

Network Working Group  
Internet Draft  
Intended status: Standards Track

Chandra Ramachandran  
Juniper Networks  
Ina Minei  
Google, Inc  
Dante Pacella  
Verizon  
Tarek Saad  
Cisco Systems Inc.

Expires: November 7, 2016

May 7, 2016

Refresh Interval Independent FRR Facility Protection  
draft-chandra-mpls-ri-rsvp-frr-04

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on November 7, 2016.

Copyright Notice

Copyright (c) 2016 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

(<http://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.

## Abstract

RSVP-TE relies on periodic refresh of RSVP messages to synchronize and maintain the LSP related states along the reserved path. In the absence of refresh messages, the LSP related states are automatically deleted. Reliance on periodic refreshes and refresh timeouts are problematic from the scalability point of view. The number of RSVP-TE LSPs that a router needs to maintain has been growing in service provider networks and the implementations should be capable of handling increase in LSP scale.

RFC 2961 specifies mechanisms to eliminate the reliance on periodic refresh and refresh timeout of RSVP messages, and enables a router to increase the message refresh interval to values much larger than the default 30 seconds defined in RFC 2205. However, the protocol extensions defined in RFC 4090 for supporting fast reroute (FRR) using bypass tunnels implicitly rely on short refresh timeouts to cleanup stale states.

In order to eliminate the reliance on refresh timeouts, the routers should unambiguously determine when a particular LSP state should be deleted. Coupling LSP state with the corresponding RSVP-TE signaling adjacencies as recommended in RSVP-TE Scaling Recommendations (draft-ietf-teas-rsvp-te-scaling-rec) will apply in scenarios other than RFC 4090 FRR using bypass tunnels. In scenarios involving RFC 4090 FRR using bypass tunnels, additional explicit tear down messages are necessary. Refresh-interval Independent RSVP FRR (RI-RSVP-FRR) extensions specified in this document consists of procedures to enable LSP state cleanup that are essential in scenarios not covered by procedures defined in RSVP-TE Scaling Recommendations.

## Requirements Language

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

## Table of Contents

1. Introduction.....	4
1.1. Motivation.....	4
2. Terminology.....	5
3. Problem Description.....	5
4. Solution Aspects.....	8
4.1. Signaling Handshake between PLR and MP.....	8
4.1.1. PLR Behavior.....	8
4.1.2. Remote Signaling Adjacency.....	10
4.1.3. MP Behavior.....	10
4.1.4. "Remote" state on MP.....	10
4.2. Impact of Failures on LSP State.....	11
4.2.1. Non-MP Behavior.....	12
4.2.2. LP-MP Behavior.....	12
4.2.3. NP-MP Behavior.....	12
4.2.4. Behavior of a Router that is both LP-MP and NP-MP...	13
4.3. Conditional Path Tear.....	14
4.3.1. Sending Conditional Path Tear.....	14
4.3.2. Processing Conditional Path Tear.....	14
4.3.3. CONDITIONS object.....	15
4.4. Remote State Teardown.....	16
4.4.1. PLR Behavior on Local Repair Failure.....	16
4.4.2. PLR Behavior on Resv RRO Change.....	17
4.4.3. LSP Preemption during Local Repair.....	17
4.4.3.1. Preemption on LP-MP after Phop Link failure....	17
4.4.3.2. Preemption on NP-MP after Phop Link failure....	18
4.5. Backward Compatibility Procedures.....	18
4.5.1. Detecting Support for Refresh interval Independent FRR	19
.....	
4.5.2. Procedures for backward compatibility.....	19
4.5.2.1. Lack of support on Downstream Node.....	19
4.5.2.2. Lack of support on Upstream Node.....	20
4.5.2.3. Incremental Deployment.....	20
5. Security Considerations.....	21
6. IANA Considerations.....	22
6.1. New Object - CONDITIONS.....	22
7. Normative References.....	22
8. Informative References.....	23
9. Acknowledgments.....	23
10. Contributors.....	23
11. Authors' Addresses.....	24

## 1. Introduction

RSVP-TE Fast Reroute [RFC4090] defines two local repair techniques to reroute label switched path (LSP) traffic over pre-established backup tunnel. Facility backup method allows one or more LSPs traversing a connected link or node to be protected using a bypass tunnel. The many-to-one nature of local repair technique is attractive from scalability point of view. This document enumerates facility backup procedures in RFC 4090 that rely on refresh timeout and hence make facility backup method refresh-interval dependent. The RSVP-TE extensions defined in this document will enhance the facility backup protection mechanism by making the corresponding procedures refresh-interval independent.

