952>>.
- [RFC7349] Zheng, L., Chen, M., and M. Bhatia, "LDP Hello Cryptographic Authentication", RFC 7349, DOI 10.17487/RFC7349, August 2014, <<https://www.rfc-editor.org/info/rfc7349>>.
- [RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <<https://www.rfc-editor.org/info/rfc7454>>.

Authors' Addresses

Loa Andersson
Bronze Dragon Consulting

Email: loa@pi.nu

Stewart Bryant
Huawei Technologies

Email: stewart.bryant@gmail.com

Andrew G. Malis
Huawei Technologies

Email: agmalis@gmail.com

Nicolai Leymann
Deutsche Telekom

Email: N.Leymann@telekom.de

George Swallow
Independent

Email: swallow.ietf@gmail.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

IJ. Wijnands
K. Raza
Cisco Systems, Inc.
Z. Zhang
Juniper Networks
A. Gulko
Thomson Reuters
March 5, 2018

mLDP Extensions for Multi-Topology Routing
draft-wijnands-mpls-mldp-multi-topology-00.txt

Abstract

Multi-Topology Routing (MTR) is a technology to enable service differentiation within an IP network. Flexible Algorithm (FA) is another mechanism of creating a sub-topology within a topology using defined topology constraints and computation algorithm. In order to deploy mLDP in a network that supports MTR and/or FA, mLDP is required to become topology and FA aware. This document specifies extensions to mLDP to support MTR with FA such that when building a Multi-Point LSPs it can follow a particular topology and algorithm.

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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 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. Glossary	2
2. Introduction	3
3. Specification of Requirements	4
4. MT Scoped mLDP FECs	4
4.1. MP FEC Extensions for MT	4
4.1.1. MP FEC Element	4
4.1.2. MT IP Address Families	5
4.1.3. MT MP FEC Element	6
4.2. Topology IDs	7
5. MT Multipoint Capability	7
6. MT Applicability on FEC-based features	8
6.1. Typed Wildcard MP FEC Elements	8
6.2. End-of-LIB	9
7. Topology-Scoped Signaling and Forwarding	10
7.1. Upstream LSR selection	10
7.2. Downstream forwarding interface selection	10
8. LSP Ping Extensions	10
9. Security Considerations	11
10. IANA Considerations	11
11. Acknowledgments	12
12. References	12
12.1. Normative References	12
12.2. Informative References	13
Authors' Addresses	13

1. Glossary

MT - Multi-Topology

MT-ID - Multi-Topology Identifier

MTR - Multi-Topology Routing

IGP - Interior Gateway Protocol

MP - Multipoint (P2MP or MP2MP)

LDP - Label Distribution Protocol

mLDP - Multipoint LDP

P2MP - Point-to-Multipoint

MP2MP - Multipoint-to-Multipoint

FEC - Forwarding Equivalence Class

LSP - Label Switched Path

FA - Flexible Algorithm

IPA - IGP Algorithm

2. Introduction

Multi-Topology Routing (MTR) is a technology to enable service differentiation within an IP network. IGP protocols (OSPF and IS-IS) and LDP have already been extended to support MTR. To support MTR, an IGP maintains independent IP topologies, termed as "Multi-Topologies" (MT), and computes/install routes per topology. OSPF extensions [RFC4915] and ISIS extensions [RFC5120] specify the MT extensions under respective IGPs. To support IGP MT, similar LDP extensions [RFC7307] have been specified to make LDP MT-aware and be able to setup unicast Label Switched Paths (LSPs) along IGP MT routing paths.

A more light weight mechanism to define constraint-based topologies is Flexible Algorithm (FA) [I-D.hegdeppsenak-isis-sr-flex-algo]. FA can be seen as creating a sub-topology within a topology using defined topology constraints and computation algorithm. This can be done within a MTR topology or just the default Topology. An instance of such a sub-topology is identified by a 1 octet value as documented in [I-D.hegdeppsenak-isis-sr-flex-algo]. Flexible Algorithm is a mechanism to create a sub-topology, but in the future different algorithms might be defined on how to achieve that. For that reason, in the remainder of this document we'll refer to this as the IGP Algorithm (IPA).

Multipoint LDP (mLDP) refers to extensions in LDP to setup multipoint LSPs (point-to-multipoint (P2MP) or multipoint-to-multipoint (MP2MP)), by means of set of extensions and procedures defined in [RFC6388]. In order to deploy mLDP in a network that supports MTR and FA, mLDP is required to become topology and algorithm aware. This document specifies extensions to mLDP to support MTR/IPA such that when building a Multi-Point LSPs it can follow a particular topology and algorithm. This means that the identifier for the

particular Topology to be used by mLDP have to become a two tuple (MTR Topology Id, IGP Algorithm).

3. Specification of Requirements

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

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

4. MT Scoped mLDP FECs

As defined in [RFC7307], MPLS Multi-Topology Identifier (MT-ID) is an identifier that is used to associate an LSP with a certain MTR topology. In the context of MP LSPs, this identifier is part of the mLDP FEC encoding so that LDP peers are able to setup an MP LSP via their own defined MTR policy. In order to avoid conflicting MTR policies for the same mLDP FEC, the MT-ID needs to be a part of the FEC, so that different MT-ID values will result in unique MP-LSP FEC elements.

The same applies to the IPA. The IPA needs to be encoded as part of the mLDP FEC to create unique MP-LSPs and at the same time is used to signal to mLDP (hop-by-hop) which Algorithm needs to be used to create the MP-LSP.

Since the MT-ID and IPA are part of the FEC, they apply to all the LDP messages that potentially include an mLDP FEC element.

4.1. MP FEC Extensions for MT

Following subsections propose the extensions to bind an mLDP FEC to a topology. The mLDP MT extensions reuse some of the extensions specified in [RFC7307].

4.1.1. MP FEC Element

Base mLDP specification [RFC6388] defines MP FEC Element as follows:

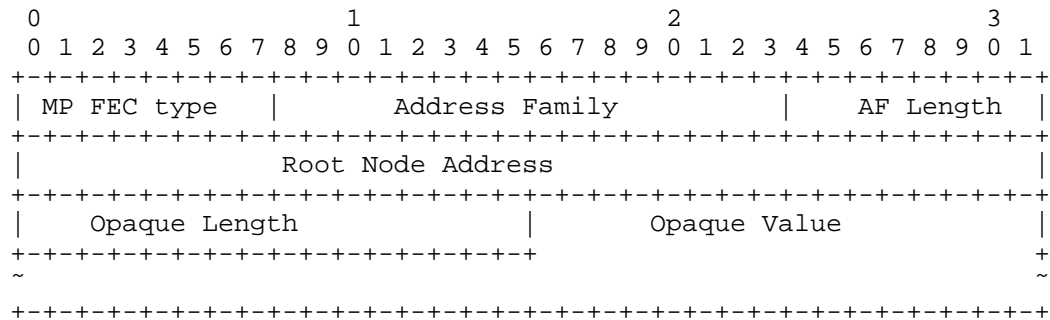


Figure 1: MP FEC Element Format [RFC6388]

Where "Root Node Address" encoding is as defined for given "Address Family", and whose length (in octets) is specified by the "AF Length" field.

To extend MP FEC elements for MT, the {MT-ID, IPA} is a tuple that is relevant in the context of the root address of the MP LSP. The {MT-ID, IPA} tuple determines in which (sub)-topology the root address needs to be resolved. Since the {MT-ID, IPA} tuple should be considered part of the mLDP FEC, the most natural place to encode this tuple is as part of the root address. While encoding it, we also propose to use "MT IP" Address Families as described in following sub section.

4.1.2. MT IP Address Families

[RFC7307] has specified new address families, named "MT IP" and "MT IPv6", to allow specification of an IP prefix within a topology scope. In addition to using this address family for mLDP, we also use 8 bits of the 16 bits Reserved field to encode the IGP Algorithm (IPA) Registry. The resulting format of the data associated with these new Address Families is as follows:

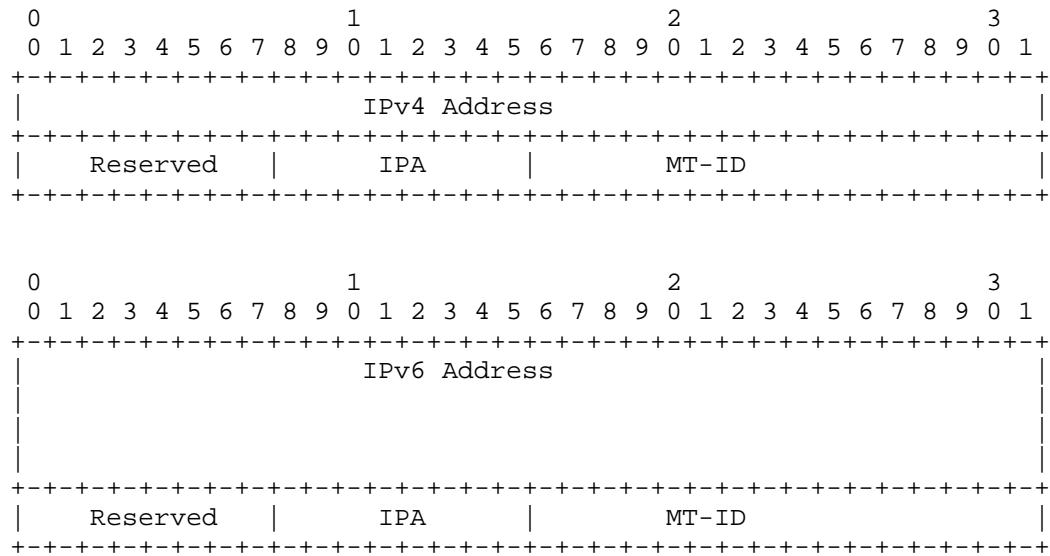


Figure 2: Modified MT IP Address Families Data Format

Where:

IPv4/IPv6 Address: An IP address corresponding to "MT IP" and "MT IPv6" address families respectively.

IPA: The IGP Algorithm, values are from the IGP Algorithm registry.

Reserved: This 8-bit field MUST be zero. If a message is received that includes a MT address family where the 8 bit Reserved value is not zero, the message must be discarded.

4.1.3. MT MP FEC Element

By using extended MT IP Address Family, the resultant MT MP FEC element is to be encoded as follows:

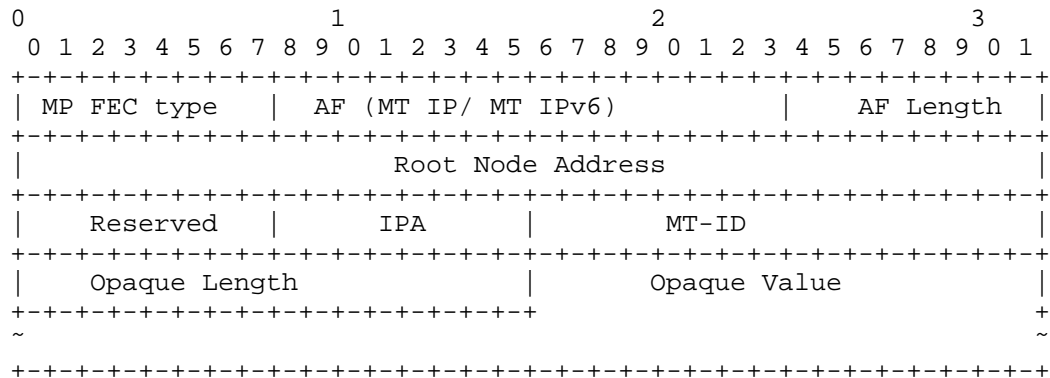


Figure 3: IP MT-Scoped MP FEC Element Format

In the context of this document, the applicable LDP FECs for MT mLDP include:

- o MP FEC Elements:
 - * P2MP (type 0x6)
 - * MP2MP-up (type 0x7)
 - * MP2MP-down (type 0x8)
- o Typed Wildcard FEC Element (type 0x5)

In case of "Typed Wildcard FEC Element", the sub FEC Element type MUST be one of the MP FECs listed above.

This specification allows the use of Topology-scoped mLDP FECs in LDP label and notification messages, as applicable.

4.2. Topology IDs

This document assumes the same definitions and procedures associated with MPLS MT-ID as defined in [RFC7307] specification.

5. MT Multipoint Capability

"MT Multipoint Capability" is a new LDP capability, defined in accordance with LDP Capability definition guidelines [RFC5561], that is to be advertised to its peers by an mLDP speaker to announce its capability to support MTR and the procedures specified in this document. This capability MAY be sent either in an Initialization

message at the session establishment time, or in a Capability message dynamically during the lifetime of a session (only if "Dynamic Announcement" capability [RFC5561] has been successfully negotiated with the peer).

The format of this capability is as follows:

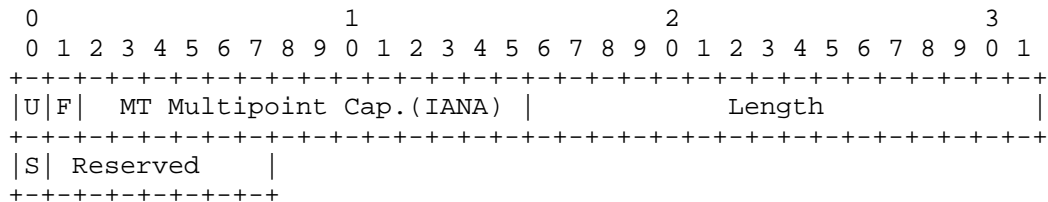


Figure 4: MT Multipoint Capability TLV Format

Where:

U- and F-bits: MUST be 1 and 0, respectively, as per Section 3 of LDP Capabilities [RFC5561].

MT Multipoint Capability: TLV type (IANA assigned).

Length: The length (in octets) of TLV. The value of this field MUST be 1 as there is no Capability-specific data [RFC5561] that follows in the TLV.

S-bit: Set to 1 to announce and 0 to withdraw the capability (as per [RFC5561]).

An mLDP speaker that has successfully advertised and negotiated "MT Multipoint" capability MUST support the following:

1. Topology-scoped mLDP FECs in LDP messages (Section 4.1)
 2. Topology-scoped mLDP forwarding setup (Section 7)
6. MT Applicability on FEC-based features
- 6.1. Typed Wildcard MP FEC Elements

[RFC5918] extends base LDP and defines Typed Wildcard FEC Element framework. Typed Wildcard FEC element can be used in any LDP message to specify a wildcard operation for a given type of FEC.

The MT extensions proposed in document do not require any extension in procedures for Typed Wildcard FEC Element support [RFC5918], and

these procedures apply as-is to Multipoint MT FEC wildcarding. Like Typed Wildcard MT Prefix FEC Element, as defined in [RFC7307], the MT extensions allow use of "MT IP" or "MT IPv6" in the Address Family field of the Typed Wildcard MP FEC element in order to use wildcard operations for MP FECs in the context of a given (sub-)topology as identified by the MT-ID and IPA field.

This document proposes following format and encoding for a Typed Wildcard MP FEC element:

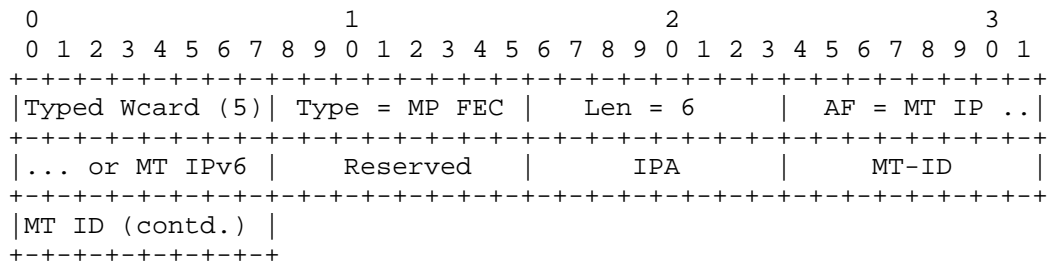


Figure 5: Typed Wildcard MT MP FEC Element

Where:

Type: One of MP FEC Element type (P2MP, MP2MPup, MP2MP-down).

MT ID: MPLS MT ID

IPA: The IGP Algorithm, values are from the IGP Algorithm registry.

The proposed format allows an LSR to perform wildcard MP FEC operations under the scope of a (sub-)topology.

6.2. End-of-LIB

[RFC5919] specifies extensions and procedures that allows an LDP speaker to signal its End-of-LIB (i.e. convergence) for a given FEC type towards a peer. MT extensions for MP FEC do not require any change in these procedures and they apply as-is to MT MP FEC elements. This means that an MT mLDp speaker MAY signal its convergence per (sub-)topology using MT Typed Wildcard MP FEC element.

7. Topology-Scoped Signaling and Forwarding

Since the {MT-ID, IPA} tuple is part of an mLDP FEC, there is no need to support the concept of multiple (sub-)topology forwarding tables in mLDP. Each MP LSP will be unique due to the tuple being part of the FEC. There is also no need to have specific label forwarding tables per topology, and each MP LSP will have its own unique local label in the table. However, In order to implement MTR in an mLDP network, the selection procedures for upstream LSR and downstream forwarding interface need be changed.

7.1. Upstream LSR selection

The procedures as described in RFC-6388 section-2.4.1.1 depend on the best path to reach the root. When the {MT-ID, IPA} tuple is signaled as part of the FEC, this tuple is used to select the (sub-)topology that must be used to find the best path to the root address. Using the next-hop from this best path, a LDP peer is selected following the procedures as defined in [RFC6388].

7.2. Downstream forwarding interface selection

The procedures as described in RFC-6388 section-2.4.1.2 describe how a downstream forwarding interface is selected. In these procedures, any interface leading to the downstream LDP neighbor can be considered as candidate forwarding interface. When the {MT-ID, IPA} tuple is part of the FEC, this is no longer true. An interface must only be selected if it is part of the same (sub-)topology that was signaled in the mLDP FEC element. Besides this restriction, the other procedures in [RFC6388] apply.

8. LSP Ping Extensions

[RFC6425] defines procedures to detect data plane failures in Multipoint MPLS LSPs. Section 3.1.2 of [RFC6425] defines new Sub-Types and Sub-TLVs for Multipoint LDP FECs to be sent in "Target FEC Stack" TLV of an MPLS echo request message [RFC4379].

To support LSP ping for MT Multipoint LSPs, this document uses existing sub-types "P2MP LDP FEC Stack" and "MP2MP LDP FEC Stack" defined in [RFC6425]. The proposed extension is to specify "MT IP" or "MT IPv6" in the "Address Family" field, set the "Address Length" field to 8 (for MT IP) or 20 (for MT IPv6), and encode the sub-TLV with additional {MT-ID, IPA} information as an extension to the "Root LSR Address" field. The resultant format of sub-tlv is as follows:

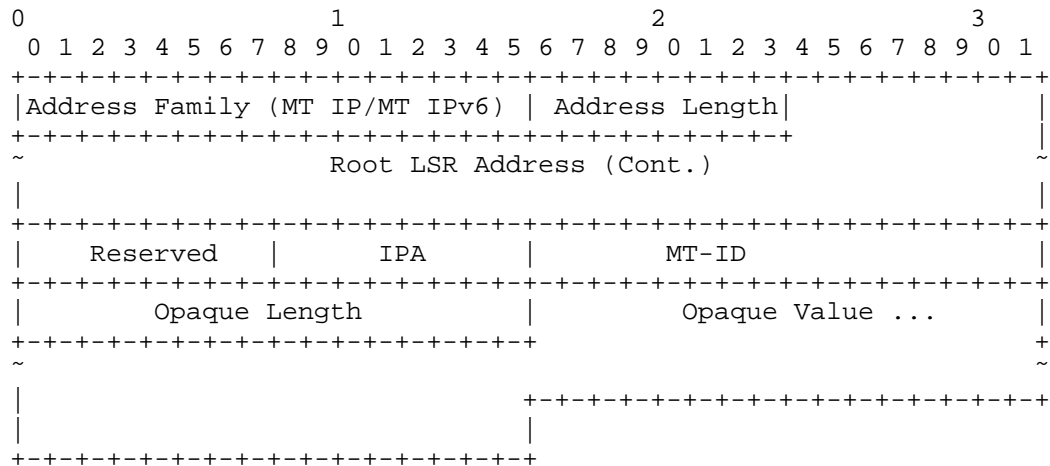


Figure 6: Multipoint LDP FEC Stack Sub-TLV Format for MT

The rules and procedures of using this new sub-TLV in an MPLS echo request message are same as defined for P2MP/MP2MP LDP FEC Stack Sub-TLV in [RFC6425] with only difference being that Root LSR address is now (sub-)topology scoped.

9. Security Considerations

This extension to mLDP does not introduce any new security considerations beyond that already apply to the base LDP specification [RFC5036], base mLDP specification [RFC6388], and MPLS security framework [RFC5920].

10. IANA Considerations

This document defines a new LDP capability parameter TLV. IANA is requested to assign the lowest available value after 0x0500 from "TLV Type Name Space" in the "Label Distribution Protocol (LDP) Parameters" registry within "Label Distribution Protocol (LDP) Name Spaces" as the new code point for the LDP TLV code point.

Value	Description	Reference	Notes/Registration Date
TBA	MT Multipoint Capability	This document	

Figure 7: IANA Code Point

11. Acknowledgments

The authors would like to acknowledge Eric Rosen for his input on this specification.

12. References

12.1. Normative References

- [I-D.hegdeppsenak-isis-sr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., and A. Gulko, "ISIS Segment Routing Flexible Algorithm", draft-hegdeppsenak-isis-sr-flex-algo-02 (work in progress), February 2018.
- [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>>.
- [RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <<https://www.rfc-editor.org/info/rfc4379>>.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <<https://www.rfc-editor.org/info/rfc4915>>.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.

- [RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <<https://www.rfc-editor.org/info/rfc6388>>.
- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <<https://www.rfc-editor.org/info/rfc7307>>.

12.2. Informative References

- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <<https://www.rfc-editor.org/info/rfc5561>>.
- [RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class (FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010, <<https://www.rfc-editor.org/info/rfc5918>>.
- [RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas, "Signaling LDP Label Advertisement Completion", RFC 5919, DOI 10.17487/RFC5919, August 2010, <<https://www.rfc-editor.org/info/rfc5919>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <<https://www.rfc-editor.org/info/rfc5920>>.

Authors' Addresses

IJsbrand Wijnands
Cisco Systems, Inc.
De kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8
Canada

Email: skraza@cisco.com

Zhaohui Zhang
Juniper Networks
10 Technology Park Dr.
Westford MA 01886
US

Email: zzhang@juniper.net

Arkadiy Gulko
Thomson Reuters
195 Broadway
New York NY 10007
USA

Email: arkadiy.gulko@thomsonreuters.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

IJ. Wijnands
Individual
K. Raza
M. Mishra
A. Budhiraja
Cisco Systems, Inc.
Z. Zhang
Juniper Networks
A. Gulko
Edward Jones wealth management
7 March 2022

mLDP Extensions for Multi-Topology Routing
draft-wijnands-mpls-mlbp-multi-topology-04

Abstract

Multi-Topology Routing (MTR) is a technology to enable service differentiation within an IP network. Flexible Algorithm (FA) is another mechanism of creating a sub-topology within a topology using defined topology constraints and computation algorithm. In order to deploy mLDP in a network that supports MTR and/or FA, mLDP is required to become topology and FA aware. This document specifies extensions to mLDP to support MTR with FA such that when building a Multi-Point LSPs it can follow a particular topology and algorithm.

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. Glossary	2
2. Introduction	3
3. Specification of Requirements	4
4. MT Scoped mLDP FECs	4
4.1. MP FEC Extensions for MT	4
4.1.1. MP FEC Element	5
4.1.2. MT IP Address Families	5
4.1.3. MT MP FEC Element	6
4.2. Topology IDs	7
5. MT Multipoint Capability	8
6. MT Applicability on FEC-based features	9
6.1. Typed Wildcard MP FEC Elements	9
6.2. End-of-LIB	10
7. Topology-Scoped Signaling and Forwarding	10
7.1. Upstream LSR selection	10
7.2. Downstream forwarding interface selection	10
8. LSP Ping Extensions	10
9. Security Considerations	11
10. IANA Considerations	11
11. Acknowledgments	12
12. References	12
12.1. Normative References	12
12.2. Informative References	13
Authors' Addresses	13

1. Glossary

MT - Multi-Topology

MT-ID - Multi-Topology Identifier

MTR - Multi-Topology Routing

IGP - Interior Gateway Protocol

MP - Multipoint (P2MP or MP2MP)

LDP - Label Distribution Protocol

mLDP - Multipoint LDP

P2MP - Point-to-Multipoint

MP2MP - Multipoint-to-Multipoint

FEC - Forwarding Equivalence Class

LSP - Label Switched Path

FA - Flexible Algorithm

IPA - IGP Algorithm

2. Introduction

Multi-Topology Routing (MTR) is a technology to enable service differentiation within an IP network. IGP protocols (OSPF and IS-IS) and LDP have already been extended to support MTR. To support MTR, an IGP maintains independent IP topologies, termed as "Multi-Topologies" (MT), and computes/installs routes per topology. OSPF extensions [RFC4915] and ISIS extensions [RFC5120] specify the MT extensions under respective IGPs. To support IGP MT, similar LDP extensions [RFC7307] have been specified to make LDP MT-aware and be able to setup unicast Label Switched Paths (LSPs) along IGP MT routing paths.