### 1.1. Motivation

Standard RSVP [RFC2205] maintains state via the generation of RSVP Path/Resv refresh messages. Refresh messages are used to both synchronize state between RSVP neighbors and to recover from lost RSVP messages. The use of Refresh messages to cover many possible failures has resulted in a number of operational problems.

- One problem relates to RSVP control plane scaling due to periodic refreshes of Path and Resv messages, another relates to the reliability and latency of RSVP signaling.
- An additional problem is the time to clean up the stale state after a tear message is lost. For more on these problems see Section 1 of RSVP Refresh Overhead Reduction Extensions [RFC2961].

The problems listed above adversely affect RSVP control plane scalability and RSVP-TE [RFC3209] inherited these problems from standard RSVP. Procedures specified in [RFC2961] address the above mentioned problems by eliminating dependency on refreshes for state synchronization and for recovering from lost RSVP messages, and by eliminating dependency on refresh timeout for stale state cleanup. Implementing these procedures allows to improve RSVP-TE control plane scalability. For more details on eliminating dependency on refresh timeout for stale state cleanup, refer to "Refresh Interval Independent RSVP" section in [TE-SCALE-REC].

However, the procedures specified in [RFC2961] do not fully address stale state cleanup for facility backup protection [RFC4090], as facility backup protection still depends on refresh timeouts for stale state cleanup. Thus [RFC2961] is insufficient to address the

problem of stale state cleanup when facility backup protection is used.

The procedures specified in this document, in combination with [RFC2961], eliminate facility backup protection dependency on refresh timeouts for stale state cleanup. These procedures, in combination with [RFC2961], fully address the above mentioned problem of RSVP-TE stale state cleanup, including the cleanup for facility backup protection.

The procedures specified in this document assume reliable delivery of RSVP messages, as specified in [RFC2961]. Therefore this document makes support for [RFC2961] a pre-requisite.

## 2. Terminology

The reader is assumed to be familiar with the terminology in [RFC2205], [RFC3209], [RFC4090] and [RFC4558].

Phop node: Previous-hop router along the label switched path

PPhop node: Previous-Previous-hop router along the LSP

LP-MP node: Merge Point router at the tail of Link-protecting bypass tunnel

NP-MP node: Merger Point router at the tail of Node-protecting bypass tunnel

TED: Traffic Engineering Database

Conditional PathTear: PathTear message containing a suggestion to a receiving downstream router to retain Path state if the receiving router is NP-MP

Remote PathTear: PathTear message sent from Point of Local Repair (PLR) to MP to delete state on MP if PLR had not reliably sent backup Path state before

## 3. Problem Description

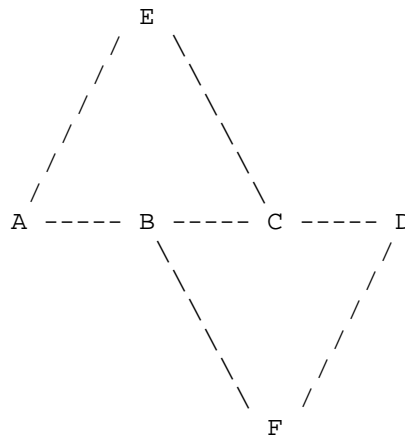


Figure 1: Example Topology

In the topology in Figure 1, consider a large number of LSPs from A to D transiting B and C. Assume that refresh interval has been configured to be large of the order of minutes and refresh reduction extensions are enabled on all routers.

Also assume that node protection has been configured for the LSPs and the LSPs are protected by each router in the following way

- A has made node protection available using bypass LSP A -> E -> C; A is the Point of Local Repair (PLR) and C is Node Protecting Merge Point (NP-MP)
- B has made node protection available using bypass LSP B -> F -> D; B is the PLR and D is the NP-MP
- C has made link protection available using bypass LSP C -> B -> F -> D; C is the PLR and D is the Link Protecting Merge Point (LP-MP)

In the above condition, assume that B-C link fails. The following is the sequence of events that is expected to occur for all protected LSPs under normal conditions.

1. B performs local repair and re-directs LSP traffic over the bypass LSP B -> F -> D.
2. B also creates backup state for the LSP and triggers sending of backup LSP state to D over the bypass LSP B -> F -> D.

3. D receives backup LSP states and merges the backups with the protected LSPs.
4. As the link on C, over which the LSP states are refreshed has failed, C will no longer receive state refreshes. Consequently the protected LSP states on C will time out and C will send tear down message for all LSPs. As each router should consider itself as a Merge Point, C will time out the state only after waiting for an additional duration equal to refresh timeout.