A more light weight mechanism to define constraint-based topologies is Flexible Algorithm (FA) [I-D.ietf-lsr-flex-algo]. FA can be seen as creating a sub-topology within a topology using defined topology constraints and computation algorithm. This can be done within a MTR topology or just the default Topology. An instance of such a sub-topology is identified by a 1 octet value as documented in [I-D.ietf-lsr-flex-algo]). Flexible Algorithm is a mechanism to create a sub-topology, but in the future different algorithms might be defined on how to achieve that. For that reason, in the remainder of this document we'll refer to this as the IGP Algorithm (IPA).

Multipoint LDP (mLDP) refers to extensions in LDP to setup multipoint LSPs (point-to-multipoint (P2MP) or multipoint-to-multipoint (MP2MP)), by means of set of extensions and procedures defined in [RFC6388]. In order to deploy mLDP in a network that supports MTR and FA, mLDP is required to become topology and algorithm aware. This document specifies extensions to mLDP to support MTR/IPA such that when building a Multi-Point LSPs it can follow a particular topology and algorithm. This means that the identifier for the particular Topology to be used by mLDP have to become a two tuple (MTR Topology Id, IGP Algorithm).

3. Specification of Requirements

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

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

4. MT Scoped mLDP FECs

As defined in [RFC7307], MPLS Multi-Topology Identifier (MT-ID) is an identifier that is used to associate an LSP with a certain MTR topology. In the context of MP LSPs, this identifier is part of the mLDP FEC encoding so that LDP peers are able to setup an MP LSP via their own defined MTR policy. In order to avoid conflicting MTR policies for the same mLDP FEC, the MT-ID needs to be a part of the FEC, so that different MT-ID values will result in unique MP-LSP FEC elements.

The same applies to the IPA. The IPA needs to be encoded as part of the mLDP FEC to create unique MP-LSPs and at the same time is used to signal to mLDP (hop-by-hop) which Algorithm needs to be used to create the MP-LSP.

Since the MT-ID and IPA are part of the FEC, they apply to all the LDP messages that potentially include an mLDP FEC element.

4.1. MP FEC Extensions for MT

Following subsections propose the extensions to bind an mLDP FEC to a topology. The mLDP MT extensions reuse some of the extensions specified in [RFC7307].

4.1.1. MP FEC Element

Base mLDP specification [RFC6388] defines MP FEC Element as follows:

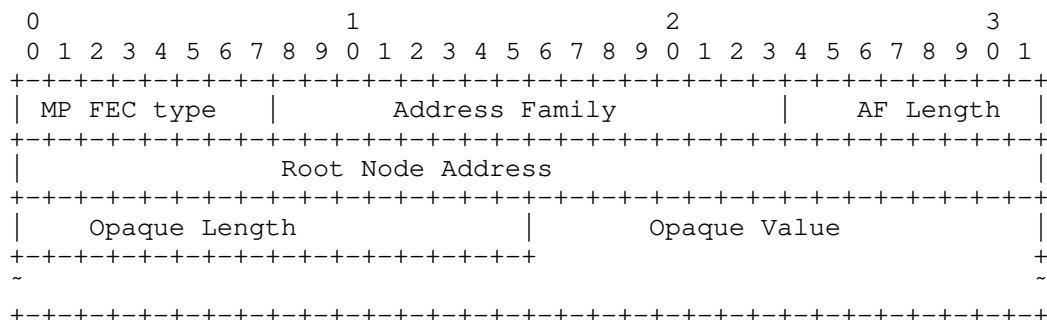


Figure 1: MP FEC Element Format [RFC6388]

Where "Root Node Address" encoding is as defined for given "Address Family", and whose length (in octets) is specified by the "AF Length" field.

To extend MP FEC elements for MT, the {MT-ID, IPA} is a tuple that is relevant in the context of the root address of the MP LSP. The {MT-ID, IPA} tuple determines in which (sub)-topology the root address needs to be resolved. Since the {MT-ID, IPA} tuple should be considered part of the mLDP FEC, the most natural place to encode this tuple is as part of the root address. While encoding it, we also propose to use "MT IP" Address Families as described in following sub section.

4.1.2. MT IP Address Families

[RFC7307] has specified new address families, named "MT IP" and "MT IPv6", to allow specification of an IP prefix within a topology scope. In addition to using this address family for mLDP, we also use 8 bits of the 16 bits Reserved field to encode the IGP Algorithm (IPA) Registry. The resulting format of the data associated with these new Address Families is as follows:

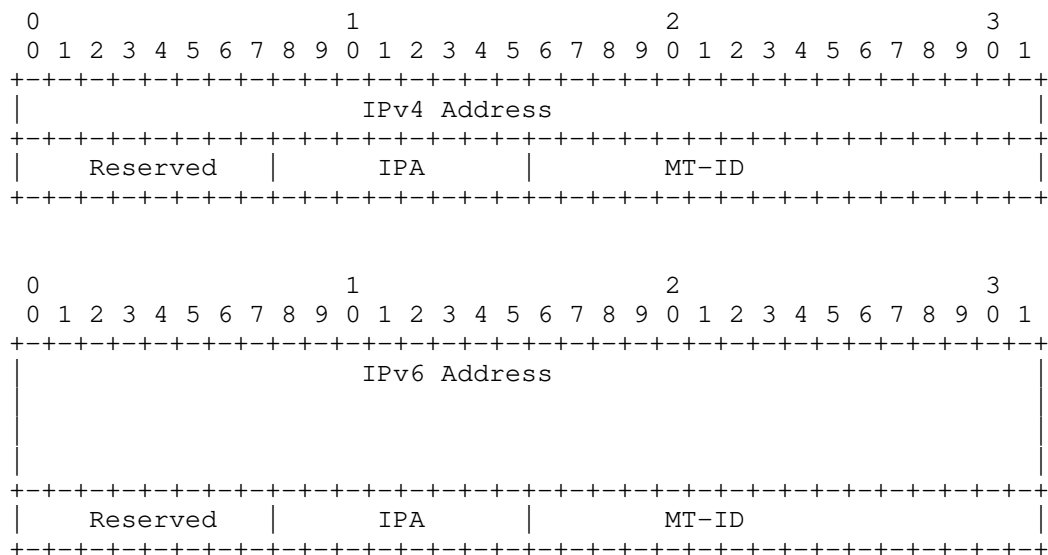


Figure 2: Modified MT IP Address Families Data Format

Where:

IPv4/IPv6 Address: An IP address corresponding to "MT IP" and "MT IPv6" address families respectively.

IPA: The IGP Algorithm, values are from the IGP Algorithm registry.

Reserved: This 8-bit field SHOULD be zero.

4.1.3. MT MP FEC Element

By using extended MT IP Address Family, the resultant MT MP FEC element is to be encoded as follows:

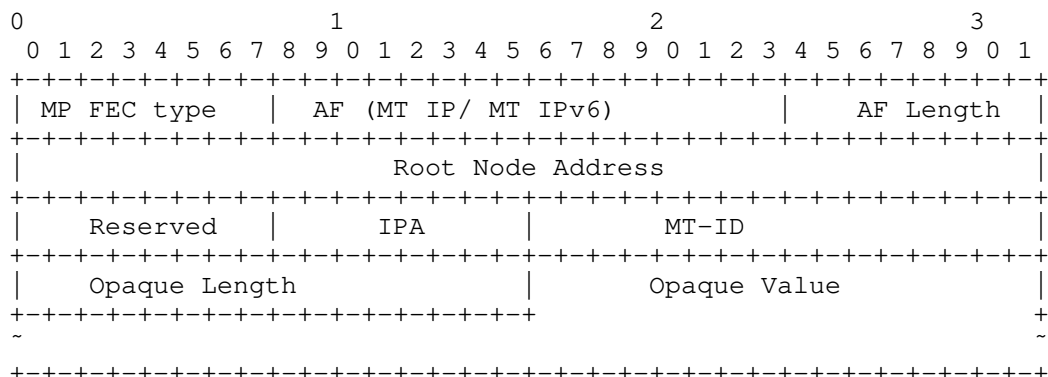


Figure 3: IP MT-Scoped MP FEC Element Format

In the context of this document, the applicable LDP FECs for MT mLDP include:

* MP FEC Elements:

- P2MP (type 0x6)
- MP2MP-up (type 0x7)
- MP2MP-down (type 0x8)

* Typed Wildcard FEC Element (type 0x5)

In case of "Typed Wildcard FEC Element", the sub FEC Element type MUST be one of the MP FECs listed above.

This specification allows the use of Topology-scoped mLDP FECs in LDP label and notification messages, as applicable.

4.2. Topology IDs

This document assumes the same definitions and procedures associated with MPLS MT-ID as defined in [RFC7307] specification.

5. MT Multipoint Capability

"MT Multipoint Capability" is a new LDP capability, defined in accordance with LDP Capability definition guidelines [RFC5561], that is to be advertised to its peers by an mLDP speaker to announce its capability to support MTR and the procedures specified in this document. This capability MAY be sent either in an Initialization message at the session establishment time, or in a Capability message dynamically during the lifetime of a session (only if "Dynamic Announcement" capability [RFC5561] has been successfully negotiated with the peer).

The format of this capability is as follows:

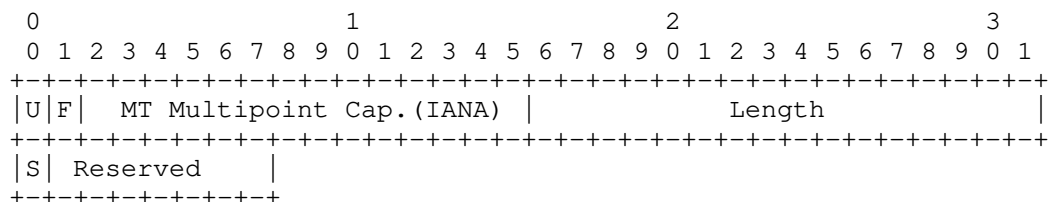


Figure 4: MT Multipoint Capability TLV Format

Where:

U- and F-bits: MUST be 1 and 0, respectively, as per Section 3 of LDP Capabilities [RFC5561].

MT Multipoint Capability: TLV type (IANA assigned).

Length: The length (in octets) of TLV. The value of this field MUST be 1 as there is no Capability-specific data [RFC5561] that follows in the TLV.

S-bit: Set to 1 to announce and 0 to withdraw the capability (as per [RFC5561]).

An mLDP speaker that has successfully advertised and negotiated "MT Multipoint" capability MUST support the following:

1. Topology-scoped mLDP FECs in LDP messages (Section 4.1)
2. Topology-scoped mLDP forwarding setup (Section 7)

6. MT Applicability on FEC-based features

6.1. Typed Wildcard MP FEC Elements

[RFC5918] extends base LDP and defines Typed Wildcard FEC Element framework. Typed Wildcard FEC element can be used in any LDP message to specify a wildcard operation for a given type of FEC.

The MT extensions proposed in document do not require any extension in procedures for Typed Wildcard FEC Element support [RFC5918], and these procedures apply as-is to Multipoint MT FEC wildcarding. Like Typed Wildcard MT Prefix FEC Element, as defined in [RFC7307], the MT extensions allow use of "MT IP" or "MT IPv6" in the Address Family field of the Typed Wildcard MP FEC element in order to use wildcard operations for MP FECs in the context of a given (sub)-topology as identified by the MT-ID and IPA field.

This document proposes following format and encoding for a Typed Wildcard MP FEC element:

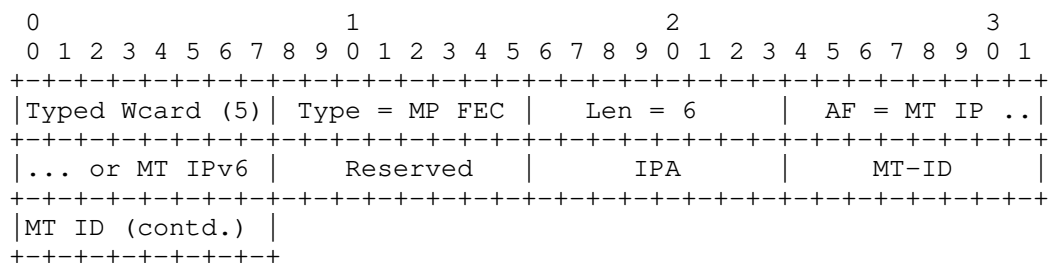


Figure 5: Typed Wildcard MT MP FEC Element

Where:

Type: One of MP FEC Element type (P2MP, MP2MPup, MP2MP-down).

MT ID: MPLS MT ID

IPA: The IGP Algorithm, values are from the IGP Algorithm registry.

The proposed format allows an LSR to perform wildcard MP FEC operations under the scope of a (sub-)topology.

6.2. End-of-LIB

[RFC5919] specifies extensions and procedures that allows an LDP speaker to signal its End-of-LIB (i.e. convergence) for a given FEC type towards a peer. MT extensions for MP FEC do not require any change in these procedures and they apply as-is to MT MP FEC elements. This means that an MT mLDP speaker MAY signal its convergence per (sub-)topology using MT Typed Wildcard MP FEC element.

7. Topology-Scoped Signaling and Forwarding

Since the {MT-ID, IPA} tuple is part of an mLDP FEC, there is no need to support the concept of multiple (sub-)topology forwarding tables in mLDP. Each MP LSP will be unique due to the tuple being part of the FEC. There is also no need to have specific label forwarding tables per topology, and each MP LSP will have its own unique local label in the table. However, In order to implement MTR in an mLDP network, the selection procedures for upstream LSR and downstream forwarding interface need to be changed.

7.1. Upstream LSR selection

The procedures as described in RFC-6388 section-2.4.1.1 depend on the best path to reach the root. When the {MT-ID, IPA} tuple is signaled as part of the FEC, this tuple is used to select the (sub-)topology that must be used to find the best path to the root address. Using the next-hop from this best path, a LDP peer is selected following the procedures as defined in [RFC6388].

7.2. Downstream forwarding interface selection

The procedures as described in RFC-6388 section-2.4.1.2 describe how a downstream forwarding interface is selected. In these procedures, any interface leading to the downstream LDP neighbor can be considered as candidate forwarding interface. When the {MT-ID, IPA} tuple is part of the FEC, this is no longer true. An interface must only be selected if it is part of the same (sub-)topology that was signaled in the mLDP FEC element. Besides this restriction, the other procedures in [RFC6388] apply.

8. LSP Ping Extensions

[RFC6425] defines procedures to detect data plane failures in Multipoint MPLS LSPs. Section 3.1.2 of [RFC6425] defines new Sub-Types and Sub-TLVs for Multipoint LDP FECs to be sent in "Target FEC Stack" TLV of an MPLS echo request message [RFC4379].

To support LSP ping for MT Multipoint LSPs, this document uses existing sub-types "P2MP LDP FEC Stack" and "MP2MP LDP FEC Stack" defined in [RFC6425]. The proposed extension is to specify "MT IP" or "MT IPv6" in the "Address Family" field, set the "Address Length" field to 8 (for MT IP) or 20 (for MT IPv6), and encode the sub-TLV with additional {MT-ID, IPA} information as an extension to the "Root LSR Address" field. The resultant format of sub-tlv is as follows:

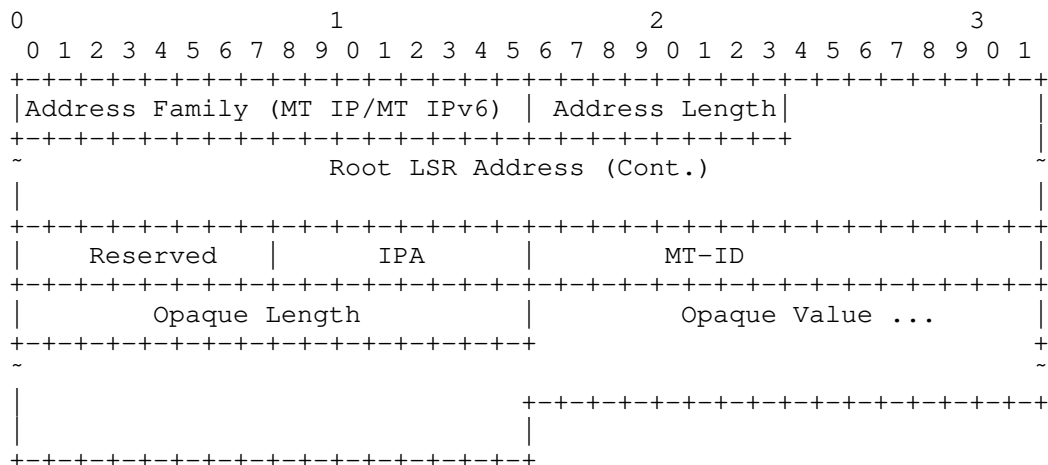


Figure 6: Multipoint LDP FEC Stack Sub-TLV Format for MT

The rules and procedures of using this new sub-TLV in an MPLS echo request message are same as defined for P2MP/MP2MP LDP FEC Stack Sub-TLV in [RFC6425] with only difference being that Root LSR address is now (sub-)topology scoped.

9. Security Considerations

This extension to mLDP does not introduce any new security considerations beyond that already apply to the base LDP specification [RFC5036], base mLDP specification [RFC6388], and MPLS security framework [RFC5920].

10. IANA Considerations

This document defines a new LDP capability parameter TLV. IANA is requested to assign the lowest available value after 0x0500 from "TLV Type Name Space" in the "Label Distribution Protocol (LDP) Parameters" registry within "Label Distribution Protocol (LDP) Name Spaces" as the new code point for the LDP TLV code point.

Value	Description	Reference	Notes/Registration Date
TBA	MT Multipoint Capability	This document	

Figure 7: IANA Code Point

11. Acknowledgments

The authors would like to acknowledge Eric Rosen for his input on this specification.

12. References

12.1. Normative References

- [I-D.ietf-lsr-flex-algo]
 Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", Work in Progress, Internet-Draft, draft-ietf-lsr-flex-algo-18, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lsr-flex-algo-18.txt>>.
- [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>>.
- [RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <<https://www.rfc-editor.org/info/rfc4379>>.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <<https://www.rfc-editor.org/info/rfc4915>>.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-IS)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.

- [RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <<https://www.rfc-editor.org/info/rfc6388>>.
- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <<https://www.rfc-editor.org/info/rfc7307>>.

12.2. Informative References

- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <<https://www.rfc-editor.org/info/rfc5561>>.
- [RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class (FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010, <<https://www.rfc-editor.org/info/rfc5918>>.
- [RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas, "Signaling LDP Label Advertisement Completion", RFC 5919, DOI 10.17487/RFC5919, August 2010, <<https://www.rfc-editor.org/info/rfc5919>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <<https://www.rfc-editor.org/info/rfc5920>>.

Authors' Addresses

IJsbrand Wijnands
Individual
Email: ice@braindump.be

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata ON K2K-3E8
Canada
Email: skraza@cisco.com

Mankamana Mishra
Cisco Systems, Inc.
821 Alder Drive
Milpitas, CA 95035
United States of America
Email: mankamis@cisco.com

Anuj Budhiraja
Cisco Systems, Inc.
821 Alder Drive
Milpitas, CA 95035
United States of America
Email: abudhira@cisco.com

Zhaohui Zhang
Juniper Networks
10 Technology Park Dr.
Westford, MA 01886
United States of America
Email: zzhang@juniper.net

Arkadiy Gulko
Edward Jones wealth management
United States of America
Email: Arkadiy.gulko@edwardjones.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 9, 2019

J. Xie
Huawei Technologies
M. Chen
R. Li
Huawei
September 5, 2018

Multipoint LDP Extension for P2MP-based BIER
draft-xie-mpls-ldp-bier-extension-01

Abstract

Bit Index Explicit Replication (BIER) is a new architecture that provides optimal multicast forwarding without requiring intermediate routers to maintain any per-flow state by using a multicast-specific BIER header. An extension to the Label Distribution Protocol (LDP) defined in [RFC5036] for the setup of point-to-multipoint (P2MP) is described in [RFC6388] is called mLDP, and is used for multicast-specific tree building. This document describes mLDP extensions to setup a P2MP with BIER information, which is called P2MP based BIER in [I-D.xie-bier-mvpn-mpls-p2mp].

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 March 9, 2019.

Copyright Notice

Copyright (c) 2018 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. MLDP P2MP BIER Signalling	4
3.1. Definitions	4
3.2. Example	4
3.3. Signaling the P2MP BIER	5
3.4. The P2MP BIER LSP Identifier	5
3.5. BIER TLV	6
3.6. Make Before Break (MBB)	7
4. Capability and Error handling	7
4.1. BIER Capability	7
4.2. BIER Status	8
4.3. Check Capability and Use Status for Error Handling	9
5. IANA Considerations	11
6. Security Considerations	11
7. Acknowledgements	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Authors' Addresses	13

1. Introduction

Bit Index Explicit Replication (BIER) is a new architecture that provides optimal multicast forwarding without requiring intermediate routers to maintain any per-flow state by using a multicast-specific BIER header. An extension to the Label Distribution Protocol (LDP) defined in [RFC5036] for the setup of point-to-multipoint (P2MP) is described in [RFC6388] is called mLDP, and is used for multicast-specific tree building. This document describes mLDP extensions to

setup a P2MP with BIER information, which is called a P2MP based BIER in [I-D.xie-bier-mvpn-mpls-p2mp] .

Related documents that may be of interest include [RFC5561], and [RFC3988].

2. Terminology

Readers of this document are assumed to be familiar with the terminology and concepts of the documents listed as Normative References. For convenience, some of the more frequently used terms and new terms list below.

- o mLDP: Multipoint extensions for LDP.
- o P2MP LSP: An LSP that has one Ingress LSR and one or more Egress LSRs.
- o Ingress LSR: An Ingress LSR for a particular LSP is an LSR that can send a data packet along the LSP. MP2MP LSPs can have multiple Ingress LSRs, P2MP LSPs have just one, and that node is often referred to as the "root node".
- o Egress LSR: An Egress LSR for a particular LSP is an LSR that can remove a data packet from that LSP for further processing. P2P and MP2P LSPs have only a single egress node, but P2MP and MP2MP LSPs can have multiple egress nodes.
- o Transit LSR: An LSR that has reachability to the root of the MP LSP via a directly connected upstream LSR and one or more directly connected downstream LSRs.
- o Bud LSR: An LSR that is an egress but also has one or more directly connected downstream LSRs.
- o Leaf node: A leaf node can be either an Egress or Bud LSR when referred to in the context of a P2MP LSP.
- o FEC: Forwarding Equivalence Class
- o P2MP FEC: defined in RFC6388.
- o F-BM: Forwarding Bit Mask
- o BSL: Bit String Length, that is 64, 128, 256, etc (per [RFC8279]).

3. MLDP P2MP BIER Signalling

3.1. Definitions

F-BM: Forwarding Bit Mask, An array of Bit, which is defined in [RFC8279].

Peer F-BM: For LSR A and P2MP FEC<Root,N>, this is the F-BM that included in Label Mapping from a downstream LSR for P2MP FEC<Root,N> to A.

Downstream F-BM: For LSR A and P2MP FEC<Root,N>, this is the OR'ing result of each of the F-BM included in Label Mapping from downstream LSR for P2MP FEC<Root,N> to A. A use this Downstream F-BM in its Label Mapping to upstream node for P2MP FEC<Root,N>. In other words, A's Downstream F-BM is A's upstream node's Peer F-BM.

3.2. Example

Consider LSRs A - F, interconnected as follows:

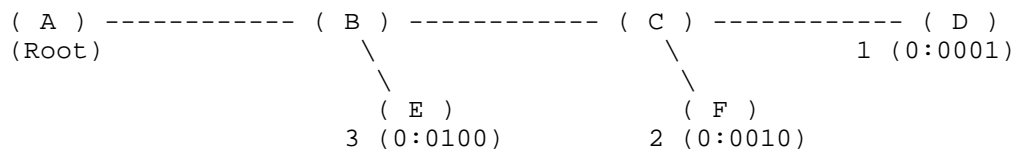


Figure 1: P2MP-based BIER Topology

Say that the node D has a BFR-id 1, F has a BFR-id 2, and E has a BFR-id 3, and we use a Bit String Length 4 (which is not valid per [RFC8296]) as an example.