While the above sequence of events has been described in [RFC4090], there are a few problems for which no mechanism has been specified explicitly.

- If the protected LSP on C times out before D receives signaling for the backup LSP, then D would receive PathTear from C prior to receiving signaling for the backup LSP, thus resulting in deleting the LSP state. This would be possible at scale even with default refresh time.
- If upon the link failure C is to keep state until its timeout, then with long refresh interval this may result in a large amount of stale state on C. Alternatively, if upon the link failure C is to delete the state and send PathTear to D, this would result in deleting the state on D, thus deleting the LSP. D needs a reliable mechanism to determine whether it is MP or not to overcome this problem.
- If head-end A attempts to tear down LSP after step 1 but before step 2 of the above sequence, then B may receive the tear down message before step 2 and delete the LSP state from its state database. If B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.
- If B fails to perform local repair in step 1, then B will delete the LSP state from its state database without informing D. As B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.

The purpose of this document is to provide solutions to the above problems which will then make it practical to scale up to a large number of protected LSPs in the network.

## 4. Solution Aspects

The solution consists of five parts.

- Utilize MP determination mechanism specified in [SUMMARY-FRR] that enables the PLR to signal availability of local protection to MP. In addition, introduce PLR and MP procedures to establish Node-ID hello session between the PLR and the MP to detect router failures and to determine capability. See section 4.1 for more details. This part of the solution re-uses some of the extensions defined in [SUMMARY-FRR] and [TE-SCALE-REC], and the subsequent sub-sections will list the extensions in these drafts that are utilized in this document.
- Handle upstream link or node failures by cleaning up LSP states if the node has not found itself as MP through the MP determination mechanism. See section 4.2 for more details.

The combination of "path state" maintained as Path State Block (PSB) and "reservation state" maintained as Reservation State Block (RSB) forms an individual LSP state on an RSVP-TE speaker.

- Introduce extensions to enable a router to send tear down message to downstream router that enables the receiving router to conditionally delete its local state. See section 4.3 for more details.
- Enhance facility protection by allowing a PLR to directly send tear down message to MP without requiring the PLR to either have a working bypass LSP or have already signaled backup LSP state. See section 4.4 for more details.
- Introduce extensions to enable the above procedures to be backward compatible with routers along the LSP path running implementation that do not support these procedures. See section 4.5 for more details.

### 4.1. Signaling Handshake between PLR and MP

#### 4.1.1. PLR Behavior

As per the procedures specified in RFC 4090, when a protected LSP comes up and if the "local protection desired" flag is set in the SESSION\_ATTRIBUTE object, each node along the LSP path attempts to make local protection available for the LSP.



- If the "node protection desired" flag is set, then the node tries to become a PLR by attempting to create a NP-bypass LSP to the NNhop node avoiding the Nhop node on protected LSP path. In case node protection could not be made available after some time out, the node attempts to create a LP-bypass LSP to Nhop node avoiding only the link that protected LSP takes to reach Nhop
- If the "node protection desired" flag is not set, then the PLR attempts to create a LP-bypass LSP to Nhop node avoiding the link that the protected LSP takes to reach Nhop

With regard to the PLR procedures described above and that are specified in RFC 4090, this document specifies the following additional procedures.

- While selecting the destination address of the bypass LSP, the PLR SHOULD attempt to select the router ID of the NNhop or Nhop node. If the PLR and the MP are in same area, then the PLR may utilize the TED to determine the router ID from the interface address in RRO (if NodeID is not included in RRO). If the PLR and the MP are in different IGP areas, then the PLR SHOULD use the NodeID address of NNhop MP if included in the RRO of RESV. If the NP-MP in a different area has not included NodeID in RRO, then the PLR SHOULD use NP-MP's interface address present in the RRO. The PLR SHOULD use its router ID as the source address of the bypass LSP. The PLR SHOULD also include its router ID as the NodeID in PATH RRO unless configured explicitly not to include NodeID.
- In parallel to the attempt made to create NP-bypass or LP-bypass, the PLR SHOULD initiate a Node-ID based Hello session to the NNhop or Nhop node respectively to establish the RSVP-TE signaling adjacency. This Hello session is used to detect MP node failure as well as determine the capability of the MP node. If the MP sets I-bit in CAPABILITY object [TE-SCALE-REC] carried in Hello message corresponding to NodeID based Hello session, then the PLR SHOULD conclude that the MP supports refresh-interval independent FRR procedures defined in this document.
- If the bypass LSP comes up, then the PLR SHOULD include Bypass Summary FRR Association object and triggers PATH to be sent. If Bypass Summary FRR Association object is included in PATH message, then the encoding rules specified in [SUMMARY-FRR] MUST be followed.