Consider an P2MP FEC<Root=A,N=10> for which A is the Root, and say that D,E,F are all the Leafs of this P2MP FEC<Root=A, N=10>.

While D send LDP Mapping to C, it includes a F-BM 0001. Say that C got a Peer<D> F-BM 0001, and then C form a Downstream F-BM 0001.

While F send LDP Mapping to C, it includes a F-BM 0010. Say that C got a Peer<F> F-BM 0010, and then C form a Downstream F-BM 0011.

While C send LDP Mapping to B, it includes a F-BM 0010. Say that B got a Peer<C> F-BM 0011, and then B form a Downstream F-BM 0011.

While E send LDP Mapping to B, it includes a F-BM 0100. Say that B got a Peer<E> F-BM 0100, and then B form a Downstream F-BM 0111.

While B send LDP Mapping to A, it includes a F-BM 0111. Say that A got a Peer F-BM 0111, and then A form a Downstream F-BM 0111.

This memo describes how every nodes in a P2MP tree get Peer F-BM from every downstream LSR, form a Downstream F-BM by OR'ing all it's Downstream Peer F-BM, and send a Mapping to it's upstream node using the Downstream F-BM.

3.3. Signaling the P2MP BIER

The procedure for signalling the P2MP BIER is performed hop-by-hop by each LSR L along an P2MP LSP for a given P2MP FEC<Root,N>. The steps are as follows:

1. First, L computes its Downstream F-BM for P2MP FEC<Root,N>:

If L is a leaf for P2MP FEC<Root,N>, L sets the F-BM with it's BFRID's BitPosition to 1.

Otherwise (L is not a Leaf), L computes the Downstream F-BM by OR'ing all it's downstream Node's F-BM, as described above.

2. For each LDP neighbor of L to which L decides to send a Mapping for FEC F, L attaches an BIER TLV with the F-BM that it computed for this P2MP FEC.

3. When a new BIER TLV is received for P2MP FEC<Root,N> from a downstream LSR or the set of downstream LSRs, L returns to step 1. If the newly computed Downstream F-BM is unchanged, L SHOULD NOT advertise new information to its upstream neighbor. Otherwise, L readvertises its Mappings to its upstream neighbor with an updated BIER TLV.

This behavior is standard for attributes such as path vector, hop count, and MTU, and the same rules apply, as specified in [RFC5036].

If the Downstream F-BM changes, L MAY readvertise the new F-BM immediately, or hold down the readvertisement for a while.

3.4. The P2MP BIER LSP Identifier

[RFC6388] defined the P2MP FEC Element, which include a LDP MP Opaque Value Element. It also defined a Generic LSP Identifier as the LDP MP Opaque Value Element, with a TLV of Type 1. This document defined a new type of LDP MP Opaque Value Element, called the P2MP BIER LSP Identifier.

The encoding for the P2MP BIER LSP Identifier is as follows:

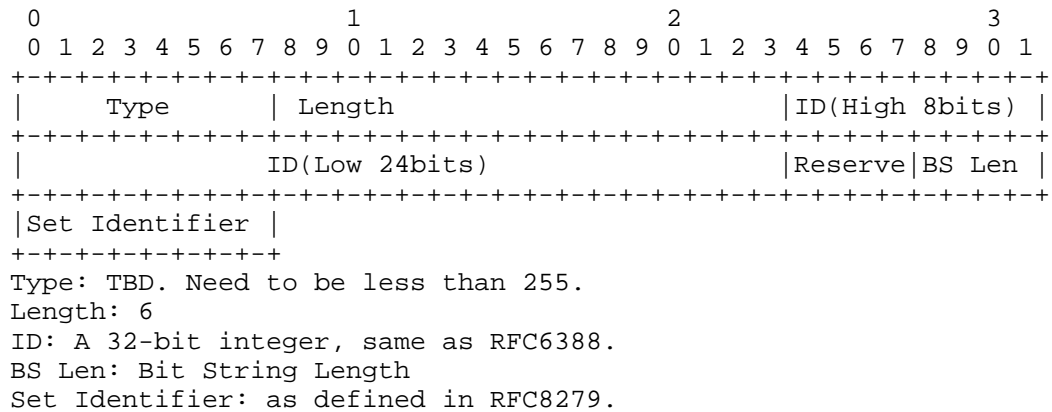


Figure 2: P2MP BIER LSP Identifier

3.5. BIER TLV

The BIER TLV encodes information on the F-BM for an LSP, from the advertising LSR to the egress(es) over all valid paths.

The encoding for the BIER TLV is as follows:

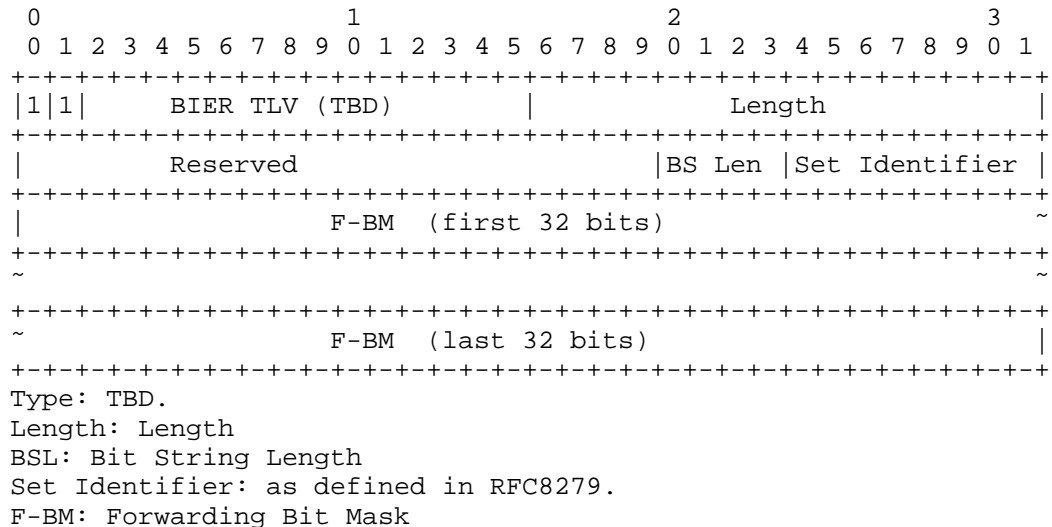


Figure 3: BIER TLV

3.6. Make Before Break (MBB)

The Make Before Break (MBB) mechanism for mLDP P2MP defined in RFC6388 also applies.

4. Capability and Error handling

The extensions defined in this document utilize the existing LDP error handling defined in [RFC5036]. If an LSR receives an error notification from a peer for a session, it terminates the LDP session by closing the TCP transport connection for the session and discarding all multi-topology label mappings learned via the session.

An LSR should respond with an LDP MP Status in LDP Notification Messages when it receives an LDP Label Mapping message with a P2MP FEC element specifying an BIER TLV that is not locally known or not supported. The LSR MUST also discard the entire message before sending the Notification message.

4.1. BIER Capability

A new optional capability parameter TLV, RMR Capability, is defined. Following is the format of the RMR Capability Parameter:

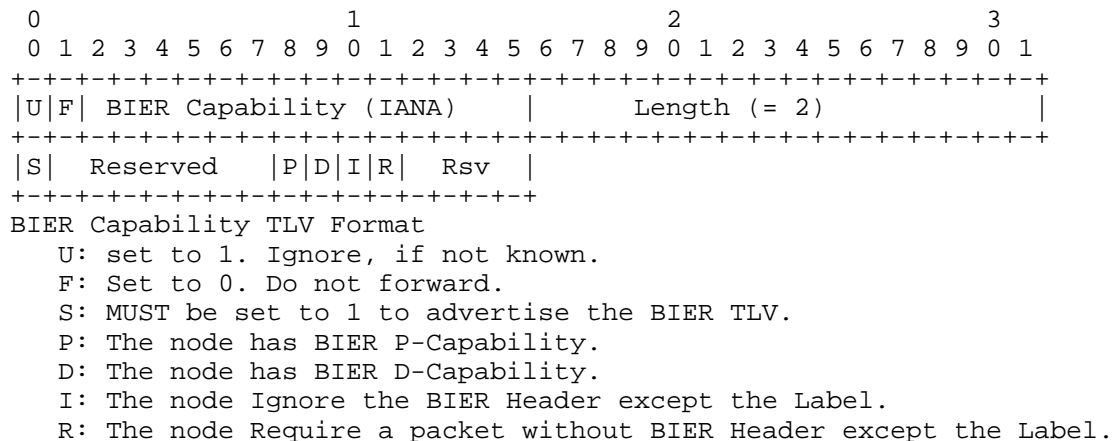


Figure 4: BIER Capability

The BIER Capability TLV MUST be advertised in the LDP Initialization message. If the peer has not advertised the BIER capability, then label messages including a BIER TLV MUST NOT be sent to the peer.

If an LSR has not advertised that it is BIER capable, its LDP peers MUST NOT send it messages that include BIER TLV. If an LSR receives

a Label Mapping message with an BIER TLV from downstream LSR-D and its upstream LSR-U has not advertised that it is BIER capable, the LSR MUST send an BIER notification immediately to LSR-D. If this happens, an P2MP BIER LSP will not be established, a normal P2MP LSP will not be established either.

P-Capability indicate a complete BIER function, which includes P-Capability and D-Capability. If a node support P-Capability, then it support the whole BIER function, which means it support both P-capability and D-capability.

D-Capability indicate a subset of BIER function, to Disposition some length of a packet from some offset. For example, from BIER Label and a whole (BIER Header Length) , or from the position after BIER Label and a length of (BIER Header Length - BIER Label Length) . If a node don't support P-Capability, it may still support D-Capability. If a node don't support D-capability, it must not support P-Capability.

If a Node doesn't have P-Capability, then P flag MUST be cleared. Whether the node will be a Branch or BUD or Leaf, the I flag SHOULD be set.

if a node doesn't have D-Capability, then P and D flag MUST be cleared. If the node will be a BUD or Leaf then R flag SHOULD be set. if the node will be a Branch then R flag MAY not be set.

if a node doesn't have P-Capability but does have D-Capability, then D flag SHOULD be set, but R flag MAY be set or not be set.

4.2. BIER Status

A new optional LDP MP Status Value Element (RFC6388), BIER Status, is defined. Following is the format of the BIER Status:

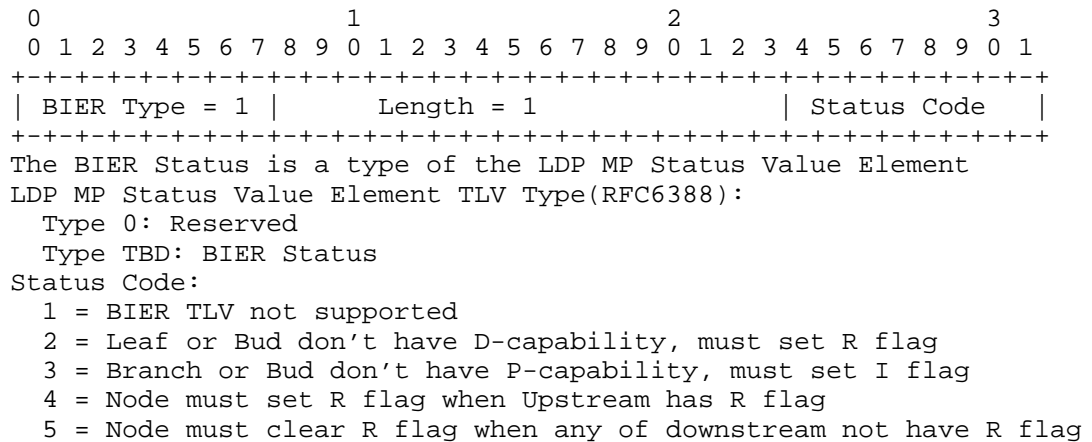


Figure 5: BIER Status TLV

4.3. Check Capability and Use Status for Error Handling

In order to deploy P2MP BIER on a network with some legacy nodes, which may not support P-capability or D-capability, some Capability flags need to be checked and notification messages may be generated.

If all edge nodes support D-capability, but some nodes don't support P-capability and they set a I flag as required, then deployment of P2MP BIER is fine, and a BIER packet can walk from Root to all Leaf(s) without any change except the BIER Label.

If an LSR support P-capability, and it's upstream node also support P-capability, and when the LSR receives a Label Mapping message with an BIER TLV and R flag set from downstream LSR-D, the LSR will accept such Label Mapping message. If receives a Label Mapping wiht an BIER TLV and R flag cleaned another downstream LSR-D', the LSR will accept too.

If an LSR receives a Label Mapping message with an BIER TLV from downstream LSR-D with a R flag, and its upstream LSR-U has no P-capability, the LSR MUST send an BIER notification immediately to LSR-D. If this happens, an P2MP BIER LSP will not be established, a normal P2MP LSP will not be established either.

When an LSR receives a Label Mapping message, it need to do some check before process and build P2MP BIER forwarding table. Such check includes:

- 1) Check if the node's P-capability and D-capability conflict with it's I flag and R flag.

The following table list the whole check matrix.

NODE's Role	NODE's PDIR-flag	Check Result	Rule
Leaf	[PDxx]	OK	(*) Comment 1
Leaf	[-Dxx]	OK	
Leaf	[--xR]	OK	Rule 1
Leaf	[--x-]	ERR	Rule 1
Branch	[PDxx]	OK	(*) Comment 1
Branch	[-DIx]	OK	Rule 2
Branch	[-D-x]	ERR	Rule 2
Branch	[--Ix]	OK	(*) Comment 2
Branch	[---x]	ERR	Rule 2
BUD	[PDxx]	OK	(*) Comment 1
BUD	[-DIx]	OK	Rule 2
BUD	[-D-x]	ERR	Rule 2
BUD	[--IR]	OK	
BUD	[--I-]	ERR	Rule 1
BUD	[---R]	ERR	Rule 2
BUD	[----]	ERR	Rule 1 & 2

(*) Comment 1: In some cases, set a R flag is useful
 (*) Comment 2: In some cases, Clear R flag on a Branch node is useful
 Rule 1: Leaf don't have D-capability, must set R flag
 Rule 2: Branch don't have P-capability, must set I flag

Figure 6: BIER Self Check Matrix

2) Check the node's R flag, the node's upstream R flag, the node's downstream R flag.

The following table list the whole check martrix about the R flag of node, the upstream node, the downstream nodes.

7. Acknowledgements

TBD

8. References

8.1. Normative References

- [I-D.ietf-bier-mvpn]
Rosen, E., Sivakumar, M., Aldrin, S., Dolganow, A., and T. Przygienda, "Multicast VPN Using BIER", draft-ietf-bier-mvpn-11 (work in progress), March 2018.
- [I-D.xie-bier-mvpn-mpls-p2mp]
Xie, J., McBride, M., Chen, M., and L. Geng, "Multicast VPN Using MPLS P2MP and BIER", draft-xie-bier-mvpn-mpls-p2mp-02 (work in progress), July 2018.
- [RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <<https://www.rfc-editor.org/info/rfc3988>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <<https://www.rfc-editor.org/info/rfc5561>>.
- [RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <<https://www.rfc-editor.org/info/rfc6388>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

8.2. 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>>.

Authors' Addresses

Jingrong Xie
Huawei Technologies
Q15 Huawei Campus, No.156 Beiqing Rd.
Beijing 100095
China

Email: xiejingrong@huawei.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Robin Li
Huawei

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 9, 2019

J. Xie
Huawei Technologies
M. Chen
R. Li
Huawei
September 5, 2018

RSVP-TE Extensions for P2MP-based BIER
draft-xie-mpls-rsvp-bier-extension-01

Abstract

Bit Index Explicit Replication (BIER) is a new architecture that provides optimal multicast forwarding without requiring intermediate routers to maintain any per-flow state by using a multicast-specific BIER header. This document describes extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for the set up of Traffic Engineered (TE) point-to-multipoint (P2MP) Label Switched Paths (LSPs) with BIER information, which is called P2MP based BIER in [I-D.xie-bier-mvpn-mpls-p2mp].

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 March 9, 2019.

Copyright Notice

Copyright (c) 2018 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
3. RSVP Extensions	3
3.1. Example of signaling the P2MP-BIER	3
3.2. PATH Message	4
3.3. RESV Message	5
3.4. SESSION Object	7
3.4.1. P2MP BIER Tunnel IPv4 SESSION Object	7
3.4.2. P2MP BIER Tunnel IPv6 SESSION Object	8
3.5. SENDER_TEMPLATE Object	8
3.5.1. P2MP BIER Tunnel IPv4 SENDER_TEMPLATE Object	9
3.5.2. P2MP BIER Tunnel IPv6 SENDER_TEMPLATE Object	9
3.6. S2L_BIER_SUB_LSP Object	10
3.6.1. S2L_BIER_SUB_LSP IPv4 Object	10
3.6.2. S2L_BIER_SUB_LSP IPv6 Object	11
3.7. FILTER_SPEC Object	11
3.7.1. P2MP BIER_IPv4 FILTER_SPEC Object	11
3.7.2. P2MP BIER_IPv6 FILTER_SPEC Object	11
4. Capability and Error Handling	11
5. IANA Considerations	12
6. Security Considerations	12
7. Acknowledgements	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Authors' Addresses	13

1. Introduction

Bit Index Explicit Replication (BIER) is a new architecture that provides optimal multicast forwarding without requiring intermediate routers to maintain any per-flow state by using a multicast-specific BIER header. [RFC4875] defines extensions to the RSVP-TE protocol ([RFC3209] and [RFC3473]) to support P2MP TE LSPs satisfying the set of requirements described in [RFC4461] .

This document extends RSVP-TE to establish P2MP TE LSPs with BIER information, which is called P2MP based BIER in [I-D.xie-bier-mvpn-mpls-p2mp].

2. Terminology

Readers of this document are assumed to be familiar with the terminology and concepts of the documents listed as Normative References. For convenience, some of the more frequently used terms and new terms list below.

- o LSP: Label Switch Path
- o LSR: Label Switching Router
- o BFR: BIER Forwarding Router
- o BFR-ID: BIER Forwarding Router IDentify.
- o P2MP: Point to Multi-point
- o P2MP based BIER: BIER using P2MP as topology

3. RSVP Extensions

RSVP Extensions to setup a P2MP-based BIER is very similar to the setup of a P2MP LSP described in [RFC4875]. Most of the structure and description are borrowed from RFC4875, and a precursive example is put in the beginning to give a whole picture of building the forwarding state of P2MP based BIER.

3.1. Example of signaling the P2MP-BIER

Consider LSRs A - F, interconnected as follows:

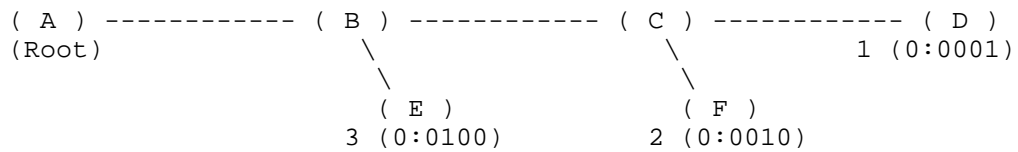


Figure 1: P2MP-based BIER Topology

Say that the node D has a BFR-id 1, F has a BFR-id 2, and E has a BFR-id 3, and we use a Bit String Length 4.

Consider an P2MP SESSION<P2MPID, TunnelID, ExtTunnelID=RootAddr>, for which A is the Root, and say that D,E,F are all the Leafs of this P2MP SESSION.

There are 3 Sub-LSPs: A-->B-->E, A-->B-->C-->D, A-->B-->C-->F.

PATH message: When PATH message walk through A-->B-->E, it include an session attribute that identify ths session is to establish a P2MP-based BIER LSP. The same to A-->B-->C-->D and A-->B-->C-->F.

RESV message: When RESV message work throuth A<--B<--E, it include an Object that identify BFR-ID of E. The same to A<--B<--C<--D and A<--B<--C<--F.

Procedure: B learns that it's downstream endpoint has a BFR-ID<3> after a RSVP message passes through A<--B<--E. B also learns a BFR-ID<1> after a RSVP message passes throuth A<--B<--C<--D, and a BFR-ID<2> after a RSVP message passes through A<--B<--C<--D.

3.2. PATH Message

This section describes modifications made to the Path message format as specified in [RFC4875]. The Path message is enhanced to signal one or more S2L sub-LSPs with BIER information. This is done by including the S2L BIER sub-LSP descriptor list in the Path message as shown below.

```

<Path Message> ::=
    <Common Header> [ <INTEGRITY> ]
    [ [ <MESSAGE_ID_ACK> | <MESSAGE_ID_NACK> ] ... ]
    [ <MESSAGE_ID> ]
    <SESSION> <RSVP_HOP>
    <TIME_VALUES>
    [ <EXPLICIT_ROUTE> ]
    <LABEL_REQUEST>
    [ <PROTECTION> ]
    [ <LABEL_SET> ... ]
    [ <SESSION_ATTRIBUTE> ]
    [ <NOTIFY_REQUEST> ]
    [ <ADMIN_STATUS> ]
    [ <POLICY_DATA> ... ]
    <sender descriptor>
    [ <S2L BIER sub-LSP descriptor list> ]
<S2L BIER sub-LSP descriptor list> ::= <S2L BIER sub-LSP descriptor>
    [ <S2L BIER sub-LSP descriptor list> ]
<S2L BIER sub-LSP descriptor> ::= <S2L_BIER_SUB_LSP>
    [ <P2MP_SECONDARY_EXPLICIT_ROUTE> ]

```

Figure 2: PATH Message

3.3. RESV Message

The Resv message follows the [RFC4875] format:


```

<Resv Message> ::=
    <Common Header> [ <INTEGRITY> ]
    [ [ <MESSAGE_ID_ACK> | <MESSAGE_ID_NACK> ] ... ]
    [ <MESSAGE_ID> ]
    <SESSION> <RSVP_HOP>
    <TIME_VALUES>
    [ <RESV_CONFIRM> ] [ <SCOPE> ]
    [ <NOTIFY_REQUEST> ]
    [ <ADMIN_STATUS> ]
    [ <POLICY_DATA> ... ]
    <STYLE> <flow descriptor list>

<flow descriptor list> ::= <FF flow descriptor list>
    | <SE flow descriptor>

<FF flow descriptor list> ::= <FF flow descriptor>
    | <FF flow descriptor list>
    <FF flow descriptor>

<SE flow descriptor> ::= <FLOWSPEC> <SE filter spec list>

<SE filter spec list> ::= <SE filter spec>
    | <SE filter spec list> <SE filter spec>

<FF flow descriptor> ::= [ <FLOWSPEC> ] <FILTER_SPEC> <LABEL>
    [ <RECORD_ROUTE> ]
    [ <S2L BIER sub-LSP flow descriptor list> ]

<SE filter spec> ::= <FILTER_SPEC> <LABEL> [ <RECORD_ROUTE> ]
    [ <S2L BIER sub-LSP flow descriptor list> ]

<S2L BIER sub-LSP flow descriptor list> ::=
    <S2L BIER sub-LSP flow descriptor>
    [ <S2L BIER sub-LSP flow descriptor list> ]

<S2L BIER sub-LSP flow descriptor> ::= <S2L_BIER_SUB_LSP>
    [ <P2MP_SECONDARY_RECORD_ROUTE> ]

```

Figure 3: RESV Message

FILTER_SPEC is defined in below section.

The S2L BIER sub-LSP flow descriptor has the same format as S2L BIER sub-LSP descriptor in previous section with the difference that a P2MP_SECONDARY_RECORD_ROUTE object is used in place of a P2MP_SECONDARY_EXPLICIT_ROUTE object.

Note that a Resv message can signal multiple S2L BIER sub-LSPs that may belong to the same FILTER_SPEC object or different FILTER_SPEC

objects. The same label SHOULD be allocated if the <Sender Address, LSP-ID> fields of the FILTER_SPEC object are the same.

However different labels MUST be allocated if the <Sender Address, LSP-ID> of the FILTER_SPEC object is different, as that implies that the FILTER_SPEC refers to a different P2MP BIER LSP.

3.4. SESSION Object

A P2MP BIER LSP SESSION object is used. This object uses the existing SESSION C-Num. New C-Types are defined to accommodate a logical P2MP destination identifier of the P2MP BIER tunnel. This SESSION object has a similar structure as the existing point-to-multipoint RSVP-TE SESSION object. However the C-Types is different. All S2L BIER sub-LSPs that are part of the same P2MP BIER LSP share the same SESSION object. This SESSION object identifies the P2MP BIER tunnel.

The combination of the SESSION object, the SENDER_TEMPLATE object and the S2L_BIER_SUB_LSP object identifies each S2L BIER sub-LSP. This follows the existing P2MP RSVP-TE notion of using the SESSION object for identifying a P2MP Tunnel, which in turn can contain multiple LSPs, each distinguished by a unique SENDER_TEMPLATE object.

3.4.1. P2MP BIER Tunnel IPv4 SESSION Object

Class = SESSION, P2MP_BIER_TUNNEL_IPv4 C-Type = TBD

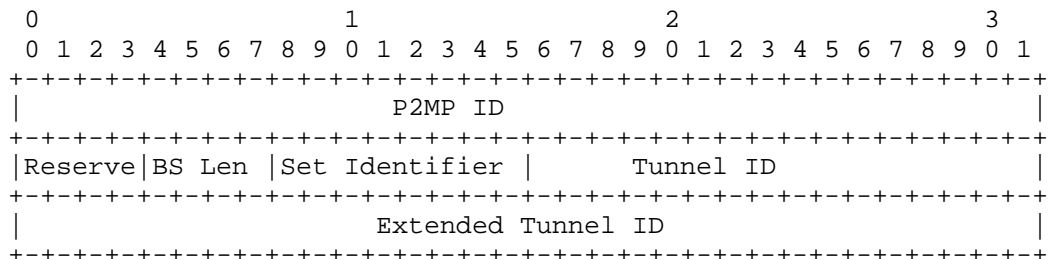


Figure 4: P2MP BIER Tunnel IPv4 SESSION Object

P2MP ID: A 32-bit identifier used in the SESSION object that remains constant over the life of the P2MP BIER tunnel. It encodes the P2MP Identifier that is unique within the scope of the ingress LSR.

BS Len: A 4 bits field encoding the supported BitString length associated with this BFR-prefix. The values allowed in this field are specified in section 2 of [RFC8296].

Set Identifier: A 8 bits fields encoding the Set Identifier (section 1 of [RFC8279])

Tunnel ID: A 16-bit identifier used in the SESSION object that remains constant over the life of the P2MP BIER tunnel.

Extended Tunnel ID: A 32-bit identifier used in the SESSION object that remains constant over the life of the P2MP BIER tunnel. Ingress LSRs that wish to have a globally unique identifier for the P2MP BIER tunnel SHOULD place their tunnel sender address here. A combination of this address, P2MP ID, and Tunnel ID provides a globally unique identifier for the P2MP BIER tunnel.

3.4.2. P2MP BIER Tunnel IPv6 SESSION Object

Class = SESSION, P2MP_BIER_TUNNEL_IPv6 C-Type = TBD

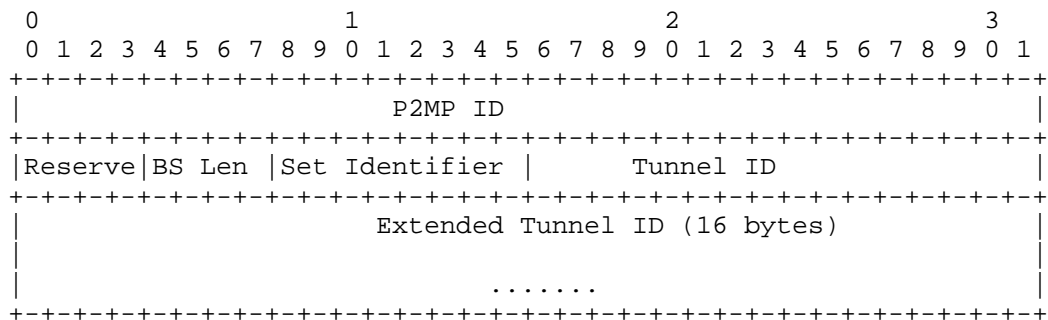


Figure 5: P2MP BIER Tunnel IPv6 SESSION Object

This is the same as the P2MP BIER Tunnel IPv4 LSP SESSION object with the difference that the extended tunnel ID may be set to a 16-byte identifier [RFC3209].

3.5. SENDER_TEMPLATE Object

The SENDER_TEMPLATE object contains the ingress LSR source address. The LSP ID can be changed to allow a sender to share resources with itself. Thus, multiple instances of the P2MP BIER tunnel can be created, each with a different LSP ID. The instances can share resources with each other. The S2L BIER sub-LSPs corresponding to a particular instance use the same LSP ID.

The combination of the SESSION object, the SENDER_TEMPLATE object and the S2L_BIER_SUB_LSP object identifies each S2L BIER sub-LSP. This follows the existing P2MP RSVP-TE notion of using the SESSION object

for identifying a P2MP Tunnel, which in turn can contain multiple LSPs, each distinguished by a unique SENDER TEMPLATE object.

3.5.1. P2MP BIER Tunnel IPv4 SENDER TEMPLATE Object

Class = SENDER TEMPLATE, P2MP BIER TUNNEL IPv4 C-Type = TBD

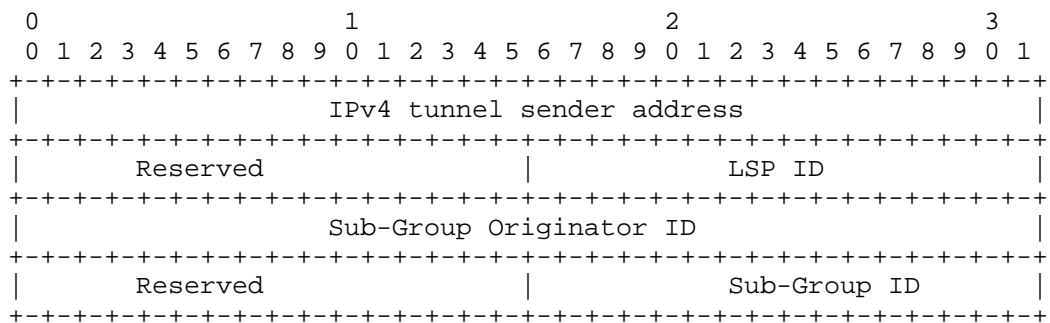


Figure 6: P2MP BIER Tunnel IPv4 SENDER TEMPLATE Object

IPv4 tunnel sender address

See [RFC3209].

Sub-Group Originator ID

The Sub-Group Originator ID is set to the TE Router ID of the LSR that originates the Path message. This is either the ingress LSR or an LSR which re-originates the Path message with its own Sub- Group Originator ID.

Sub-Group ID

An identifier of a Path message used to differentiate multiple Path messages that signal state for the same P2MP BIER LSP. This may be seen as identifying a group of one or more egress nodes targeted by this Path message.

LSP ID

See [RFC3209].

3.5.2. P2MP BIER Tunnel IPv6 SENDER_TEMPLATE Object

Class = SENDER_TEMPLATE, P2MP_BIER_TUNNEL_IPv6 C-Type = TBD

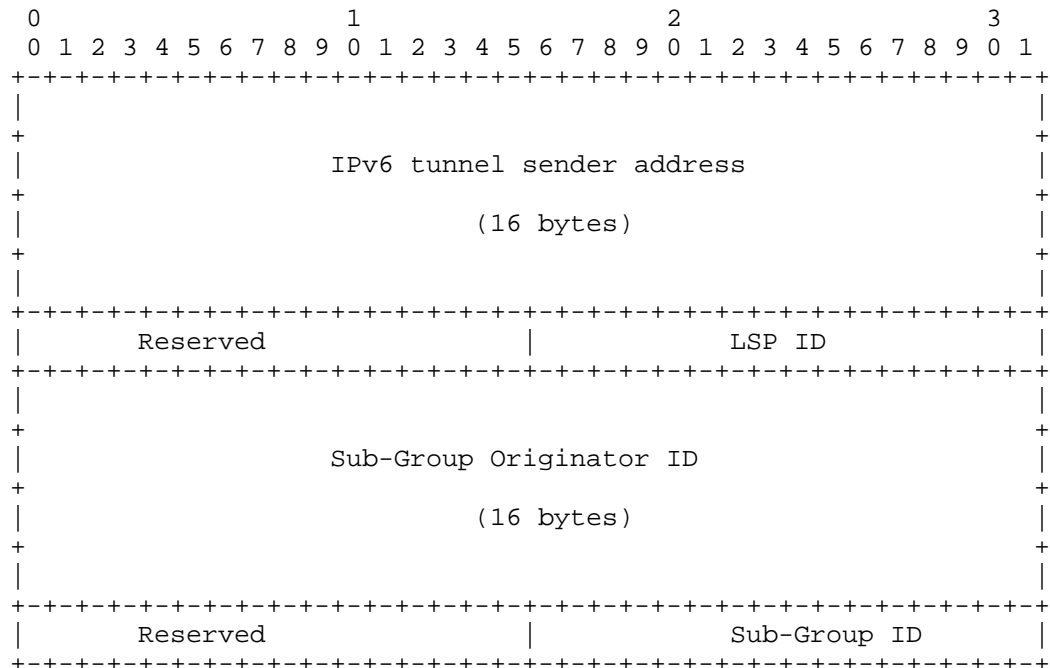


Figure 7: P2MP BIER Tunnel IPv6 SENDER_TEMPLATE Object

This is the same as the P2MP BIER Tunnel IPv4 SENDER_TEMPLATE Object with the difference that the sender address and Sub-Group Originator ID may be set to a 16-byte identifier [RFC3209].

3.6. S2L_BIER_SUB_LSP Object

An S2L_BIER_SUB_LSP object identifies a particular S2L BIER sub-LSP belonging to the P2MP BIER LSP.

3.6.1. S2L_BIER_SUB_LSP IPv4 Object

S2L_BIER_SUB_LSP Class = TBD, S2L_BIER_SUB_LSP_IPv4 C-Type = TBD

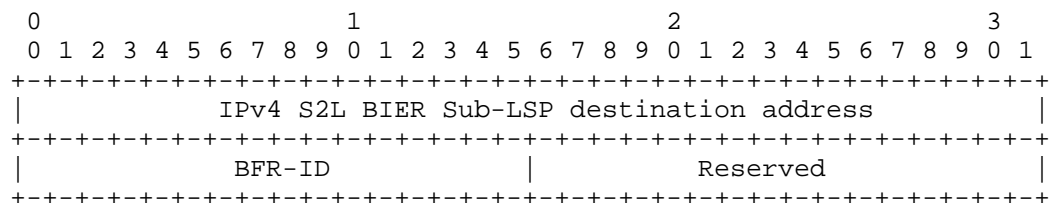


Figure 8: S2L_BIER_SUB_LSP IPv4 Object

5. IANA Considerations

Allocation is expected from IANA for codepoints from the "Class Names, Class Numbers, and Class Types" registry.

6. Security Considerations

TBD

7. Acknowledgements

TBD

8. References

8.1. Normative References

- [I-D.ietf-bier-mvpn]
Rosen, E., Sivakumar, M., Aldrin, S., Dolganow, A., and T. Przygienda, "Multicast VPN Using BIER", draft-ietf-bier-mvpn-11 (work in progress), March 2018.
- [I-D.xie-bier-mvpn-mpls-p2mp]
Xie, J., McBride, M., Chen, M., and L. Geng, "Multicast VPN Using MPLS P2MP and BIER", draft-xie-bier-mvpn-mpls-p2mp-02 (work in progress), July 2018.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <<https://www.rfc-editor.org/info/rfc3473>>.
- [RFC4461] Yasukawa, S., Ed., "Signaling Requirements for Point-to-Multipoint Traffic-Engineered MPLS Label Switched Paths (LSPs)", RFC 4461, DOI 10.17487/RFC4461, April 2006, <<https://www.rfc-editor.org/info/rfc4461>>.

- [RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <<https://www.rfc-editor.org/info/rfc4875>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

8.2. 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>>.

Authors' Addresses

Jingrong Xie
Huawei Technologies
Q15 Huawei Campus, No.156 Beiqing Rd.
Beijing 100095
China

Email: xiejingrong@huawei.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Robin Li
Huawei

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2018

X. Xu
Alibaba
S. Bryant
Huawei
A. Farrel
Juniper
S. Hassan
Cisco
W. Henderickx
Nokia
Z. Li
Huawei
June 1, 2018

SR-MPLS over IP
draft-xu-mpls-sr-over-ip-01

Abstract

MPLS Segment Routing (SR-MPLS in short) is an MPLS data plane-based source routing paradigm in which the sender of a packet is allowed to partially or completely specify the route the packet takes through the network by imposing stacked MPLS labels on the packet. SR-MPLS could be leveraged to realize a source routing mechanism across MPLS, IPv4, and IPv6 data planes by using an MPLS label stack as a source routing instruction set while preserving backward compatibility with SR-MPLS.

This document describes how SR-MPLS capable routers and IP-only routers can seamlessly co-exist and interoperate through the use of SR-MPLS label stacks and IP encapsulation/tunneling such as MPLS-in-UDP as defined in RFC 7510.

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 December 3, 2018.

Copyright Notice

Copyright (c) 2018 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. Use Cases	3
4. Procedures of SR-MPLS over IP	5
4.1. Forwarding Entry Construction	5
4.2. Packet Forwarding Procedures	7
4.2.1. Packet Forwarding with Penultimate Hop Popping	8
4.2.2. Packet Forwarding without Penultimate Hop Popping	9
4.2.3. Additional Forwarding Procedures	10
5. IANA Considerations	11
6. Security Considerations	12
7. Contributors	12
8. Acknowledgements	13
9. References	13
9.1. Normative References	13
9.2. Informative References	15
Authors' Addresses	15

1. Introduction

MPLS Segment Routing (SR-MPLS in short) [I-D.ietf-spring-segment-routing-mpls] is an MPLS data plane-based source routing paradigm in which the sender of a packet is allowed to partially or completely specify the route the packet takes through the network by imposing stacked MPLS labels on the packet. SR-MPLS could be leveraged to realize a source routing mechanism across MPLS, IPv4, and IPv6 data planes by using an MPLS label stack as a source routing instruction set while preserving backward compatibility with SR-MPLS. More specifically, the source routing instruction set

information contained in a source routed packet could be uniformly encoded as an MPLS label stack no matter whether the underlay is IPv4, IPv6, or MPLS.

This document describes how SR-MPLS capable routers and IP-only routers can seamlessly co-exist and interoperate through the use of SR-MPLS label stacks and IP encapsulation/tunneling such as MPLS-in-UDP [RFC7510].

Although the source routing instructions are encoded as MPLS labels, this is a hardware convenience rather than an indication that the whole MPLS protocol stack needs to be deployed. In particular, the MPLS control protocols are not used in this or any other form of SR-MPLS.

Section 3 describes various use cases for the tunneling SR-MPLS over IP. Section 4 describes a typical application scenario and how the packet forwarding happens.

2. Terminology

This memo makes use of the terms defined in [RFC3031] and [I-D.ietf-spring-segment-routing-mpls].

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. Use Cases

Tunneling SR-MPLS using IPv4 and/or IPv6 tunnels is useful at least in the following use cases:

- o Incremental deployment of the SR-MPLS technology may be facilitated by tunneling SR-MPLS packets across parts of a network that are not SR-MPLS enabled using an IP tunneling mechanism such as MPLS-in-UDP [RFC7510]. The tunnel destination address is the address of the next SR-MPLS-capable node along the path (i.e., the egress of the active node segment). This is shown in Figure 1.

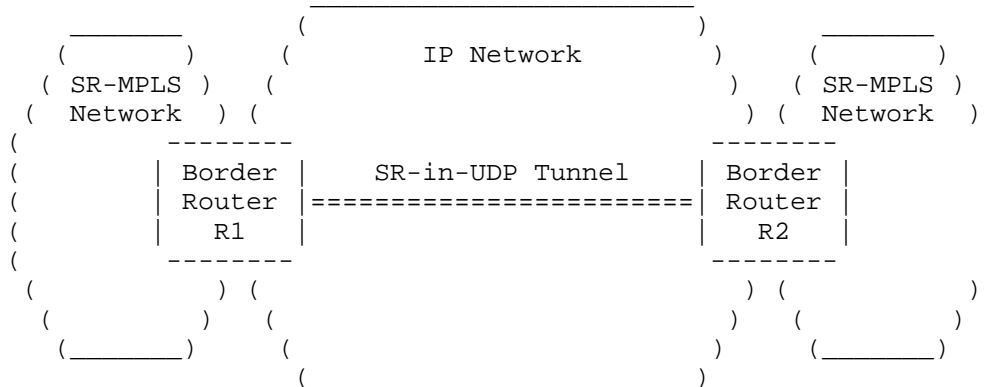
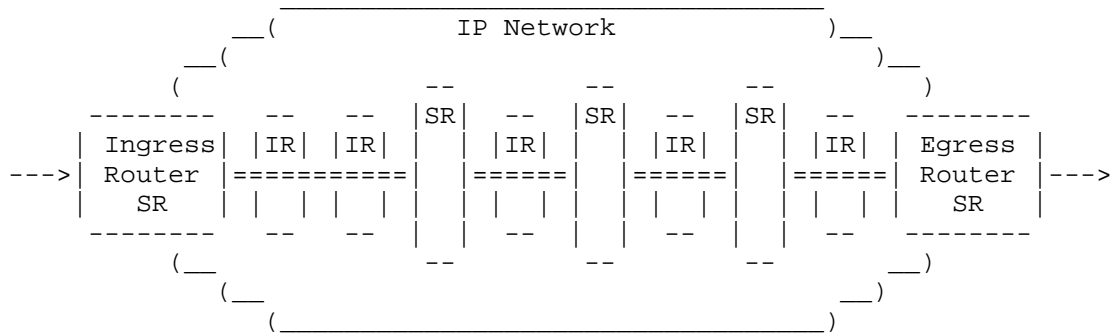


Figure 1: SR-MPLS in UDP to Tunnel Between SR-MPLS Sites

- o If encoding of entropy is desired, IP tunneling mechanisms that allow encoding of entropy, such as MPLS-in-UDP encapsulation [RFC7510] where the source port of the UDP header is used as an entropy field, may be used to maximize the utilization of ECMP and/or UCMP, specially when it is difficult to make use of entropy label mechanism. Refer to [I-D.ietf-mpls-spring-entropy-label]) for more discussion about using entropy label in SR-MPLS.
- o Tunneling MPLS into IP provides a technology that enables SR in an IPv4 and/or IPv6 network where the routers do not support SRv6 capabilities [I-D.ietf-6man-segment-routing-header] and where MPLS forwarding is not an option. This is shown in Figure Figure 2.



Key:

IR : IP-only Router

SR : SR-MPLS-capable Router

== : SR-MPLS in UDP Tunnel

Figure 2: SR-MPLS Enabled Within an IP Network

4. Procedures of SR-MPLS over IP

This section describes the construction of forwarding information base (FIB) entries and the forwarding behavior that allow the deployment of SR-MPLS when some routers in the network are IP only (i.e., do not support SR-MPLS). Note that the examples described in Section 4.1 and Section 4.2 assume that OSPF or ISIS is enabled: in fact, other mechanisms of discovery and advertisement could be used including other routing protocols (such as BGP) or a central controller.

4.1. Forwarding Entry Construction

This sub-section describes the how to construct the forwarding information base (FIB) entry on an SR-MPLS-capable router when some or all of the next-hops along the shortest path towards a prefix-SID are IP-only routers.

Consider router A that receives a labeled packet with top label $L(E)$ that corresponds to the prefix-SID $SID(E)$ of prefix $P(E)$ advertised by router E. Suppose the i th next-hop router (termed NH_i) along the shortest path from router A toward $SID(E)$ is not SR-MPLS capable while both routers A and E are SR-MPLS capable. The following processing steps apply:

- o Router E is SR-MPLS capable so it advertises the SR-Capabilities sub-TLV including the SRGB as described in [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-isis-segment-routing-extensions].
- o Router E advertises the prefix-SID SID(E) of prefix P(E) so MUST also advertise the encapsulation endpoint and the tunnel type of any tunnel used to reach E. It does this using the mechanisms described in [I-D.ietf-isis-encapsulation-cap] or [I-D.ietf-ospf-encapsulation-cap].
- o If A and E are in different IGP areas/levels, then:
 - * The OSPF Tunnel Encapsulation TLV [I-D.ietf-ospf-encapsulation-cap] or the ISIS Tunnel Encapsulation sub-TLV [I-D.ietf-isis-encapsulation-cap] is flooded domain-wide.
 - * The OSPF SID/label range TLV [I-D.ietf-ospf-segment-routing-extensions] or the ISIS SR-Capabilities Sub-TLV [I-D.ietf-isis-segment-routing-extensions] is advertised domain-wide. This way router A knows the characteristics of the router that originated the advertisement of SID(E) (i.e., router E).
 - * When router E advertises the prefix P(E):
 - + If router E is running ISIS it uses the extended reachability TLV (TLVs 135, 235, 236, 237) and associates the IPv4/IPv6 or IPv4/IPv6 source router ID sub-TLV(s) [RFC7794].
 - + If router E is running OSPF it uses the OSPFv2 Extended Prefix Opaque LSA [RFC7684] and sets the flooding scope to AS-wide.
 - * If router E is running ISIS and advertises the ISIS capabilities TLV (TLV 242) [RFC7981], it MUST set the "router-ID" field to a valid value or include an IPV6 TE router-ID sub-TLV (TLV 12), or do both. The "S" bit (flooding scope) of the ISIS capabilities TLV (TLV 242) MUST be set to "1" .
- o Router A programs the FIB entry for prefix P(E) corresponding to the SID(E) as follows:
 - * If the NP flag in OSPF or the P flag in ISIS is clear:

pop the top label

- * If the NP flag in OSPF or the P flag in ISIS is set:
 - swap the top label to a value equal to SID(E) plus the lower bound of the SRGB of E
- * Encapsulate the packet according to the encapsulation advertised in [I-D.ietf-isis-encapsulation-cap] or [I-D.ietf-ospf-encapsulation-cap]
- * Send the packet towards the next hop NHi.

4.2. Packet Forwarding Procedures

[RFC7510] specifies an IP-based encapsulation for MPLS, i.e., MPLS-in-UDP, which is applicable in some circumstances where IP-based encapsulation for MPLS is required and further fine-grained load balancing of MPLS packets over IP networks over Equal-Cost Multipath (ECMP) and/or Link Aggregation Groups (LAGs) is required as well. This section provides details about the forwarding procedure when when UDP encapsulation is adopted for SR-MPLS over IP.

Nodes that are SR-MPLS capable can process SR-MPLS packets. Not all of the nodes in an SR-MPLS domain are SR-MPLS capable. Some nodes may be "legacy routers" that cannot handle SR-MPLS packets but can forward IP packets. An SR-MPLS-capable node may advertise its capabilities using the IGP as described in Section 4. There are six types of node in an SR-MPLS domain:

- o Domain ingress nodes that receive packets and encapsulate them for transmission across the domain. Those packets may be any payload protocol including native IP packets or packets that are already MPLS encapsulated.
- o Legacy transit nodes that are IP routers but that are not SR-MPLS capable (i.e., are not able to perform segment routing).
- o Transit nodes that are SR-MPLS capable but that are not identified by a SID in the SID stack.
- o Transit nodes that are SR-MPLS capable and need to perform SR-MPLS routing because they are identified by a SID in the SID stack.
- o The penultimate SR-MPLS capable node on the path that processes the last SID on the stack on behalf of the domain egress node.
- o The domain egress node that forwards the payload packet for ultimate delivery.

4.2.1. Packet Forwarding with Penultimate Hop Popping

The description in this section assumes that the label associated with each prefix-SID is advertised by the owner of the prefix-SID is a Penultimate Hop Popping (PHP) label. That is, the NP flag in OSPF or the P flag in ISIS associated with the prefix SID is not set.