#### 4.1.2. Remote Signaling Adjacency

A NodeID based RSVP-TE Hello session is one in which NodeID is used in source and destination address fields in RSVP Hello. [RFC4558] formalizes NodeID based Hello messages between two routers. This document extends NodeID based RSVP Hello session to track the state of RSVP-TE neighbor that is not directly connected by at least one interface. In order to apply NodeID based RSVP-TE Hello session between any two routers that are not immediate neighbors, the router that supports the extensions defined in the document SHOULD set TTL to 255 in the NodeID based Hello messages exchanged between PLR and MP. The default hello interval for this NodeID hello session SHOULD be set to the default specified in [TE-SCALE-REC].

In the rest of the document the term "signaling adjacency", or "remote signaling adjacency" refers specifically to the RSVP-TE signaling adjacency.

#### 4.1.3. MP Behavior

When the NNhop or Nhop node receives the triggered PATH with a "matching" Bypass Summary FRR Association object, the node should consider itself as the MP for the PLR IP address "corresponding" to the Bypass Summary FRR Association object. The matching and ordering rules of Bypass Summary FRR Association specified in [SUMMARY-FRR] SHOULD be followed by implementations supporting this document.

In addition to the above procedures, the node SHOULD check the presence of remote signaling adjacency with PLR (this check is needed to detect network being partitioned). If a matching Bypass Summary FRR Association object is found in PATH and the RSVP-TE signaling adjacency is present, the node concludes that the PLR will undertake refresh-interval independent FRR procedures specified in this document. If the PLR has included NodeID in PATH RRO, then that NodeID is the remote neighbor address. Otherwise, the PLR's interface address in RRO will be the remote neighbor address. If a matching Bypass Summary FRR Association object is included by PPhop node, then it is NP-MP. If a matching Bypass Summary FRR Association object is included by Phop node, it concludes it is LP-MP.

#### 4.1.4. "Remote" state on MP

Once a router concludes it is MP for a PLR running refresh-interval independent FRR procedures, it SHOULD create a remote path state for

the LSP. The "remote" state is identical to the protected LSP path state except for the difference in RSVP\_HOP object. The RSVP\_HOP object in "remote" Path state contains the address that the PLR uses to send NodeID hello messages to MP.

The MP SHOULD consider the "remote" path state automatically deleted if:

- MP later receives a PATH with no matching Bypass Summary FRR Association object corresponding to the PLR RRO, or
- Node signaling adjacency with PLR goes down, or
- MP receives backup LSP signaling from PLR or
- MP receives PathTear, or
- MP deletes the LSP state on local policy or exception event

Unlike the normal path state that is either locally generated on Ingress or created from PATH message from Phop node, the "remote" path state is not signaled explicitly from PLR. The purpose of "remote" path state is to enable the PLR to explicitly tear down path and reservation states corresponding to the LSP by sending tear message for the "remote" path state. Such message tearing down "remote" path state is called "Remote PathTear."

The scenarios in which "Remote" PathTear is applied are described in Section 4.4 - Remote State Teardown.

#### 4.2. Impact of Failures on LSP State

This section describes the procedures for routers on the LSP path for different kinds of failures. The procedures described on detecting RSVP control plane adjacency failures do not impact the RSVP-TE graceful restart mechanisms ([RFC3473], [RFC5063]). If the router executing these procedures act as helper for neighboring router, then the control plane adjacency will be declared as having failed after taking into account the grace period extended for neighbor by the helper.

Immediate node failures are detected from the state of NodeID hello sessions established with immediate neighbors. [TE-SCALE-REC] recommends each router to establish NodeID hello sessions with all its immediate neighbors. PLR or MP node failure is detected from the

state of remote signaling adjacency established according to Section 4.1.2 of this document.

#### 4.2.1. Non-MP Behavior

When a router detects Phop link or Phop node failure and the router is not an MP for the LSP, then it SHOULD send Conditional PathTear (refer to Section "Conditional PathTear" below) and delete PSB and RSB states corresponding to the LSP.

#### 4.2.2. LP-MP Behavior

When the Phop link for an LSP fails on a router that is LP-MP for the LSP, the LP-MP SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Node-ID signaling adjacency with Phop PLR goes down, or
- MP receives normal or "Remote" PathTear for PSB, or
- MP receives ResvTear RSB.