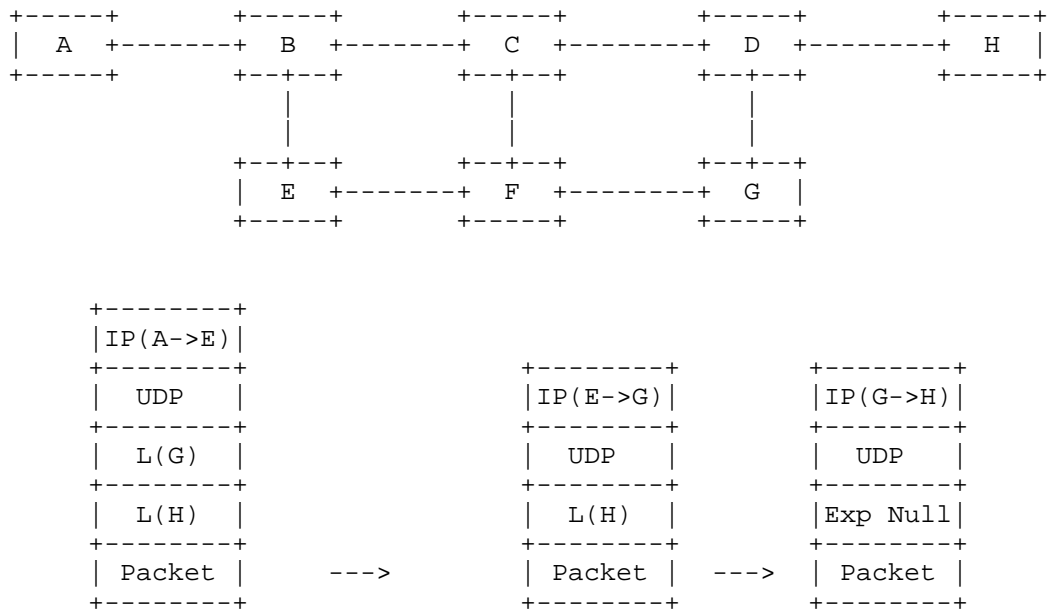


Figure 3: Packet Forwarding Example with PHP

In the example shown in Figure 3, assume that routers A, E, G and H are SR-MPLS-capable while the remaining routers (B, C, D and F) are only capable of forwarding IP packets. Routers A, E, G, and H advertise their Segment Routing related information via IS-IS or OSPF.

Now assume that router A (the Domain ingress) wants to send a packet to router H (the Domain egress) via the explicit path {E->G->H}. Router A will impose an MPLS label stack on the packet that corresponds to that explicit path. Since the next hop toward router E is only IP-capable (B is a legacy transit node), router A replaces the top label (that indicated router E) with a UDP-based tunnel for MPLS (i.e., MPLS-over-UDP [RFC7510]) to router E and then sends the

packet. In other words, router A pops the top label and then encapsulates the MPLS packet in a UDP tunnel to router E.

When the IP-encapsulated MPLS packet arrives at router E (which is an SR-MPLS-capable transit node), router E strips the IP-based tunnel header and then process the decapsulated MPLS packet. The top label indicates that the packet must be forwarded toward router G. Since the next hop toward router G is only IP-capable, router E replaces the current top label with an MPLS-over-UDP tunnel toward router G and sends it out. That is, router E pops the top label and then encapsulates the MPLS packet in a UDP tunnel to router G.

When the packet arrives at router G, router G will strip the IP-based tunnel header and then process the decapsulated MPLS packet. The top label indicates that the packet must be forwarded toward router H. Since the next hop toward router H is only IP-capable (D is a legacy transit router), router G would replace the current top label with an MPLS-over-UDP tunnel toward router H and send it out. However, since router G reaches the bottom of the label stack (G is the penultimate SR-MPLS capable node on the path) this would leave the original packet that router A wanted to send to router H encapsulated in UDP as if it was MPLS (i.e., with a UDP header and destination port indicating MPLS) even though the original packet could have been any protocol. That is, the final SR-MPLS has been popped exposing the payload packet.

To handle this, when a router (here it is router G) pops the final SR-MPLS label, it inserts an explicit null label [RFC3032] before encapsulating the packet in an MPLS-over-UDP tunnel toward router H and sending it out. That is, router G pops the top label, discovers it has reached the bottom of stack, pushes an explicit null label, and then encapsulates the MPLS packet in a UDP tunnel to router H.

4.2.2. Packet Forwarding without Penultimate Hop Popping

Figure 4 demonstrates the packet walk in the case where the label associated with each prefix-SID advertised by the owner of the prefix-SID is not a Penultimate Hop Popping (PHP) label (i.e., the the NP flag in OSPF or the P flag in ISIS associated with the prefix SID is set). Apart from the PHP function the roles of the routers is unchanged from Section 4.2.1.

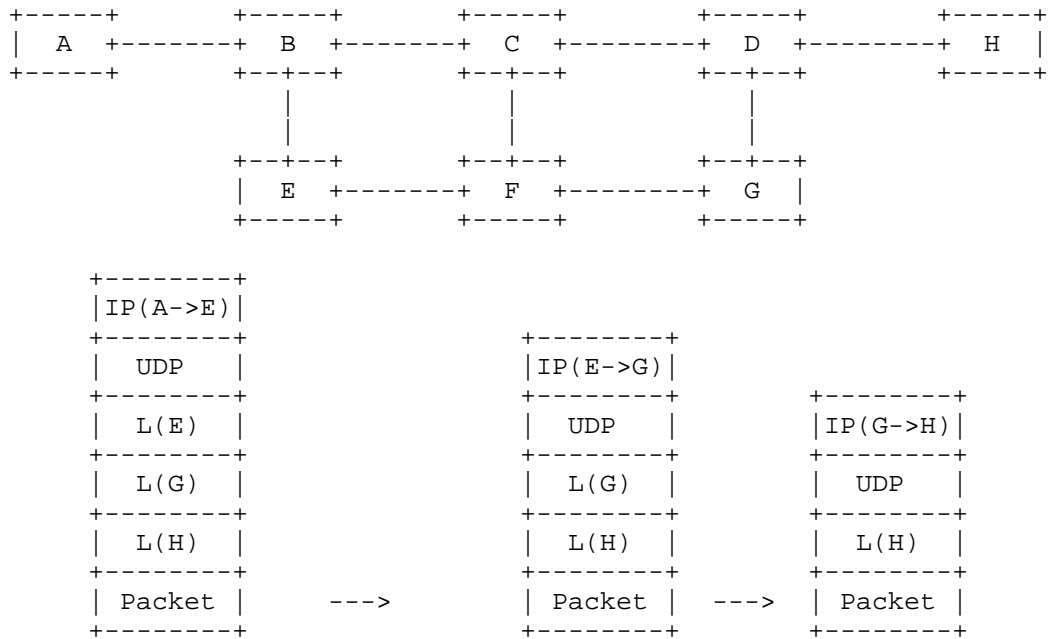


Figure 4: Packet Forwarding Example without PHP

As can be seen from the figure, the SR-MPLS label for each segment is left in place until the end of the segment where it is popped and the next instruction is processed.

4.2.3. Additional Forwarding Procedures

Non-MPLS Interfaces: Although the description in the previous two sections is based on the use of prefix-SIDs, tunneling SR-MPLS packets is useful when the top label of a received SR-MPLS packet indicates an adjacency-SID and the corresponding adjacent node to that adjacency-SID is not capable of MPLS forwarding but can still process SR-MPLS packets. In this scenario the top label would be replaced by an IP tunnel toward that adjacent node and then forwarded over the corresponding link indicated by the adjacency-SID.

When to use IP-based Tunnel: The description in the previous two sections is based on the assumption that MPLS-over-UDP tunnel is used when the nexthop towards the next segment is not MPLS-enabled. However, even in the case where the nexthop towards the next segment is MPLS-capable, an MPLS-over-UDP tunnel towards the

next segment could still be used instead due to local policies. For instance, in the example as described in Figure 4, assume F is now an SR-MPLS-capable transit node while all the other assumptions keep unchanged, since F is not identified by a SID in the stack and an MPLS-over-UDP tunnel is preferred to an MPLS LSP according to local policies, router E would replace the current top label with an MPLS-over-UDP tunnel toward router G and send it out.

IP Header Fields: When encapsulating an MPLS packet in UDP, the resulting packet is further encapsulated in IP for transmission. IPv4 or IPv6 may be used according to the capabilities of the network. The address fields are set as described in Section 3. The other IP header fields (such as DSCP code point, or IPv6 Flow Label) on each UDP-encapsulated segment can be set according to the operator's policy: they may be copied from the header of the incoming packet; they may be promoted from the header of the payload packet; they may be set according to instructions programmed to be associated with the SID; or they may be configured dependent on the outgoing interface and payload.

Entropy and ECMP: When encapsulating an MPLS packet with an IP tunnel header that is capable of encoding entropy (such as [RFC7510]), the corresponding entropy field (the source port in case UDP tunnel) MAY be filled with an entropy value that is generated by the encapsulator to uniquely identify a flow. However, what constitutes a flow is locally determined by the encapsulator. For instance, if the MPLS label stack contains at least one entropy label and the encapsulator is capable of reading that entropy label, the entropy label value could be directly copied to the source port of the UDP header. Otherwise, the encapsulator may have to perform a hash on the whole label stack or the five-tuple of the SR-MPLS payload if the payload is determined as an IP packet. To avoid re-performing the hash or hunting for the entropy label each time the packet is encapsulated in a UDP tunnel it MAY be desirable that the entropy value contained in the incoming packet (i.e., the UDP source port value) is retained when stripping the UDP header and is re-used as the entropy value of the outgoing packet.

5. IANA Considerations

This document makes no requests for IANA action.

6. Security Considerations

The security consideration of [RFC8354] and [RFC7510] apply. DTLS [RFC6347] SHOULD be used where security is needed on an MPLS-SR-over-UDP segment.

It is difficult for an attacker to pass a raw MPLS encoded packet into a network and operators have considerable experience at excluding such packets at the network boundaries.

It is easy for an ingress node to detect any attempt to smuggle an IP packet into the network since it would see that the UDP destination port was set to MPLS. SR packets not having a destination address terminating in the network would be transparently carried and would pose no security risk to the network under consideration.

Where control plane techniques are used (as described in Authors' Addresses) it is important that these protocols are adequately secured for the environment in which they are run.

7. Contributors

Ahmed Bashandy
Individual
Email: abashandy.ietf@gmail.com

Clarence Filsfils
Cisco
Email: cfilsfil@cisco.com

John Drake
Juniper
Email: jdrake@juniper.net

Shaowen Ma
Juniper
Email: mashao@juniper.net

Mach Chen
Huawei
Email: mach.chen@huawei.com

Hamid Assarpour
Broadcom
Email: hamid.assarpour@broadcom.com

Robert Raszuk
Bloomberg LP

Email: robert@raszuk.net

Uma Chunduri
Huawei
Email: uma.chunduri@gmail.com

Luis M. Contreras
Telefonica I+D
Email: luismiguel.contrerasmurillo@telefonica.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

Gunter Van De Velde
Nokia
Email: gunter.van_de_velde@nokia.com

Tal Mizrahi
Marvell
Email: talmi@marvell.com

Jeff Tantsura
Individual
Email: jefftant@gmail.com

8. Acknowledgements

Thanks to Joel Halpern, Bruno Decraene, Loa Andersson, Ron Bonica, Eric Rosen, Jim Guichard, and Gunter Van De Velde for their insightful comments on this draft.

9. References

9.1. Normative References

[I-D.ietf-isis-encapsulation-cap]
Xu, X., Decraene, B., Raszuk, R., Chunduri, U., Contreras, L., and L. Jalil, "Advertising Tunnelling Capability in IS-IS", draft-ietf-isis-encapsulation-cap-01 (work in progress), April 2017.

- [I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-16 (work in progress), April 2018.
- [I-D.ietf-ospf-encapsulation-cap]
Xu, X., Decraene, B., Raszuk, R., Contreras, L., and L. Jalil, "The Tunnel Encapsulations OSPF Router Information", draft-ietf-ospf-encapsulation-cap-09 (work in progress), October 2017.
- [I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-25 (work in progress), April 2018.
- [I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-13 (work in progress), April 2018.
- [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>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [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>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.

- [RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", RFC 7684, DOI 10.17487/RFC7684, November 2015, <<https://www.rfc-editor.org/info/rfc7684>>.
- [RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <<https://www.rfc-editor.org/info/rfc7794>>.
- [RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions for Advertising Router Information", RFC 7981, DOI 10.17487/RFC7981, October 2016, <<https://www.rfc-editor.org/info/rfc7981>>.
- [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>>.

9.2. Informative References

- [I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.
- [I-D.ietf-mpls-spring-entropy-label]
Kini, S., Kompella, K., Sivabalan, S., Litkowski, S., Shakir, R., and J. Tantsura, "Entropy label for SPRING tunnels", draft-ietf-mpls-spring-entropy-label-11 (work in progress), May 2018.
- [RFC8354] Brzozowski, J., Leddy, J., Filsfils, C., Maglione, R., Ed., and M. Townsley, "Use Cases for IPv6 Source Packet Routing in Networking (SPRING)", RFC 8354, DOI 10.17487/RFC8354, March 2018, <<https://www.rfc-editor.org/info/rfc8354>>.

Authors' Addresses

Xiaohu Xu
Alibaba

Email: xiaohu.xxh@alibaba-inc.com

Stewart Bryant
Huawei

Email: stewart.bryant@gmail.com

Adrian Farrel
Juniper

Email: afarrel@juniper.net

Syed Hassan
Cisco

Email: shassan@cisco.com

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2018

L. Zhang
L. Zheng
Huawei Technologies
S. Aldrin
Google
G. Mirsky
ZTE Corp.
October 29, 2017

YANG Data Model for MPLS-TP Operations, Administration, and Maintenance
(OAM)
draft-zhang-mpls-tp-yang-oam-05

Abstract

The Transport Profile of Multiprotocol Label Switching (MPLS-TP), specified in RFC 5921, is a packet-based transport technology based on the MPLS Traffic Engineering (MPLS-TE) and pseudowire (PW) data-plane architectures. A comprehensive set of Operations, Administration, and Maintenance (OAM) procedures that fulfill the MPLS-TP OAM requirements for fault, performance, and protection-switching management had been defined. YANG, defined in RFC 6020 and RFC 7950, is a data model definition language that was introduced to define the content of a conceptual data stores that allows networked devices to be managed using NETCONF, as specified in RFC 6241. This document presents the YANG Data model for MPLS-TP OAM, including the basic functions of Fault Management and Performance Monitoring.

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 May 2, 2018.

Copyright Notice

Copyright (c) 2017 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. Conventions used in this document	3
2.1. Terminology	3
2.2. Requirements Language	4
3. Design of the Data Model	4
3.1. Maintenance Entity Group (MEG) Configuration	4
3.2. Maintenance Entities (MEs) Configuration	4
3.3. MPLS-TP OAM Fault Management And Performance Moinitoring Configuration	6
3.4. Display of ME Status	7
3.5. Display of Detect Result	10
4. MPLS-TP OAM Data Hierarchy	12
5. Interaction with other MPLS OAM Tools Models	17
6. MPLS-TP OAM YANG module	17
7. Examples	47
8. Security Considerations	47
9. IANA Considerations	48
10. Acknowledgements	48
11. References	48
11.1. Normative References	48
11.2. Infomative References	49
Authors' Addresses	49

1. Introduction

The Transport Profile of Multiprotocol Label Switching (MPLS-TP) [RFC5921] is a packet-based transport technology based on the MPLS Traffic Engineering (MPLS-TE) and pseudowire (PW) data-plane architectures. A comprehensive set of Operations, Administration, and Maintenance (OAM) procedures that fulfill the MPLS-TP OAM requirements for fault, performance, and protection-switching

management had been defined. YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document presents the YANG Data model for MPLS-TP OAM, including the basic functions of Fault Management and Performance Monitoring.

The rest of this document is organized as follows. Section 2 presents the conventions used in this document. Section 3 provides the design of the MPLS-TP OAM data model in details. Section 4 presents the complete data hierarchy of LSP-Ping YANG model. Section 5 discusses the interaction between MPLS-TP OAM data model and other MPLS tools data models. Section 6 specifies the YANG module and section 7 lists examples which conform to the YANG module specified in this document. Finally, security considerations are discussed in Section 8.

2. Conventions used in this document

2.1. Terminology

CC - Continuity Check

CV - Conectivity Verification

LM - Loss Measurement

ME - Maintenance Entity

MEG - Maintenance Entity Group

MEP - Maintenance Entity Group End Point

MIP - Maintenance Entity Group Intermediate Point

PM - Performance Monitoring

PW - Pseudowire

DM - Delay Measurement

AIS - Alarm Indication Signal

LKR - Lock Report

2.2. Requirements Language

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. Design of the Data Model

This YANG data model is defined to be used to configure and manage MPLS-TP OAM. Under the top level container `mplstp-oam` is the container `meg`, which contains the configuration and detect result information of multi instances of Maintenance Entity Group (MEG). Under `meg` container, configuration of each Maintenance Entity (ME) type are defined in corresponding list for a particular MEG. Different OAM function configuration are also defined for each MEG. The ME status and detect result information is also shown on per MEG basis. In order to facilitate zero-touch experience, this document defines a default value of the related detect parameters, such as detection intervals, the exp of OAM packet and OAM packet size.

3.1. Maintenance Entity Group (MEG) Configuration

The container "meg" holds the configuration and detect result information of multi instances of Maintenance Entity Group (MEG). Each MEG is indexed by `meg-name`. The data hierarchy for MEG configuration is presented below:

```
module: ietf-mplstp-oam
  +--rw mplstp-oam
    +--rw ais-enable?    enable
    +--rw meg* [meg-name]
      +--rw meg-name          string
      +--rw me-type?         me-type
      +--rw meg-id?          string
      +--rw meg-level?       uint8
      +--rw oam-active-state? active-type
```

3.2. Maintenance Entities (MEs) Configuration

Within a given Maintenance Entity Group there could be one or more type of Maintenance Entity (ME), configuration of different types of MEs are represented in its corresponding list and indexed by its own key. The data hierarchy for ME configuration is presented below:

```
module: ietf-mplstp-oam
```

```

+--rw mplstp-oam
  +--rw ais-enable?  enable
  +--rw meg* [meg-name]
    +--rw meg-name      string
    +--rw me-type?      me-type
    +--rw meg-id?       string
    +--rw meg-level?    uint8
    +--rw oam-active-state?  active-type
    +--rw pw* [local-peer-ip local-vc-id local-vc-type
remote-peer-ip remote-vc-id remote-vc-type]
      +--rw local-peer-ip    inet:ip-address
      +--rw local-vc-id      uint32
      +--rw local-vc-type    vc-type
      +--rw remote-peer-ip   inet:ip-address
      +--rw remote-vc-id     uint32
      +--rw remote-vc-type   vc-type
      +--rw mep-id?         uint16
      +--rw remote-mep-id?   uint16
      +--rw vll-ttl?        uint8
      +--rw gal-enable?     enable
      +--rw gal-mode?       gal-mode
  +--rw lsp* [tunnel-name tunnel-id ingress-lsr-id]
    +--rw tunnel-name      string
    +--rw tunnel-id        uint32
    +--rw ingress-lsr-id   inet:ip-address
    +--rw mep-id?         uint16
    +--rw remote-mep-id?   uint16
    +--rw reverse-tunnel-name  string
    +--rw reverse-tunnel-id?  uint16
    +--rw reverse-ingress-lsr-id?  inet:ip-address
    +--rw tunnel-description?  string
    +--rw tunnel-type?       tunnel-type
    +--rw tunnel-direction?  tunnel-direction-type
  +--rw section* [section-id]
    +--rw section-id      uint64
    +--rw if-name?        string
    +--rw peer-ip         inet:ip-address
    +--rw peer-lsr-id?    inet:ip-address
    +--rw mep-id?         uint16
    +--rw remote-mep-id?   uint16
  +--rw pw-spme* [local-peer-ip local-vc-id switch-peer-ip
switch-vc-id vc-type instance-name]
    +--rw local-peer-ip    inet:ip-address
    +--rw local-vc-id      uint32
    +--rw switch-peer-ip   inet:ip-address
    +--rw switch-vc-id     uint32
    +--rw vc-type          vc-type
    +--rw instance-name    string

```

```

+--rw lsp-spme* [tunnel-id local-lsr-id remote-lsr-id]
|   +--rw tunnel-id          uint32
|   +--rw local-lsr-id       inet:ip-address
|   +--rw remote-lsr-id      inet:ip-address

```

3.3. MPLS-TP OAM Fault Management And Performance Monitoring Configuration

Different OAM function configuration are also defined for each MEG. The data hierarchy for OAM function configuration is presented below:

```

module: ietf-mplstp-oam
+--rw mplstp-oam
|   +--rw ais-enable?        enable
|   +--rw meg* [meg-name]
|   |   +--rw meg-name          string
|   |   +--rw me-type?         me-type
|   |   +--rw meg-id?          string
|   |   +--rw meg-level?       uint8
|   |   +--rw oam-active-state? active-type
|   |   +--rw pw* [local-peer-ip local-vc-id local-vc-type
|   |   remote-peer-ip remote-vc-id remote-vc-type]
|   |   ...
|   |   +--rw lsp* [tunnel-name tunnel-id ingress-lsr-id]
|   |   ...
|   +--rw section* [section-id]
|   |   +--rw section-id        uint64
|   |   +--rw if-name?          string
|   |   +--rw peer-ip           inet:ip-address
|   |   +--rw peer-lsr-id?      inet:ip-address
|   |   +--rw mep-id?           uint16
|   |   +--rw remote-mep-id?    uint16
|   |   +--rw pw-spme* [local-peer-ip local-vc-id switch-peer-ip
|   |   switch-vc-id vc-type instance-name]
|   |   ...
|   |   +--rw lsp-spme* [tunnel-id local-lsr-id remote-lsr-id]
|   |   ...
|   +--rw cc
|   |   +--rw cc-session-mode?   cc-session-mode
|   |   +--rw cc-authentication-enable? enable
|   |   +--rw cc-exp?           uint8
|   |   +--rw cc-transmit-interval? cc-interval
|   |   +--rw cc-recieve-interval? cc-interval
|   |   +--rw cc-detect-multiplier? cc-detect-multiplier
|   |   +--rw cc-enable?        enable
|   +--rw cv
|   |   +--rw cv-session-mode?   cc-session-mode

```

```

|   +---rw cv-authentication-enable?   enable
|   +---rw cv-exp?                     uint8
|   +---rw cv-interval?                cv-interval
|   +---rw cv-detect-multiplier?      cv-detect-multiplier
|   +---rw cv-enable?                 enable
+---rw ais
|   +---rw ais-exp?                    uint8
|   +---rw ais-interval?               ais-interval
+---rw lkr
|   +---rw lkr-exp?                    uint8
|   +---rw lkr-interval?               lkr-interval
|   +---rw lkr-enable?                 enable
+---rw one-way-dm-send
|   +---rw one-dm-send-enable?         enable
|   +---rw one-dm-interval?            dm-interval
|   +---rw one-dm-exp?                 uint8
|   +---rw one-dm-packet-size?         uint16
|   +---rw one-dm-pad-value?           dm-padding-value
+---rw one-way-dm-rcv
|   +---rw onr-dm-rcv-enable?          enable
|   +---rw one-dm-rcv-enable-type?    one-way-rcv-type
+---rw two-way-dm-send
|   +---rw two-dm-send-enable?         enable
|   +---rw two-dm-interval?            dm-interval
|   +---rw two-dm-exp?                 uint8
|   +---rw two-dm-packet-size?         uint16
|   +---rw two-dm-pad-value?           dm-padding-value
|   +---rw two-dm-time-stamp?          enable
+---rw two-way-dm-rcv
|   +---rw two-dm-rcv-enable?          enable
+---rw single-lm-send
|   +---rw slm-send-enable?            enable
|   +---rw slm-interval?               lm-interval
|   +---rw slm-exp?                    uint8
+---rw single-lm-rcv
|   +---rw slm-rcv-enable?             enable
+---rw dual-lm
|   +---rw dlm-enable?                 enable

```

3.4. Display of ME Status

The data hierarchy for display of ME status is presented below:

```

module: ietf-mplstp-oam
+---rw mplstp-oam
|   +---rw ais-enable?   enable
|   +---rw meg* [meg-name]

```

```

+--rw meg-name          string
+--rw me-type?          me-type
+--rw meg-id?           string
+--rw meg-level?        uint8
+--rw oam-active-state? active-type
+--rw pw* [local-peer-ip local-vc-id local-vc-type
remote-peer-ip remote-vc-id remote-vc-type]
...
+--rw lsp* [tunnel-name tunnel-id ingress-lsr-id]
...
+--rw section* [section-id]
...
+--rw pw-spme* [local-peer-ip local-vc-id switch-peer-ip
switch-vc-id vc-type instance-name]
...
+--rw lsp-spme* [tunnel-id local-lsr-id remote-lsr-id]
...
+--rw cc
...
+--rw cv
...
+--rw ais
...
+--rw lkr
...
+--rw one-way-dm-send
...
+--rw one-way-dm-rcv
...
+--rw two-way-dm-send
...
+--rw two-way-dm-rcv
...
+--rw single-lm-send
...
+--rw single-lm-rcv
...
+--rw dual-lm
...
+--ro status-info
|   +--ro pw*
|   |   +--ro me-index?          uint32
|   |   +--ro me-direction?      me-direction
|   |   +--ro me-state?          me-state
|   |   +--ro local-state?       me-state
|   |   +--ro remote-state?      me-state
|   |   +--ro alarm-indicate?    string
|   |   +--ro local-defect-status? defect-status-type

```



```

+---ro local-invalid-time?          uint32
+---ro local-defect-location?       string
+---ro local-defect-type?           defect-type
+---ro remote-defect-status?         defect-status-type
+---ro remote-invalid-time?         uint32
+---ro remote-defect-location?      string
+---ro remote-defect-type?          defect-type
+---ro lsp*
+---ro me-index?                    uint32
+---ro me-direction?                me-direction
+---ro me-state?                    me-state
+---ro local-state?                  me-state
+---ro remote-state?                me-state
+---ro alarm-indicate?              string
+---ro local-defect-status?          defect-status-type
+---ro local-invalid-time?          uint32
+---ro local-defect-location?        string
+---ro local-defect-type?            defect-type
+---ro remote-defect-status?          defect-status-type
+---ro remote-invalid-time?          uint32
+---ro remote-defect-location?        string
+---ro remote-defect-type?            defect-type
+---ro me-index-egress?              uint32
+---ro me-direct-egress?             me-direction
+---ro status-board-egress?          string
+---ro state-egress?                me-state
+---ro alarm-egress?                string
+---ro section*
+---ro me-index?                    uint32
+---ro me-direction?                me-direction
+---ro me-state?                    me-state
+---ro local-state?                  me-state
+---ro remote-state?                me-state
+---ro alarm-indicate?              string
+---ro local-defect-status?          defect-status-type
+---ro local-invalid-time?          uint32
+---ro local-defect-location?        string
+---ro local-defect-type?            defect-type
+---ro remote-defect-status?          defect-status-type
+---ro remote-invalid-time?          uint32
+---ro remote-defect-location?        string
+---ro remote-defect-type?            defect-type
+---ro pw-spme*
+---ro me-index?                    uint32
+---ro me-direction?                me-direction
+---ro me-state?                    me-state
+---ro mip-id?                      uint16
+---ro lsp-spme*

```

	+++ro me-index?	uint32
	+++ro me-direction?	me-direction
	+++ro me-state?	me-state
	+++ro mip-id?	uint16

3.5. Display of Detect Result

The data hierarchy for display of detect result is presented below:

```

module: ietf-mplstp-oam
  +-rw mplstp-oam
    +-rw ais-enable?  enable
    +-rw meg* [meg-name]
      +-rw meg-name      string
      +-rw me-type?     me-type
      +-rw meg-id?      string
      +-rw meg-level?   uint8
      +-rw oam-active-state?  active-type
      +-rw pw* [local-peer-ip local-vc-id local-vc-type
remote-peer-ip remote-vc-id remote-vc-type]
      ...
      +-rw lsp* [tunnel-name tunnel-id ingress-lsr-id]
      ...
      +-rw section* [section-id]
      ...
      +-rw pw-spme* [local-peer-ip local-vc-id switch-peer-ip
switch-vc-id vc-type instance-name]
      ...
      +-rw lsp-spme* [tunnel-id local-lsr-id remote-lsr-id]
      ...
      +-rw cc
      ...
      +-rw cv
      ...
      +-rw ais
      ...
      +-rw lkr
      ...
      +-rw one-way-dm-send
      ...
      +-rw one-way-dm-rcv
      ...
      +-rw two-way-dm-send
      ...
      +-rw two-way-dm-rcv
      ...
      +-rw single-lm-send

```

```

...
+--rw single-lm-rcv
...
+--rw dual-lm
...
+--ro status-info
...
+--ro detect-result
  +--ro one-way-dm-result
    |   +--ro send-pkt-num?      uint32
    |   +--ro rcv-pkt-num?      uint32
    |   +--ro delay-min?        uint32
    |   +--ro delay-max?        uint32
    |   +--ro delay-avg?        uint32
    |   +--ro jitter-min?       uint32
    |   +--ro jitter-max?       uint32
    |   +--ro jitter-avg?       uint32
    |   +--ro one-way-dm-data
    |     +--ro one-way-dm-data* [index]
    |       +--ro index          uint32
    |       +--ro one-delay?     uint32
    |       +--ro one-delay-var? uint32
    |       +--ro error-info?    error-info
    +--ro one-way-send-result
      |   +--ro measure-mode?    measure-mode
      |   +--ro status?          statistics-status
    +--ro two-way-dm-result
      |   +--ro measure-mode?    measure-mode
      |   +--ro status?          statistics-status
      |   +--ro send-pkt-num?    uint32
      |   +--ro rcv-pkt-num?    uint32
      |   +--ro delay-min?      uint32
      |   +--ro delay-max?      uint32
      |   +--ro delay-avg?      uint32
      |   +--ro jitter-min?     uint32
      |   +--ro jitter-max?     uint32
      |   +--ro jitter-avg?     uint32
      |   +--ro two-way-dm-data
      |     +--ro two-way-dm-data* [index]
      |       +--ro index          uint32
      |       +--ro two-delay?     uint32
      |       +--ro two-delay-var? uint32
      |       +--ro error-info?    error-info
    +--ro single-lm-result
      |   +--ro measure-mode?    measure-mode
      |   +--ro status?          statistics-status
      |   +--ro send-pkt-num?    uint32
      |   +--ro rcv-pkt-num?    uint32

```

```

|   +--ro rmt-loss-ratio-min?    uint32
|   +--ro rmt-loss-ratio-max?    uint32
|   +--ro rmt-loss-atio-avg?     uint32
|   +--ro rmt-loss-count-min?    uint32
|   +--ro rmt-loss-count-max?    uint32
|   +--ro rmt-loss-count-avg?    uint32
|   +--ro single-lm-data
|       +--ro single-lm-data* [index]
|           +--ro index          uint32
|           +--ro slm-loss-lcl?   uint32
|           +--ro slm-loss-lcl-rat? string
|           +--ro slm-loss-rmt?   uint32
|           +--ro slm-loss-rmt-rat? string
|           +--ro error-info?     error-info
+--ro dual-lm-data
    +--ro dual-lm-data* [index]
        +--ro index          uint32
        +--ro dlm-loss-lcl?   uint32
        +--ro dlm-loss-lcl-rat? string
        +--ro dlm-loss-rmt?   uint32
        +--ro dlm-loss-rmt-rat? string
        +--ro error-info?     error-info

```

4. MPLS-TP OAM Data Hierarchy

The complete data hierarchy related to the MPLS-TP OAM YANG model is presented below.

```

module: ietf-mplstp-oam
  +--rw mplstp-oam
    +--rw ais-enable?    enable
    +--rw meg* [meg-name]
      +--rw meg-name      string
      +--rw me-type?      me-type
      +--rw meg-id?       string
      +--rw meg-level?    uint8
      +--rw oam-active-state? active-type
      +--rw pw* [local-peer-ip local-vc-id local-vc-type
remote-peer-ip remote-vc-id remote-vc-type]
        +--rw local-peer-ip    inet:ip-address
        +--rw local-vc-id      uint32
        +--rw local-vc-type    vc-type
        +--rw remote-peer-ip   inet:ip-address
        +--rw remote-vc-id     uint32
        +--rw remote-vc-type   vc-type
        +--rw mep-id?         uint16
        +--rw remote-mep-id?   uint16
        +--rw vll-ttl?        uint8

```

```

|   +--rw gal-enable?          enable
|   +--rw gal-mode?           gal-mode
+--rw lsp* [tunnel-name tunnel-id ingress-lsr-id]
|   +--rw tunnel-name          string
|   +--rw tunnel-id            uint32
|   +--rw ingress-lsr-id       inet:ip-address
|   +--rw mep-id?              uint16
|   +--rw remote-mep-id?       uint16
|   +--rw reverse-tunnel-name  string
|   +--rw reverse-tunnel-id?   uint16
|   +--rw reverse-ingress-lsr-id? inet:ip-address
|   +--rw tunnel-description?  string
|   +--rw tunnel-type?         tunnel-type
|   +--rw tunnel-direction?    tunnel-direction-type
+--rw section* [section-id]
|   +--rw section-id           uint64
|   +--rw if-name?             string
|   +--rw peer-ip              inet:ip-address
|   +--rw peer-lsr-id?         inet:ip-address
|   +--rw mep-id?              uint16
|   +--rw remote-mep-id?       uint16
+--rw pw-spme* [local-peer-ip local-vc-id switch-peer-ip
switch-vc-id vc-type instance-name]
|   +--rw local-peer-ip        inet:ip-address
|   +--rw local-vc-id          uint32
|   +--rw switch-peer-ip       inet:ip-address
|   +--rw switch-vc-id         uint32
|   +--rw vc-type              vc-type
|   +--rw instance-name        string
+--rw lsp-spme* [tunnel-id local-lsr-id remote-lsr-id]
|   +--rw tunnel-id            uint32
|   +--rw local-lsr-id         inet:ip-address
|   +--rw remote-lsr-id        inet:ip-address
+--rw cc
|   +--rw cc-session-mode?     cc-session-mode
|   +--rw cc-authentication-enable? enable
|   +--rw cc-exp?              uint8
|   +--rw cc-transmit-interval? cc-interval
|   +--rw cc-recieve-interval? cc-interval
|   +--rw cc-detect-multiplier? cc-detect-multiplier
|   +--rw cc-enable?           enable
+--rw cv
|   +--rw cv-session-mode?     cc-session-mode
|   +--rw cv-authentication-enable? enable
|   +--rw cv-exp?              uint8
|   +--rw cv-interval?         cv-interval
|   +--rw cv-detect-multiplier? cv-detect-multiplier
|   +--rw cv-enable?           enable

```

```

+--rw ais
|   +--rw ais-exp?          uint8
|   +--rw ais-interval?    ais-interval
+--rw lkr
|   +--rw lkr-exp?          uint8
|   +--rw lkr-interval?    lkr-interval
|   +--rw lkr-enable?      enable
+--rw one-way-dm-send
|   +--rw one-dm-send-enable?  enable
|   +--rw one-dm-interval?    dm-interval
|   +--rw one-dm-exp?        uint8
|   +--rw one-dm-packet-size? uint16
|   +--rw one-dm-pad-value?   dm-padding-value
+--rw one-way-dm-rcv
|   +--rw onr-dm-rcv-enable?  enable
|   +--rw one-dm-rcv-enable-type? one-way-rcv-type
+--rw two-way-dm-send
|   +--rw two-dm-send-enable?  enable
|   +--rw two-dm-interval?    dm-interval
|   +--rw two-dm-exp?        uint8
|   +--rw two-dm-packet-size? uint16
|   +--rw two-dm-pad-value?   dm-padding-value
|   +--rw two-dm-time-stamp?  enable
+--rw two-way-dm-rcv
|   +--rw two-dm-rcv-enable?  enable
+--rw single-lm-send
|   +--rw slm-send-enable?    enable
|   +--rw slm-interval?      lm-interval
|   +--rw slm-exp?           uint8
+--rw single-lm-rcv
|   +--rw slm-rcv-enable?    enable
+--rw dual-lm
|   +--rw dlm-enable?        enable
+--ro status-info
|   +--ro pw*
|   |   +--ro me-index?          uint32
|   |   +--ro me-direction?      me-direction
|   |   +--ro me-state?          me-state
|   |   +--ro local-state?       me-state
|   |   +--ro remote-state?      me-state
|   |   +--ro alarm-indicate?    string
|   |   +--ro local-defect-status? defect-status-type
|   |   +--ro local-invalid-time? uint32
|   |   +--ro local-defect-location? string
|   |   +--ro local-defect-type?  defect-type
|   |   +--ro remote-defect-status? defect-status-type
|   |   +--ro remote-invalid-time? uint32
|   |   +--ro remote-defect-location? string

```

```

|   |--ro remote-defect-type?      defect-type
+--ro lsp*
|   |--ro me-index?                uint32
|   |--ro me-direction?            me-direction
|   |--ro me-state?                me-state
|   |--ro local-state?             me-state
|   |--ro remote-state?            me-state
|   |--ro alarm-indicate?          string
|   |--ro local-defect-status?      defect-status-type
|   |--ro local-invalid-time?       uint32
|   |--ro local-defect-location?    string
|   |--ro local-defect-type?        defect-type
|   |--ro remote-defect-status?     defect-status-type
|   |--ro remote-invalid-time?      uint32
|   |--ro remote-defect-location?   string
|   |--ro remote-defect-type?       defect-type
|   |--ro me-index-egress?          uint32
|   |--ro me-direct-egress?         me-direction
|   |--ro status-board-egress?      string
|   |--ro state-egress?             me-state
|   |--ro alarm-egress?             string
+--ro section*
|   |--ro me-index?                uint32
|   |--ro me-direction?            me-direction
|   |--ro me-state?                me-state
|   |--ro local-state?             me-state
|   |--ro remote-state?            me-state
|   |--ro alarm-indicate?          string
|   |--ro local-defect-status?      defect-status-type
|   |--ro local-invalid-time?       uint32
|   |--ro local-defect-location?    string
|   |--ro local-defect-type?        defect-type
|   |--ro remote-defect-status?     defect-status-type
|   |--ro remote-invalid-time?      uint32
|   |--ro remote-defect-location?   string
|   |--ro remote-defect-type?       defect-type
+--ro pw-spme*
|   |--ro me-index?                uint32
|   |--ro me-direction?            me-direction
|   |--ro me-state?                me-state
|   |--ro mip-id?                  uint16
+--ro lsp-spme*
|   |--ro me-index?                uint32
|   |--ro me-direction?            me-direction
|   |--ro me-state?                me-state
|   |--ro mip-id?                  uint16
+--ro detect-result
+--ro one-way-dm-result

```

```

|   +-ro send-pkt-num?          uint32
|   +-ro recv-pkt-num?          uint32
|   +-ro delay-min?             uint32
|   +-ro delay-max?             uint32
|   +-ro delay-avg?             uint32
|   +-ro jitter-min?            uint32
|   +-ro jitter-max?            uint32
|   +-ro jitter-avg?            uint32
|   +-ro one-way-dm-data
|       +-ro one-way-dm-data* [index]
|           +-ro index          uint32
|           +-ro one-delay?      uint32
|           +-ro one-delay-var?  uint32
|           +-ro error-info?     error-info
+--ro one-way-send-result
|   +-ro measure-mode?          measure-mode
|   +-ro status?                statistics-status
+--ro two-way-dm-result
|   +-ro measure-mode?          measure-mode
|   +-ro status?                statistics-status
|   +-ro send-pkt-num?          uint32
|   +-ro rcv-pkt-num?           uint32
|   +-ro delay-min?             uint32
|   +-ro delay-max?             uint32
|   +-ro delay-avg?             uint32
|   +-ro jitter-min?            uint32
|   +-ro jitter-max?            uint32
|   +-ro jitter-avg?            uint32
|   +-ro two-way-dm-data
|       +-ro two-way-dm-data* [index]
|           +-ro index          uint32
|           +-ro two-delay?      uint32
|           +-ro two-delay-var?  uint32
|           +-ro error-info?     error-info
+--ro single-lm-result
|   +-ro measure-mode?          measure-mode
|   +-ro status?                statistics-status
|   +-ro send-pkt-num?          uint32
|   +-ro rcv-pkt-num?           uint32
|   +-ro rmt-loss-ratio-min?     uint32
|   +-ro rmt-loss-ratio-max?     uint32
|   +-ro rmt-loss-atio-avg?      uint32
|   +-ro rmt-loss-count-min?     uint32
|   +-ro rmt-loss-count-max?     uint32
|   +-ro rmt-loss-count-avg?     uint32
|   +-ro single-lm-data
|       +-ro single-lm-data* [index]
|           +-ro index          uint32

```



```

|         +--ro slm-loss-lcl?          uint32
|         +--ro slm-loss-lcl-rat?      string
|         +--ro slm-loss-rmt?          uint32
|         +--ro slm-loss-rmt-rat?      string
|         +--ro error-info?            error-info
+--ro dual-lm-data
  +--ro dual-lm-data* [index]
    +--ro index                        uint32
    +--ro dlm-Loss-lcl?                uint32
    +--ro dlm-loss-lcl-rat?            string
    +--ro dlm-loss-rmt?                uint32
    +--ro dlm-loss-rmt-rat?            string
    +--ro error-info?                  error-info

```

5. Interaction with other MPLS OAM Tools Models

TBA.

6. MPLS-TP OAM YANG module

```

<CODE BEGINS> file "ietf-mplstpoam@2017-10-29.yang"
module ietf-mplstpoam {
  namespace "urn:ietf:params:xml:ns:yang:ietf-mplstpoam";
  //namespace need to be assigned by IANA
  prefix "mplstpoam";
  import ietf-inet-types {
    prefix inet;
  }
  organization "IETF MPLS Working Group";
  contact "draft-zhang-mpls-tp-yang-oam";
  description "MPLS TP OAM YANG Module";
  revision "2017-10-29" {
    description "05 revision";
    reference "draft-zhang-mpls-tp-yang-oam";
  }

  typedef enable {
    type boolean;
    description "enable";
  }

  typedef me-type {
    type enumeration {
      enum "none" {
        value 0;
        description "ME type is not assigned.";
      }
      enum "section" {

```

```
        value 1;
        description "ME type is MPLS-TP Sections (between MPLS
        LSRs).";
    }
    enum "lsp" {
        value 2;
        description "ME type is an end-to-end LSP (between LERs).";
    }
    enum "pw" {
        value 3;
        description "ME type is an end-to-end Single-Segment
        Pseudowire (SS-PW) or MS-PW (between T-PEs).";
    }
    enum "lsp-spme" {
        value 4;
        description "ME type is an SPME (between a given pair
        of LERs and/or LSRs along an LSP).";
    }
    enum "pw-spme" {
        value 5;
        description "ME type is an SPME (between a given pair
        of T-PEs and/or S-PEs along an (MS-)PW).";
    }
}
description "ME type";
}
typedef cc-session-mode {
    type enumeration {
        enum "coordinated" {
            value 0;
            description "coordinated";
        }
        enum "independent" {
            value 1;
            description "independent";
        }
    }
}
description "CC session mode";
}
typedef cc-interval {
    type uint32 {
        range "1..65535";
    }
    description "The value rang for cc packet transmit and receive
    interval.";
}
typedef cv-interval {
    type uint32 {
```

```
        range "1..65535";
    }
    description "The value rang for cv packet transmit interval.";
}
typedef cc-detect-multiplier {
    type uint8{
        range "1..255";
    }
    description "The value rang for cv packet detect multiplier";
}
typedef cv-detect-multiplier {
    type uint8{
        range "1..255";
    }
    description "The value rang for cv packet detect multiplier";
}
typedef lkr-interval {
    type enumeration {
        enum "interval1000ms" {
            value 0;
            description "1000ms";
        }
        enum "interval60000ms" {
            value 1;
            description "60000ms";
        }
    }
    description "lkr-interval";
}
typedef ais-interval {
    type enumeration {
        enum "interval1000ms" {
            value 0;
            description "1000ms";
        }
        enum "interval60000ms" {
            value 1;
            description "60000ms";
        }
    }
    description "ais-interval";
}
typedef me-direction {
    type enumeration {
        enum "ingress" {
            value 0;
            description "The direction to the ME is ingress";
        }
    }
}
```

```
    enum "egress" {
        value 1;
        description "The direction to the ME is egress";
    }
    enum "dual" {
        value 2;
        description "The direction to the ME is dual";
    }
    enum "none" {
        value 3;
        description "The direction to the ME is none";
    }
}
description "me-direction";
}
typedef me-state {
    type enumeration {
        enum "init" {
            value 0;
            description "The me state is init";
        }
        enum "down" {
            value 1;
            description "The me state is down";
        }
        enum "up" {
            value 2;
            description "The me state is up";
        }
    }
    description "me-state";
}
typedef dm-interval {
    type uint32 {
        range "1..65535";
    }
    description "The value rang for dm packet transmit interval";
}
typedef dm-padding-value {
    type enumeration {
        enum "paddingvalue0" {
            value 0;
            description "0";
        }
        enum "paddingvalue1" {
            value 1;
            description "1";
        }
    }
}
```

```
    }
    description "dm-padding-value";
}
typedef lm-interval {
  type uint32 {
    range "1..65535";
  }
  description "The value rang for lm packet transmit interval";
}
typedef measure-mode {
  type enumeration {
    enum "on-demand" {
      value 0;
      description "on-demand";
    }
    enum "proactive" {
      value 1;
      description "proactive";
    }
  }
  description "measure mode";
}
typedef vc-type {
  type uint16 {
    range "1..65535";
  }
  description "The namespace of the vc type of pw";
}
typedef statistics-status {
  type enumeration {
    enum "finished" {
      value 0;
      description "finished";
    }
    enum "working" {
      value 1;
      description "working";
    }
  }
  description "statistics status";
}
typedef error-info {
  type enumeration {
    enum "valid" {
      value 0;
      description "valid";
    }
    enum "invalid-loss" {
```

```
        value 1;
        description "invalid-loss";
    }
    enum "invalid-delay" {
        value 2;
        description "invalid-delay";
    }
}
description "error-info";
}
typedef defect-status-type {
    type string {
        length "1..8191";
    }
    description "The namespace of defect status type";
}
typedef defect-type {
    type string {
        length "1..8191";
    }
    description "The namespace of defect type";
}
typedef tunnel-type {
    type enumeration {
        enum "ingress" {
            value 0;
            description "ingress";
        }
        enum "egress" {
            value 1;
            description "egress";
        }
        enum "bidirectional" {
            value 2;
            description "bidirectional";
        }
    }
    description "tunnel type";
}
typedef tunnel-direction-type {
    type enumeration {
        enum "unidirectional" {
            value 0;
            description "unidirectional";
        }
        enum "bidirectional" {
            value 1;
            description "bidirectional";
        }
    }
}
```

```
    }
  }
  description "tunnel direction type";
}
typedef active-type {
  type enumeration {
    enum "deactive" {
      value 0;
      description "deactive";
    }
    enum "active" {
      value 1;
      description "active";
    }
  }
  description "active-type";
}
typedef gal-mode {
  type enumeration {
    enum "with-13" {
      value 0;
      description "Gal mode is with label 13";
    }
    enum "without-13" {
      value 1;
      description "Gal mode is without label 13";
    }
  }
  description "gal mode";
}
typedef one-way-rcv-type {
  type enumeration {
    enum "on-demand" {
      value 0;
      description "The switch of receive enable takes effect
on-demand one-way delay-measure";
    }
    enum "proactive" {
      value 1;
      description "The switch of receive enable takes effect
proactive one-way delay-measure";
    }
  }
  description "one way receive type";
}
grouping pw {
  leaf local-peer-ip {
    type inet:ip-address;
```

```
        mandatory "true";
        description "This object indicates the peer IP address";
    }
    leaf local-vc-id {
        type uint32 {
            range "1..4294967295";
        }
        mandatory "true";
        description "This object indicates the vc ID of PW
        type ME";
    }
    leaf local-vc-type {
        type vc-type;
        mandatory "true";
        description "This object indicates the vc type of VC
        type ME";
    }
    leaf remote-peer-ip {
        type inet:ip-address;
        description "This object indicates the remote peer IP of
        PW type ME";
    }
    leaf remote-vc-id {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates the remote vc ID of
        PW type ME";
    }
    leaf remote-vc-type {
        type vc-type;
        description "This object indicates the remote vc type of
        PW type ME";
    }
    description "pw";
}
grouping lsp {
    leaf tunnel-name {
        type string {
            length "0..63";
        }
        mandatory "true";
        description "The object indicates the name of tunnel";
    }
    leaf tunnel-id {
        type uint32 {
            range "1..65535";
        }
    }
}
```



```
    description "The object indicates the tunnel id";
  }
  leaf ingress-lsr-id {
    type inet:ip-address;
    description "The object indicates the ingress LSR-ID";
  }
  description "lsp";
}
grouping pw-spme {
  leaf local-peer-ip {
    type inet:ip-address;
    mandatory "true";
    description "This object indicates the peer IP address of
      PW type MIP";
  }
  leaf local-vc-id {
    type uint32 {
      range "1..4294967295";
    }
    mandatory "true";
    description "This object indicates the vc ID of PW type MIP";
  }
  leaf switch-peer-ip {
    type inet:ip-address;
    mandatory "true";
    description "This object indicates the peer IP address of
      PW switch node";
  }
  leaf switch-vc-id {
    type uint32 {
      range "1..4294967295";
    }
    mandatory "true";
    description "This object indicates the vc id of PW switch
      node";
  }
  leaf vc-type {
    type vc-type;
    mandatory "true";
    description "This object indicates the vc type of PW type MIP";
  }
  leaf instance-name {
    type string {
      length "1..31";
    }
    mandatory "true";
    description "This object specifies the VPWS instance name";
  }
}
```

```
    description "pw spme";
}
grouping me-detect-status {
  leaf me-index {
    type uint32 {
      range "1..65535";
    }
    description "The object indicates the index of ME";
  }
  leaf me-direction {
    type me-direction;
    description "The object indicates the direction of ME";
  }
  leaf me-state {
    type me-state;
    description "The object indicates the state of ME";
  }
  leaf local-state {
    type me-state;
    description "The object indicates the local status of ME";
  }
  leaf remote-state {
    type me-state;
    description "The object indicates the remote state of ME";
  }
  leaf alarm-indicate {
    type string {
      length "1..26";
    }
    description "The object indicates the alarm of ME";
  }
  leaf local-defect-status {
    type defect-status-type;
    default "init";
    description "This object indicates the local defect status";
  }
  leaf local-invalid-time {
    type uint32 {
      range "0..4294967295";
    }
    description "This object indicates the invalid Time of
    local detect";
  }
  leaf local-defect-location {
    type string {
      length "1..32";
    }
    description "This object indicates the local defect location";
  }
}
```

```
    }
    leaf local-defect-type {
        type defect-type;
        description "This object indicates the local defect type";
    }
    leaf remote-defect-status {
        type defect-status-type;
        default "init";
        description "This object indicates the remote defect status";
    }
    leaf remote-invalid-time {
        type uint32 {
            range "0..4294967295";
        }
        description "This object indicates the invalid Time of
remote detect";
    }
    leaf remote-defect-location {
        type string {
            length "1..32";
        }
        description "This object indicates the remote defect location";
    }
    leaf remote-defect-type {
        type defect-type;
        description "This object indicates the remote defect type";
    }
    description "This node indicate detect status of ME";
}
grouping gal-set {
    leaf gal-enable {
        type enable;
        default "true";
        description "This object indicates the gal flag";
    }
    leaf gal-mode {
        type gal-mode;
        description "This object indicates the gal flag";
    }
    description "This object indicates the gal set";
}

container mplstp-oam {
    description "Top level container.";
    leaf ais-enable {
        type enable;
        default "false";
        description "This object indicates the global ais flag
```

```
    of mpls-tp oam";
  }

  list meg {
    key "meg-name";
    description "meg multi instances.";
    leaf meg-name {
      type string {
        length "1..14";
      }
      mandatory "true";
      description "The object indicates the name of MEG";
    }
    leaf me-type {
      type me-type;
      default "none";
      description "The object indicates the type of ME";
    }
    leaf meg-id {
      type string {
        length "1..96";
      }
      description "The object indicates the ID of MEG";
    }
    leaf meg-level {
      type uint8 {
        range "0..7";
      }
      default "7";
      description "The object indicates the level of MEG";
    }
    leaf oam-active-state {
      type active-type;
      default "deactive";
      description "This object indicates the oam active state";
    }
  }

  list pw {
    key "local-peer-ip local-vc-id local-vc-type remote-peer-ip
    remote-vc-id remote-vc-type";
    description "PW";
    uses pw;
    leaf mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of local ME";
    }
  }
```

```
    leaf remote-mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of remote
        ME";
    }
    leaf vll-ttl {
      type uint8 {
        range "1..255";
      }
      description "This object indicates the VLL ttl of PW
        type ME";
    }
    uses gal-set;
  }

  list lsp {
    key "tunnel-name tunnel-id ingress-lsr-id";
    description "LSP";
    uses lsp;
    leaf mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of
        local ME";
    }
    leaf remote-mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of
        remote ME";
    }
    leaf reverse-tunnel-name {
      type string {
        length "0..63";
      }
      mandatory "true";
      description "The object indicates the name of
        reverse tunnel";
    }
    leaf reverse-tunnel-id {
      type uint16 {
        range "1..65535";
      }
      description "The object indicates the ingress
```

```
        reverse tunnelId";
    }
    leaf reverse-ingress-lsr-id {
        type inet:ip-address;
        description "The object indicates the ingress
        reverse LSR-ID";
    }
    leaf tunnel-description {
        type string {
            length "1..32";
        }
        description "The object indicates the description
        of tunnel";
    }
    leaf tunnel-type {
        type tunnel-type;
        default "ingress";
        description "The object indicates the type of tunnel";
    }
    leaf tunnel-direction {
        type tunnel-direction-type;
        description "The object indicates the direction of tunnel";
    }
}

list section {
    key "section-id";
    description "Section";
    leaf section-id {
        type uint64 {
            range "1..2147483647";
        }
        description "This object indicates the section ID";
    }
    leaf if-name {
        type string {
            length "1..63";
        }
        description "The object indicates the interface name";
    }
    leaf peer-ip {
        type inet:ip-address;
        mandatory "true";
        description "This object indicates the peer IP address";
    }
    leaf peer-lsr-id {
        type inet:ip-address;
        description "This object indicates the peer lsr ID";
    }
}
```

```
    }
    leaf mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of
        local ME";
    }
    leaf remote-mep-id {
      type uint16 {
        range "1..8191";
      }
      description "This object indicates the MEP Id of
        remote ME";
    }
  }
}

list pw-spme {
  key "local-peer-ip local-vc-id switch-peer-ip
    switch-vc-id vc-type instance-name";
  description "PW-SPME";
  uses pw-spme;
}

list lsp-spme {
  key "tunnel-id local-lsr-id remote-lsr-id";
  description "LSP-SPME";
  leaf tunnel-id {
    type uint32 {
      range "1..65535";
    }
    description "The object indicates the tunnel id";
  }
  leaf local-lsr-id {
    type inet:ip-address;
    description "The object indicates the ingress LSR-ID";
  }
  leaf remote-lsr-id {
    type inet:ip-address;
    description "The object indicates the egress LSR-ID";
  }
}

container cc {
  description "CC";
  leaf cc-session-mode {
    type cc-session-mode;
    default "coordinated";
  }
}
```

```
        description "This object indicates the session
        mode of CC";
    }
    leaf cc-authentication-enable {
        type enable;
        default "true";
        description "CC authentication enable";
    }
    leaf cc-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
        description "This object indicates the exp of CC
        packet which is sent in the MEG";
    }
    leaf cc-transmit-interval {
        type cc-interval;
        default "1";
        description "The interval of CC packet which is
        transmit in the MEG";
    }
    leaf cc-recieve-interval {
        type cc-interval;
        default "1";
        description "The interval of CC packet which is
        recieved in the MEG";
    }
    leaf cc-detect-multiplier {
        type cc-detect-multiplier;
        default "3";
        description "The object indicate the detect
        multiplier of CC packet";
    }
    leaf cc-enable {
        type enable;
        default "true";
        description "The object indicates whether CC can be
        sent by the MEG";
    }
}
container cv {
    description "CV";
    leaf cv-session-mode {
        type cc-session-mode;
        default "coordinated";
        description "This object indicates the session
        mode of CC";
    }
}
```



```
    }
    leaf cv-authentication-enable {
        type enable;
        default "true";
        description "CV authentication enable";
    }
    leaf cv-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
        description "This object indicates the exp of CV packet
            which is sent in the MEG";
    }
    leaf cv-interval {
        type cv-interval;
        default "1";
        description "The interval of CV packet which is sent
            in the MEG";
    }
    leaf cv-detect-multiplier {
        type cv-detect-multiplier;
        default "3";
        description "The object indicate the detect multiplier
            of CV packet";
    }
    leaf cv-enable {
        type enable;
        default "true";
        description "The object indicates whether CC can be
            received by the MEG";
    }
}

container ais {
    description "AIS";
    leaf ais-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
        description "This object indicates the exp of AIS packet
            which is sent in the MEG";
    }
    leaf ais-interval {
        type ais-interval;
        default "interval1000ms";
        description "This object indicates the interval of AIS
```

```
        packet which is sent in the MEG";
    }
}

container lkr {
    description "LKR";
    leaf lkr-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
        description "This object indicates the exp of lock report
        packet which is sent in the MEG";
    }
    leaf lkr-interval {
        type lkr-interval;
        default "interval1000ms";
        description "This object indicates the interval of lock
        report packet which is sent in the MEG";
    }
    leaf lkr-enable {
        type enable;
        default "false";
        description "The object indicates whether lock report
        is enabled in the MEG";
    }
}

container one-way-dm-send {
    description "One way delay measurement send";
    leaf one-dm-send-enable {
        type enable;
        default "false";
        description "This object indicates the 1DM statistics
        is enabled in the MEG";
    }
    leaf one-dm-interval {
        type dm-interval;
        default "1000";
        description "This object indicates the interval of
        1DM statistics in the MEG";
    }
    leaf one-dm-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
    }
}
```

```
        description "This object indicates the exp of 1DM
        packet which is sent in the MEG";
    }
    leaf one-dm-packet-size {
        type uint16 {
            range "64..1518";
        }
        description "This object indicates the packet size
        of 1DM packet which is sent in the MEG";
    }
    leaf one-dm-pad-value {
        type dm-padding-value;
        default "paddingvalue0";
        description "This object indicates the padding value
        of 1DM packet which is sent in the MEG";
    }
}

container one-way-dm-rcv {
    description "One way delay measurement received";
    leaf onr-dm-rcv-enable {
        type enable;
        default "false";
        description "This object indicates the 1DM receive
        is enabled in the MEG";
    }
    leaf one-dm-rcv-enable-type {
        type one-way-rcv-type;
        description "This object indicates the 1DM receive type";
    }
}

container two-way-dm-send {
    description "Two way delay measurement send";
    leaf two-dm-send-enable {
        type enable;
        default "false";
        description "This object indicates the 2DM statistics
        is enabled in the MEG";
    }
    leaf two-dm-interval {
        type dm-interval;
        default "1000";
        description "This object indicates the interval of
        2DM statistics in the MEG";
    }
    leaf two-dm-exp {
        type uint8 {
```

```
        range "0..7";
    }
    default "7";
    description "This object indicates the exp of 2DM
    packet which is sent in the MEG";
}
leaf two-dm-packet-size {
    type uint16 {
        range "64..1518";
    }
    description "This object indicates the packet size of
    2DM packet which is sent in the MEG";
}
leaf two-dm-pad-value {
    type dm-padding-value;
    default "paddingvalue0";
    description "This object indicates the padding value of
    2DM packet which is sent in the MEG";
}
leaf two-dm-time-stamp {
    type enable;
    default "false";
    description "This object indicates whether two-way delay
    measurement time stamp is enable in the MEG";
}
}

container two-way-dm-rcv {
    description "Two way delay measurement recieved";
    leaf two-dm-rcv-enable {
        type enable;
        default "false";
        description "This object indicates the 2DM receiving
        statistics is enabled in the MEG";
    }
}

container single-lm-send {
    description "Single loss measurment send";
    leaf slm-send-enable {
        type enable;
        default "false";
        description "This object indicates whether slm send
        is enable in the MEG";
    }
    leaf slm-interval {
        type lm-interval;
        default "1000";
    }
}
```

```
        description "This object indicates the interval of
        slm statistics in the MEG";
    }
    leaf slm-exp {
        type uint8 {
            range "0..7";
        }
        default "7";
        description "This object indicates the exp of slm
        packet which is sent in the MEG";
    }
}

container single-lm-rcv {
    description "Single loss measurment received";
    leaf slm-rcv-enable {
        type enable;
        default "false";
        description "This object indicates whether slm
        receive is enable in the MEG";
    }
}

container dual-lm {
    description "Dual loss measurement";
    leaf dlm-enable {
        type enable;
        default "false";
        description "This object indicates the dual loss
        statistics is enabled in the MEG";
    }
}

container status-info {
    config "false";
    description "Status info";
    list pw {
        uses me-detect-status;
        description "PW";
    }
    list lsp {
        uses me-detect-status;
        leaf me-index-egress {
            type uint32 {
                range "1..65535";
            }
            description "The object indicates the egress index
            of ME";
        }
    }
}
```

```
    }
    leaf me-direct-egress {
        type me-direction;
        description "The object indicates the direction of
egress ME";
    }
    leaf status-board-egress {
        type string {
            length "1..19";
        }
        description "The object indicates the selected status
board of ME";
    }
    leaf state-egress {
        type me-state;
        description "The object indicates the status of ME";
    }
    leaf alarm-egress {
        type string {
            length "1..26";
        }
        description "The object indicates the alarm of ME";
    }
    description "LSP";
}
list section {
    uses me-detect-status;
    description "Section";
}
list pw-spme {
    leaf me-index {
        type uint32 {
            range "1..65535";
        }
        description "The object indicates the index of MIP";
    }
    leaf me-direction {
        type me-direction;
        description "The object indicates the direction of MIP";
    }
    leaf me-state {
        type me-state;
        description "The object indicates the state of MIP";
    }
    leaf mip-id {
        type uint16 {
            range "1..8191";
        }
    }
}
```

```
        description "The object indicates the ID of MIP";
    }
    description "PW-SPME";
}
list lsp-spme{
    leaf me-index {
        type uint32 {
            range "1..65535";
        }
        description "The object indicates the index of te MIP";
    }
    leaf me-direction {
        type me-direction;
        description "The object indicates the direction of
te MIP";
    }
    leaf me-state {
        type me-state;
        description "The object indicates the state of te MIP";
    }
    leaf mip-id {
        type uint16 {
            range "1..8191";
        }
        description "The object indicates the ID of te MIP";
    }
    description "LSP-SPME";
}
}
container detect-result {
    config "false";
    description "Detect result";
    container one-way-dm-result {
        config "false";
        description "One way delay measurement result";
        leaf send-pkt-num {
            type uint32 {
                range "1..4294967295";
            }
            description "Send packet number";
        }
        leaf rcv-pkt-num {
            type uint32 {
                range "1..4294967295";
            }
            description "Recieved packet number";
        }
        leaf delay-min {
```

```
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the minimum delay
of received LB packets in the MEG";
  }
  leaf delay-max {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the maximum delay
of received LB packets in the MEG";
  }
  leaf delay-avg {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the average delay
of received LB packets in the MEG";
  }
  leaf jitter-min {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the minimum jitter
of received LB packets in the MEG";
  }
  leaf jitter-max {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the average jitter
of received LB packets in the MEG";
  }
  leaf jitter-avg {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the average jitter
of received LB packets in the MEG";
  }
}

container one-way-dm-data {
  config "false";
  description "One way delay measurement data";
  list one-way-dm-data {
    key "index";
    leaf index {
```



```
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates index of 1DM
        statistics record in the MEG";
    }
    leaf one-delay {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates delay of 1DM
        statistics in the MEG";
    }
    leaf one-delay-var {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates delay Variation
        of 1DM statistics in the MEG";
    }
    leaf error-info {
        type error-info;
        description "This object indicates the error info
        of statistics record in the MEG";
    }
    description "One way delay measurement data";
}
}

container one-way-send-result {
    config "false";
    description "One way send result";
    leaf measure-mode {
        type measure-mode;
        default "on-demand";
        description "The flag indicates whether the measurement
        is an on-demand or a continue measurement";
    }
    leaf status {
        type statistics-status;
        default "finished";
        description "The flag indicates whether the measurement
        is finished";
    }
}

container two-way-dm-result {
```

```
config "false";
description "Two way delay measurement result.";
leaf measure-mode {
    type measure-mode;
    default "on-demand";
    description "The flag indicates whether the measurement
        is an on-demand or a continue measurement";
}
leaf status {
    type statistics-status;
    default "finished";
    description "The flag indicates whether the measurement
        is finished";
}
leaf send-pkt-num {
    type uint32 {
        range "1..4294967295";
    }
    description "Send packet number";
}
leaf rcv-pkt-num {
    type uint32 {
        range "1..4294967295";
    }
    description "Received packet number";
}
leaf delay-min {
    type uint32 {
        range "1..4294967295";
    }
    description "This object indicates the minimum delay
        of received LB packets in the MEG";
}
leaf delay-max {
    type uint32 {
        range "1..4294967295";
    }
    description "This object indicates the maximum delay
        of received LB packets in the MEG";
}
leaf delay-avg {
    type uint32 {
        range "1..4294967295";
    }
    description "This object indicates the average delay
        of received LB packets in the MEG";
}
leaf jitter-min {
```

```
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the minimum jitter
of received LB packets in the MEG";
  }
  leaf jitter-max {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the average jitter
of received LB packets in the MEG";
  }
  leaf jitter-avg {
    type uint32 {
      range "1..4294967295";
    }
    description "This object indicates the average jitter
of received LB packets in the MEG";
  }
  container two-way-dm-data {
    config "false";
    description "Two way delay measurement data";
    list two-way-dm-data {
      key "index";
      leaf index {
        type uint32 {
          range "1..4294967295";
        }
        description "This object indicates index of 2DM
statistics record in the MEG";
      }
      leaf two-delay {
        type uint32 {
          range "1..4294967295";
        }
        description "This object indicates delay of 2DM
statistics in the MEG";
      }
      leaf two-delay-var {
        type uint32 {
          range "1..4294967295";
        }
        description "This object indicates delay Variation
of 2DM statistics in the MEG";
      }
      leaf error-info {
        type error-info;
      }
    }
  }
}
```

```
        description "This object indicates the error info
        of statistics record in the MEG";
    }
    description "Two way delay measurement data";
}
}
}

container single-lm-result {
    config "false";
    description "Single loss measurement result.";
    leaf measure-mode {
        type measure-mode;
        default "on-demand";
        description "The flag indicates whether the measurement
        is an on-demand or a continue measurement";
    }
    leaf status {
        type statistics-status;
        default "finished";
        description "The flag indicates whether the measurement
        is finished";
    }
    leaf send-pkt-num {
        type uint32 {
            range "1..4294967295";
        }
        description "Send packet number";
    }
    leaf rcv-pkt-num {
        type uint32 {
            range "1..4294967295";
        }
        description "Received packet number";
    }
    leaf rmt-loss-ratio-min {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates the minimum loss-ratio
        of received LB packets in the MEG";
    }
    leaf rmt-loss-ratio-max {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates the maximum loss-ratio
        of received LB packets in the MEG";
    }
}
```

```
}
leaf rmt-loss-atio-avg {
  type uint32 {
    range "1..4294967295";
  }
  description "This object indicates the average loss-ratio
of received LB packets in the MEG";
}
leaf rmt-loss-count-min {
  type uint32 {
    range "1..4294967295";
  }
  description "This object indicates the minimum packet
lost of received LB packets in the MEG";
}
leaf rmt-loss-count-max {
  type uint32 {
    range "1..4294967295";
  }
  description "This object indicates the average packet
lost of received LB packets in the MEG";
}
leaf rmt-loss-count-avg {
  type uint32 {
    range "1..4294967295";
  }
  description "This object indicates the average packet
lost of received LB packets in the MEG";
}

container single-lm-data {
  config "false";
  description "Single loss measurement data";
  list single-lm-data {
    key "index";
    leaf index {
      type uint32 {
        range "1..4294967295";
      }
      description "This object indicates index of slm
statistics record in the MEG";
    }
    leaf slm-loss-lcl {
      type uint32 {
        range "1..4294967295";
      }
      description "This object indicates local packet
loss of slm statistics in the MEG";
    }
  }
}
```

```
    }
    leaf slm-loss-lcl-rat {
      type string {
        length "1..24";
      }
      description "This object indicates local packet
        loss rate of slm statistics in the MEG";
    }
    leaf slm-loss-rmt {
      type uint32 {
        range "1..4294967295";
      }
      description "This object indicates remote packet
        loss of slm statistics in the MEG";
    }
    leaf slm-loss-rmt-rat {
      type string {
        length "1..24";
      }
      description "This object indicates remote packet
        loss rate of slm statistics in the MEG";
    }
    leaf error-info {
      type error-info;
      description "This object indicates the error
        info of statistics record in the MEG";
    }
    description "Single loss measurement data";
  }
}

container dual-lm-data {
  config "false";
  description "Dula loss measurement data";
  list dual-lm-data {
    key "index";
    leaf index {
      type uint32 {
        range "1..4294967295";
      }
      description "This object indicates index of dlm
        statistics record in the MEG";
    }
    leaf dlm-Loss-lcl {
      type uint32 {
        range "1..4294967295";
      }
    }
  }
}
```

```

        description "This object indicates local packet
        loss of dlm statistics in the MEG";
    }
    leaf dlm-loss-lcl-rat {
        type string {
            length "1..24";
        }
        description "This object indicates local packet
        loss rate of dlm statistics in the MEG";
    }
    leaf dlm-loss-rmt {
        type uint32 {
            range "1..4294967295";
        }
        description "This object indicates remote packet
        loss of dlm statistics in the MEG";
    }
    leaf dlm-loss-rmt-rat {
        type string {
            length "1..24";
        }
        description "This object indicates remote packet
        loss rate of dlm statistics in the MEG";
    }
    leaf error-info {
        type error-info;
        description "This object indicates the error info
        of statistics record in the MEG";
    }
    description "Dual loss measurement data";
}
}
}
}
}
<CODE ENDS>
```

7. Examples

Examples of using YANG module to configure and manage MPLS-TP OAM will be given here in the update.

8. Security Considerations

The configuration and state data defined in this document is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-

implement secure transport is SSH [RFC6242]. The authors recommend to implement the NETCONF access control model [RFC6536] to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content. There are a number of config true nodes defined in the YANG module which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. Write operations to these data nodes without proper protection can have a negative effect on network operations.

9. IANA Considerations

The IANA is requested to as assign a new namespace URI from the IETF XML registry.

URI:TBA

10. Acknowledgements

TBD

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>>.
- [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>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.

[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. Infomative References

[RFC5921] Bocci, M., Ed., Bryant, S., Ed., Frost, D., Ed., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, DOI 10.17487/RFC5921, July 2010, <<https://www.rfc-editor.org/info/rfc5921>>.

Authors' Addresses

Li Zhang
Huawei Technologies
China

Email: monica.zhangli@huawei.com

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Sam K. Aldrin
Google
USA

Email: aldrin.ietf@gmail.com

Greg Mirsky
ZTE Corp.
USA

Email: gregimirsky@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 13, 2019

L. Zheng
G. Zheng
Huawei Technologies
G. Mirsky
ZTE Corp.
R. Rahman
F. Iqbal
Cisco Systems
January 9, 2019

YANG Data Model for LSP-Ping
draft-zheng-mpls-lsp-ping-yang-cfg-10

Abstract

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. RFC 8029 defines a mechanism that would enable users to detect such failure and to isolate faults. YANG, defined in RFC 6020 and RFC 7950, is a data modeling language used to specify the contents of a conceptual data stores that allows networked devices to be managed using NETCONF, as specified in RFC 6241. This document defines a YANG data model that can be used to configure and manage LSP-Ping.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 July 13, 2019.

Copyright Notice

Copyright (c) 2019 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
1.1. Requirements Language	3
1.2. Support of Long Running Command with NETCONF	3
2. Scope	3
3. Design of the Data Model	4
3.1. The Configuration of Control Information	4
3.2. The Configuration of Schedule Parameters	5
3.3. Display of Result Information	6
4. Data Hierarchy	7
5. Interaction with other MPLS OAM Tools Models	9
6. LSP-Ping YANG Module	10
7. Examples	21
7.1. Configuration of Control Information	21
7.2. The Configuration of Schedule Parameters	22
7.3. Display of Result Information	23
8. Security Considerations	25
9. IANA Considerations	26
Contributors	26
Acknowledgments	27
12. References	27
12.1. Normative References	27
12.2. Informative References	27
Authors' Addresses	28

1. Introduction

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. [RFC8029] defines a mechanism that would enable users to detect such failure and to isolate faults. YANG, defined in [RFC6020] and [RFC7950], is a data modeling language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model that can be used to configure and manage LSP-Ping [RFC8029].

The rest of this document is organized as follows. Section 2 presents the scope of this document. Section 3 provides the design of the LSP-Ping configuration data model in details by containers. Section 4 presents the complete data hierarchy of LSP-Ping YANG model. Section 5 discusses the interaction between LSP-Ping data model and other MPLS tools data models. Section 6 specifies the YANG module and section 7 lists examples which conform to the YANG module specified in this document. Finally, security considerations are discussed in Section 8.

This version of the LSP Ping data model conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Requirements Language

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.

1.2. Support of Long Running Command with NETCONF

LSP Ping is one of the examples of what can be described as "long-running operation". Unlike most of the configuration operations that result in single response execution of an LSP Ping triggers multiple responses from a node under control. The question of implementing the long-running operation in NETCONF is still open and possible solutions being discussed:

1. Consecutive Remote Processing Calls (RPC) to poll for results.
2. Model presented in [RFC4560].
3. The one outlined in [I-D.mahesh-netconf-persistent].

The problem of long-running operation as well can be considered as a case of controlling and obtaining results from a Measurement Agent (MA) as defined in [RFC7594].

2. Scope

The fundamental mechanism of LSP-Ping is defined in [RFC8029]. Extensions of LSP-Ping has been developed over the years. There are extensions for performing LSP ping, for example, over P2MP MPLS LSPs [RFC6425] or for Segment Routing IGP Prefix and Adjacency SIDs with an MPLS data plane [RFC8287]. These extensions will be considered in a later update of this document.

3. Design of the Data Model

This YANG data model is defined to be used to configure and manage LSP-Ping and it provides the following features:

1. The configuration of control information of an LSP-Ping test.
2. The configuration of schedule parameters of an LSP-Ping test.
3. Display of result information of an LSP-Ping test.

The top-level container `lsp-pings` holds the configuration of the control information, schedule parameters and result information for multiple instances of LSP-Ping test.

3.1. The Configuration of Control Information

Container `lsp-pings:lsp-ping:control-parameters` defines the configuration parameters which control an LSP-Ping test. Examples are the `target-fec-type/target-fec` of the echo request packet and the `reply mode` of the echo reply packet. Values of some parameters may be auto-assigned by the system, but in several cases, there is a requirement for configuration of these parameters. Examples of such parameters are source address and outgoing interface.

The data hierarchy for control information configuration is presented below:

```

module: ietf-lsp-ping
  +--rw lsp-pings
    +--rw lsp-ping* [lsp-ping-name]
      +--rw lsp-ping-name          string
      +--rw control-parameters
        +--rw target-fec-type?      target-fec-type
        +--rw (target-fec)?
          +--:(ip-prefix)
            +--rw ip-address?        inet:ip-address
          +--:(bgp)
            +--rw bgp?                inet:ip-address
          +--:(rsvp)
            +--rw tunnel-interface?  string
          +--:(vpn)
            +--rw vrf-name?           uint32
            +--rw vpn-ip-address?    inet:ip-address
          +--:(pw)
            +--rw vcid?               uint32
          +--:(vpls)
            +--rw vsi-name?           string
        +--rw traffic-class?        uint8
        +--rw reply-mode?            reply-mode
        +--rw timeout?               uint32
        +--rw timeout-units?         units
        +--rw interval?              uint32
        +--rw interval-units?        units
        +--rw probe-count?           uint32
        +--rw data-size?              uint32
        +--rw data-fill?              string
        +--rw description?            string
        +--rw source-address?         inet:ip-address
        +--rw ttl?                    uint8
        +--rw (outbound)?
          +--:(interface)
            +--rw interface-name?     string
          +--:(nexthop)
            +--rw nexthop?             inet:ip-address

```

3.2. The Configuration of Schedule Parameters

Container `lsp-pings:lsp-ping:scheduling-parameters` defines the schedule parameters of an LSP-Ping test, which describes when to start and when to end the test. Four start modes and three end modes are defined respectively. To be noted that, the configuration of "interval" and "probe-count" parameter defined in container `lsp-pings:lsp-ping:control-parameters` could also determine when the test ends implicitly. All these three parameters are optional. If the user

does not configure either "interval" or "probe-count" parameter, then the default values will be used by the system. If the user configures "end-test", then the actual end time of the LSP-Ping test is the smaller one between the configuration value of "end-test" and the time implicitly determined by the configuration value of "interval"/"probe-count".

The data hierarchy for schedule information configuration is presented below:

```

module: ietf-lsp-ping
  +--rw lsp-pings
    +--rw lsp-ping* [lsp-ping-name]
      +--rw lsp-ping-name          string
      +--rw control-parameters
      ...
      +--rw scheduling-parameters
        +--rw (start-test)?
          +--:(now)
            | +--rw start-test-now?          empty
          +--:(at)
            | +--rw start-test-at?           yang:date-and-time
          +--:(delay)
            | +--rw start-test-delay?        uint32
            | +--rw start-test-delay-units?  units
          +--:(daily)
            | +--rw start-test-daily?        yang:date-and-time
        +--rw (end-test)?
          +--:(at)
            | +--rw end-test-at?             yang:date-and-time
          +--:(delay)
            | +--rw end-test-delay?          uint32
            | +--rw end-test-delay-units?    units
          +--:(lifetime)
            | +--rw end-test-lifetime?       uint32
            | +--rw lifetime-units?         units

```

3.3. Display of Result Information

Container `lsp-pings:lsp-ping:result-info` shows the result of the current LSP-Ping test. Both the statistical result e.g. `min-rtt`, `max-rtt`, and per test probe result e.g. `return code`, `return subcode`, are shown.

The data hierarchy for display of result information is presented below:

```

module: ietf-lsp-ping
  +--rw lsp-pings
    +--rw lsp-ping* [lsp-ping-name]
      +--rw lsp-ping-name          string
      +--rw control-parameters
      ...
      +--rw scheduling-parameters
      ...
      +--ro result-info
        +--ro operational-status?    operational-status
        +--ro source-address?        inet:ip-address
        +--ro target-fec-type?       target-fec-type
        +--ro (target-fec)?
          +--:(ip-prefix)
            | +--ro ip-address?       inet:ip-address
          +--:(bgp)
            | +--ro bgp?              inet:ip-address
          +--:(rsvp)
            | +--ro tunnel-interface? string
          +--:(vpn)
            | +--ro vrf-name?         uint32
            | +--ro vpn-ip-address?   inet:ip-address
          +--:(pw)
            | +--ro vcid?              uint32
          +--:(vpls)
            | +--ro vsi-name?          string
        +--ro min-rtt?                uint32
        +--ro max-rtt?                uint32
        +--ro average-rtt?            uint32
        +--ro probe-responses?        uint32
        +--ro sent-probes?            uint32
        +--ro sum-of-squares?         uint32
        +--ro last-good-probe?        yang:date-and-time
        +--ro probe-results
          +--ro probe-result* [probe-index]
            +--ro probe-index          uint32
            +--ro return-code?         uint8
            +--ro return-sub-code?     uint8
            +--ro rtt?                 uint32
            +--ro result-type?         result-type

```

4. Data Hierarchy

The complete data hierarchy of LSP-Ping YANG model is presented below.

```

module: ietf-lsp-ping

```



```

+--rw lsp-pings
  +--rw lsp-ping* [lsp-ping-name]
    +--rw lsp-ping-name          string
    +--rw control-parameters
      +--rw target-fec-type?      target-fec-type
      +--rw (target-fec)?
        +--:(ip-prefix)
          +--rw ip-address?       inet:ip-address
        +--:(bgp)
          +--rw bgp?              inet:ip-address
        +--:(rsvp)
          +--rw tunnel-interface? string
        +--:(vpn)
          +--rw vrf-name?         uint32
          +--rw vpn-ip-address?   inet:ip-address
        +--:(pw)
          +--rw vcid?             uint32
        +--:(vpls)
          +--rw vsi-name?         string
      +--rw traffic-class?       uint8
      +--rw reply-mode?          reply-mode
      +--rw timeout?             uint32
      +--rw timeout-units?       units
      +--rw interval?            uint32
      +--rw interval-units?      units
      +--rw probe-count?         uint32
      +--rw data-size?           uint32
      +--rw data-fill?           string
      +--rw description?         string
      +--rw source-address?      inet:ip-address
      +--rw ttl?                 uint8
      +--rw (outbound)?
        +--:(interface)
          +--rw interface-name?   string
        +--:(nexthop)
          +--rw nexthop?          inet:ip-address
    +--rw scheduling-parameters
      +--rw (start-test)?
        +--:(now)
          +--rw start-test-now?   empty
        +--:(at)
          +--rw start-test-at?     yang:date-and-time
        +--:(delay)
          +--rw start-test-delay?  uint32
          +--rw start-test-delay-units? units
        +--:(daily)
          +--rw start-test-daily?  yang:date-and-time
      +--rw (end-test)?

```

```

    +---:(at)
    |   +---rw end-test-at?                yang:date-and-time
    +---:(delay)
    |   +---rw end-test-delay?            uint32
    |   +---rw end-test-delay-units?      units
    +---:(lifetime)
    |   +---rw end-test-lifetime?         uint32
    |   +---rw lifetime-units?           units
+---ro result-info
+---ro operational-status?               operational-status
+---ro source-address?                  inet:ip-address
+---ro target-fec-type?                 target-fec-type
+---ro (target-fec)?
|   +---:(ip-prefix)
|   |   +---ro ip-address?               inet:ip-address
+---:(bgp)
|   +---ro bgp?                         inet:ip-address
+---:(rsvp)
|   +---ro tunnel-interface?            string
+---:(vpn)
|   +---ro vrf-name?                    uint32
|   +---ro vpn-ip-address?              inet:ip-address
+---:(pw)
|   +---ro vcid?                        uint32
+---:(vpls)
|   +---ro vsi-name?                    string
+---ro min-rtt?                         uint32
+---ro max-rtt?                         uint32
+---ro average-rtt?                     uint32
+---ro probe-responses?                 uint32
+---ro sent-probes?                     uint32
+---ro sum-of-squares?                  uint32
+---ro last-good-probe?                 yang:date-and-time
+---ro probe-results
    +---ro probe-result* [probe-index]
        +---ro probe-index              uint32
        +---ro return-code?              uint8
        +---ro return-sub-code?          uint8
        +---ro rtt?                      uint32
        +---ro result-type?              result-type

```

5. Interaction with other MPLS OAM Tools Models

TBA

6. LSP-Ping YANG Module

```
<CODE BEGINS> file "ietf-lsp-ping@2018-11-29.yang"
module ietf-lsp-ping {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lsp-ping";
  //namespace need to be assigned by IANA
  prefix "lsp-ping";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Types.";
  }
  import ietf-yang-types{
    prefix yang;
    reference "RFC 6991: Common YANG Types.";
  }

  organization "IETF Multiprotocol Label Switching Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/mppls/
    WG List: mppls@ietf.org

    Editor: Greg Mirsky
      gregimirsky@gmail.com
    Editor: Lianshu Zheng
      vero.zheng@huawei.com
    Editor: Guangying Zheng
      zhengguangying@huawei.com
    Editor: Reshad Rahman
      rrahman@cisco.com
    Editor: Faisal Iqbal
      faiqbal@cisco.com";

  description
    "This YANG module specifies a vendor-independent model
    for the LSP Ping.

    This YANG data model is defined to be used to configure and manage
    LSP-Ping and it provides the following features:
    1. The configuration of control information of an LSP-Ping test.
    2. The configuration of schedule parameters of an LSP-Ping test.
    3. Display of result information of an LSP-Ping test.

    Copyright (c) 2018 IETF Trust and the persons identified as
    the document authors. 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 (<http://trustee.ietf.org/license-info>).

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

```
reference "draft-zheng-mpls-lsp-ping-yang-cfg";

revision "2018-11-29" {
  description
    "10 version, refine the target fec type,
    as per RFC8029 and update Security Considerations section.";
  reference "draft-zheng-mpls-lsp-ping-yang-cfg";
}

typedef target-fec-type {
  type enumeration {
    enum ip-prefix {
      value "0";
      description "IPv4/IPv6 prefix";
    }
    enum bgp {
      value "1";
      description "BGP IPv4/IPv6 prefix";
    }
    enum rsvp {
      value "2";
      description "Tunnel interface";
    }
    enum vpn {
      value "3";
      description "VPN IPv4/IPv6 prefix";
    }
    enum pw {
      value "4";
      description "FEC 128 pseudowire IPv4/IPv6";
    }
    enum vpls {
      value "5";
      description "FEC 129 pseudowire IPv4/IPv6";
    }
  }
  description "Target FEC type, as defined in RFC 8029";
}
```

```
typedef reply-mode {
  type enumeration {
    enum do-not-reply {
      value "1";
      description "Do not reply";
    }
    enum reply-via-udp {
      value "2";
      description "Reply via an IPv4/IPv6 UDP packet";
    }
    enum reply-via-udp-router-alert {
      value "3";
      description
        "Reply via an IPv4/IPv6 UDP packet with Router Alert";
    }
    enum reply-via-control-channel {
      value "4";
      description
        "Reply via application level control channel";
    }
  }
  description "Reply mode";
}

typedef units {
  type enumeration {
    enum seconds {
      description "Seconds";
    }
    enum milliseconds {
      description "Milliseconds";
    }
    enum microseconds {
      description "Microseconds";
    }
    enum nanoseconds {
      description "Nanoseconds";
    }
  }
  description "Time units";
}

typedef operational-status {
  type enumeration {
    enum enabled {
      value "1";
      description "The Test is active";
    }
  }
}
```

```
    enum disabled {
        value "2";
        description "The test has stopped";
    }
    enum completed {
        value "3";
        description "The test is completed";
    }
}
description "Operational state of an LSP Ping test";
}

typedef result-type {
    type enumeration {
        enum success {
            value "1";
            description "The test probe is successful";
        }
        enum fail {
            value "2";
            description "The test probe has failed";
        }
        enum timeout {
            value "3";
            description "The time of the test probe has expired";
        }
    }
    description "Result of each LSP Ping test probe";
}

container lsp-pings {
    description "Multi-instance of the LSP Ping test";
    list lsp-ping {
        key "lsp-ping-name";
        description "LSP Ping test";
        leaf lsp-ping-name {
            type string {
                length "1..31";
            }
            mandatory "true";
            description "LSP Ping test name";
        }
        container control-parameters {
            description "Control information of the LSP Ping test";
            leaf target-fec-type {
                type target-fec-type;
                description "Specifies the address type of the Target FEC";
            }
        }
    }
}
```

```
choice target-fec {
  case ip-prefix {
    leaf ip-address {
      type inet:ip-address;
      description "IPv4/IPv6 Prefix";
    }
  }
  case bgp {
    leaf bgp {
      type inet:ip-address;
      description "BGP IPv4/IPv6 Prefix";
    }
  }
  case rsvp {
    leaf tunnel-interface {
      type string;
      description "Tunnel interface";
    }
  }
  case vpn {
    leaf vrf-name {
      type uint32;
      description "Layer3 VPN Name";
    }
    leaf vpn-ip-address {
      type inet:ip-address;
      description "Layer3 VPN IPv4 Prefix";
    }
  }
  case pw {
    leaf vcid {
      type uint32;
      description "VC ID";
    }
  }
  case vpls {
    leaf vsi-name {
      type string;
      description "VPLS VSI";
    }
  }
  description "Specifies the type of the Target FEC";
}
leaf traffic-class {
  type uint8;
  description "Specifies the Traffic Class";
}
leaf reply-mode {
```

```
    type reply-mode;
    description "Specifies the Reply Mode";
  }
  leaf timeout {
    type uint32;
    description
      "Specifies the time-out value for a LSP Ping operation.";
  }
  leaf timeout-units {
    type units;
    description "Time-out units";
  }
  leaf interval {
    type uint32;
    default 1;
    description
      "Specifies the interval between transmissions
       of LSP Ping echo request packets (probes)
       as part of the LSP Ping test.";
  }
  leaf interval-units {
    type units;
    default seconds;
    description "Interval units";
  }
  leaf probe-count {
    type uint32;
    default 5;
    description
      "Specifies the number of probes sent in the LSP Ping test.";
  }
  leaf data-size {
    type uint32;
    description
      "Specifies the size of the data portion to
       be transmitted in an LSP Ping operation, in octets.";
  }
  leaf data-fill {
    type string{
      length "0..1564";
    }
    description
      "Used together with the corresponding
       data-size value to determine how to fill the data
       portion of a probe packet.";
  }
  leaf description {
    type string{
```



```
        length "1..31";
    }
    description "A descriptive name of the LSP Ping test";
}
leaf source-address {
    type inet:ip-address;
    description "Specifies the source address";
}
leaf ttl {
    type uint8;
    default 255;
    description "Time to live";
}
choice outbound {
    case interface {
        leaf interface-name {
            type string {
                length "1..255";
            }
            description "Specifies the outgoing interface";
        }
    }
    case nexthop {
        leaf nexthop {
            type inet:ip-address;
            description "Specifies the nexthop";
        }
    }
    description "Specifies the out interface or nexthop";
}
}

container scheduling-parameters {
    description "LSP Ping test schedule parameter";
    choice start-test {
        case now {
            leaf start-test-now {
                type empty;
                description "Start test now";
            }
        }
        case at {
            leaf start-test-at {
                type yang:date-and-time;
                description "Start test at a specific time";
            }
        }
        case delay {
```

```
    leaf start-test-delay {
        type uint32;
        description "Start after a specific delay";
    }
    leaf start-test-delay-units {
        type units;
        default seconds;
        description "Delay units";
    }
}
case daily {
    leaf start-test-daily {
        type yang:date-and-time;
        description "Start test daily";
    }
}
description
    "Specifies when the test begins to start,
    include 4 schedule method: start now(1), start at(2),
    start delay(3), start daily(4).";
}

choice end-test{
    case at {
        leaf end-test-at{
            type yang:date-and-time;
            description "End test at a specific time";
        }
    }
    case delay {
        leaf end-test-delay {
            type uint32;
            description "End after a specific delay";
        }
        leaf end-test-delay-units {
            type units;
            default seconds;
            description "Delay units";
        }
    }
}
case lifetime {
    leaf end-test-lifetime {
        type uint32;
        description "Set the test lifetime";
    }
    leaf lifetime-units {
        type units;
        default seconds;
    }
}
```

```
        description "Lifetime units";
    }
}
description
    "Specifies when the test ends, include 3
    schedule method: end at(1), end delay(2),
    end lifetime(3).";
}
}

container result-info {
    config "false";
    description "LSP Ping test result information";
    leaf operational-status {
        type operational-status;
        description "Operational state of a LSP Ping test";
    }
    leaf source-address {
        type inet:ip-address;
        description "The source address of the test";
    }
    leaf target-fec-type {
        type target-fec-type;
        description "The Target FEC address type";
    }
    choice target-fec {
        case ip-prefix {
            leaf ip-address {
                type inet:ip-address;
                description "IPv4/IPv6 Prefix";
            }
        }
        case bgp {
            leaf bgp {
                type inet:ip-address;
                description "BGP IPv4/IPv6 Prefix";
            }
        }
        case rsvp {
            leaf tunnel-interface {
                type string;
                description "Tunnel interface";
            }
        }
        case vpn {
            leaf vrf-name {
                type uint32;
                description "Layer3 VPN Name";
            }
        }
    }
}
```

```
    }
    leaf vpn-ip-address {
        type inet:ip-address;
        description "Layer3 VPN IPv4 Prefix";
    }
}
case pw {
    leaf vcid {
        type uint32;
        description "VC ID";
    }
}
case vpls {
    leaf vsi-name {
        type string;
        description "VPLS VSI";
    }
}
description "The Target FEC address";
}
leaf min-rtt {
    type uint32;
    description
        "The minimum LSP Ping round-trip-time (RTT)
        received measured in usec.";
}
leaf max-rtt {
    type uint32;
    description
        "The maximum LSP Ping round-trip-time (RTT)
        received measured in usec.";
}
leaf average-rtt {
    type uint32;
    description
        "The current average LSP Ping round-trip-time
        (RTT) measured in usec.";
}
leaf probe-responses {
    type uint32;
    description
        "Number of responses received for the
        corresponding LSP Ping test.";
}
leaf sent-probes {
    type uint32;
    description
        "Number of probes sent for the
```

```
        corresponding LSP Ping test.";
    }
    leaf sum-of-squares {
        type uint32;
        description
            "The sum of the squares of RTT,
            calculated as the sum of the squared
            differences between each RTT and the overall
            mean RTT, for all replies received.";
    }
    leaf last-good-probe {
        type yang:date-and-time;
        description
            "Date and time when the last response
            was received for a probe.";
    }
}

container probe-results {
    description "Result info of test probes";
    list probe-result {
        key "probe-index";
        description "Result info of each test probe";
        leaf probe-index {
            type uint32;
            config false;
            description "Probe index";
        }
        leaf return-code {
            type uint8;
            config false;
            description "The Return Code set in the echo reply";
        }
        leaf return-sub-code {
            type uint8;
            config false;
            description
                "The Return Sub-code set in the echo reply.";
        }
        leaf rtt {
            type uint32;
            config false;
            description "The round-trip-time (RTT) received";
        }
        leaf result-type {
            type result-type;
            config false;
            description "The probe result type";
        }
    }
}
```

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

7. Examples

The following examples show the netconf RPC communication between client and server for one LSP-Ping test case.

7.1. Configuration of Control Information

Configure the control-parameters for sample-test-case.

Request from netconf client:

```
<rpc
  message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <control-parameters>
            <target-fec-type>ip-prefix</target-fec-type>
            <ip-prefix>2001:db8::1:100/64</ip-prefix>
            <reply-mode>reply-via-udp</reply-mode>
            <timeout>1</timeout>
            <timeout-units>seconds</timeout-units>
            <interval>1</interval>
            <interval-units>seconds</interval-units>
            <probe-count>6</probe-count>
            <admin-status>enabled</admin-status>
            <data-size>64</data-size>
            <data-fill>this is a lsp ping test</data-fill>
            <source-address>2001:db8::4</source-address>
            <ttl>56</ttl>
          </control-parameters>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>
```

Reply from netconf server:

```
<rpc-reply
  message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.2. The Configuration of Schedule Parameters

Set the scheduling-parameters for sample-test-case to start the test.

Request from netconf client:

```
<rpc
  message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <scheduling-parameters>
            <start-test-now/>
          </scheduling-parameters>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>
```

Reply from netconf server:

```
<rpc-reply
  message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.3. Display of Result Information

Get the result-info of sample-test-case.

Request from netconf client:

```
<rpc
  message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <result-info/>
        </lsp-ping>
      </lsp-pings>
    </filter>
  </get>
</rpc>
```

Reply from netconf server:

```
<rpc-reply
  message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```



```
<data>
  <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
    <lsp-ping>
      <lsp-ping-name>sample-test-case</lsp-ping-name>
      <result-info>
        <operational-status>completed</operational-status>
        <source-address>2001:db8::4</source-address>
        <target-fec-type>ip-prefix</target-fec-type>
        <ip-prefix>2001:db8::1:100/64</ip-prefix>
        <min-rtt>10</min-rtt>
        <max-rtt>56</max-rtt>
        <average-rtt>36</average-rtt>
        <probe-responses>6</probe-responses>
        <sent-probes>6</sent-probes>
        <sum-of-squares>8882</sum-of-squares>
        <last-good-probe>2015-07-01T10:36:56</last-good-probe>
        <probe-results>
          <probe-result>
            <probe-index>0</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>10</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>1</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>56</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>2</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>35</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>3</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>38</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>4</probe-index>
            <return-code>0</return-code>
          </probe-result>
        </probe-results>
      </result-info>
    </lsp-ping>
  </lsp-pings>
```

```
        <return-sub-code>3</return-sub-code>
        <rtt>36</rtt>
        <result-type>success</result-type>
    </probe-result>
    <probe-result>
        <probe-index>5</probe-index>
        <return-code>0</return-code>
        <return-sub-code>3</return-sub-code>
        <rtt>41</rtt>
        <result-type>success</result-type>
    </probe-result>
</probe-results>
</result-info>
</lsp-ping>
</lsp-pings>
</data>
</rpc-reply>
```

8. 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 NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured 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 an adverse effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

TBD

Unauthorized access to any data node of these subtrees can adversely affect the routing subsystem of both the local device and the network. This may lead to corruption of the measurement that may result in false corrective action, e.g., false negative or false positive. That could be, for example, prolonged and undetected

deterioration of the quality of service or actions to improve the quality unwarranted by the real network conditions.

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:

TBD

Unauthorized access to any data node of these subtrees can disclose the operational state information of VRRP on this device.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

TBD

The LSP ping YANG module inherits all security consideration of [RFC8029].

9. IANA Considerations

The IANA is requested to assign a new namespace URI from the IETF XML registry.

URI:TBA

Contributors

Yanfeng Zhang

Huawei Technologies

zhangyanfeng@huawei.com

Sam Aldrin

Google

aldrin.ietf@gmail.com

Acknowledgments

TBD

12. References

12.1. 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>>.
- [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>>.
- [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>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [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>>.

12.2. Informative References

- [I-D.mahesh-netconf-persistent] Jethanandani, M., "NETCONF and persistent responses", draft-mahesh-netconf-persistent-00 (work in progress), October 2014.
- [RFC4560] Quittek, J., Ed. and K. White, Ed., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 4560, DOI 10.17487/RFC4560, June 2006, <<https://www.rfc-editor.org/info/rfc4560>>.
- [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>>.
- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <<https://www.rfc-editor.org/info/rfc7594>>.
- [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>>.
- [RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <<https://www.rfc-editor.org/info/rfc8287>>.
- [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>>.

Authors' Addresses

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Guangying Zheng
Huawei Technologies
China

Email: zhengguangying@huawei.com

Greg Mirsky
ZTE Corp.
USA

Email: gregimirsky@gmail.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com

Faisal Iqbal
Cisco Systems

Email: faiqbal@cisco.com