When a router that is LP-MP for an LSP detects Phop node failure from Node-ID signaling adjacency state, the LP-MP SHOULD send normal PathTear and delete PSB and RSB states corresponding to the LSP.

#### 4.2.3. NP-MP Behavior

When a router that is NP-MP for an LSP detects Phop link failure, or Phop node failure from Node-ID signaling adjacency, the router SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or
- MP receives normal or "Remote" PathTear for PSB, or
- MP receives ResvTear for RSB.

When a router that is NP-MP does not detect Phop link or node failure, but receives Conditional PathTear from the Phop node, then the router SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or

- MP receives normal or "Remote" PathTear for PSB, or
- MP receives ResvTear for RSB.

Receiving Conditional PathTear from the Phop node will not impact the "remote" state from the PLR. Note that Phop node would send Conditional PathTear if it was not an MP.

In the example topology in Figure 1, assume C & D are NP-MP for PLRs A & B respectively. Now when A-B link fails, as B is not MP and its Phop link signaling adjacency has failed, B will delete LSP state (this behavior is required for unprotected LSPs - Section 4.2.1). In the data plane, that would require B to delete the label forwarding entry corresponding to the LSP. So if B's downstream nodes C and D continue to retain state, it would not be correct for D to continue to assume itself as NP-MP for PLR B.

The mechanism that enables D to stop considering itself as NP-MP and delete "remote" path state is given below.

1. When C receives Conditional PathTear from B, it decides to retain LSP state as it is NP-MP of PLR A. C also SHOULD check whether Phop B had previously signaled availability of node protection. As B had previously signaled NP availability in its PATH RRO, C SHOULD remove SUMMARY\_FRR\_BYPASS\_ASSOCIATION sub-object corresponding to B from the RRO and trigger PATH to D.
2. When D receives triggered PATH, it realizes that it is no longer NP-MP and so deletes the "remote" path state. D does not propagate PATH further down because the only change is in PATH RRO SUMMARY\_FRR\_BYPASS\_ASSOCIATION sub-object corresponding to B.

#### 4.2.4. Behavior of a Router that is both LP-MP and NP-MP

A router may be both LP-MP as well as NP-MP at the same time for Phop and PPhop nodes respectively of an LSP. If Phop link fails on such node, the node SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Both Node-ID signaling adjacencies with Phop and PPhop nodes go down, or
- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear for RSB.

If a router that is both LP-MP and NP-MP detects Phop node failure, then the node SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or
- MP receives normal or "Remote" PathTear for PSB, or
- MP receives ResvTear for RSB.

#### 4.3. Conditional Path Tear

In the example provided in the Section 4.2.5 "NP-MP Behavior on PLR link failure", B deletes PSB and RSB states corresponding to the LSP once B detects its link to Phop went down as B is not MP. If B were to send PathTear normally, then C would delete LSP state immediately. In order to avoid this, there should be some mechanism by which B can indicate to C that B does not require the receiving node to unconditionally delete the LSP state immediately. For this, B SHOULD add a new optional object called CONDITIONS object in PathTear. The new optional object is defined in Section 4.3.3. If node C also understands the new object, then C SHOULD delete LSP state only if it is not an NP-MP - in other words C SHOULD delete LSP state if there is no "remote" PLR state on C.

##### 4.3.1. Sending Conditional Path Tear

A router that is not an MP for an LSP SHOULD delete PSB and RSB states corresponding to the LSP if Phop link or Phop Node-ID signaling adjacency goes down (Section 4.2.1). The router SHOULD send Conditional PathTear if the following are also true.

- Ingress has requested node protection for the LSP, and
- PathTear is not received from upstream node

##### 4.3.2. Processing Conditional Path Tear

When a router that is not an NP-MP receives Conditional PathTear, the node SHOULD delete PSB and RSB states corresponding to the LSP, and process Conditional PathTear by considering it as normal PathTear. Specifically, the node SHOULD NOT propagate Conditional PathTear downstream but remove the optional object and send normal PathTear downstream.

When a node that is an NP-MP receives Conditional PathTear, it SHOULD NOT delete LSP state. The node SHOULD check whether the Phop node had previously included Bypass Summary FRR Association object in PATH. If the object had been included previously by Phop, then the node processing Conditional PathTear from Phop SHOULD remove the corresponding object and trigger PATH downstream.

If Conditional PathTear is received from a neighbor that has not advertised support (refer to Section 4.5) for the new procedures defined in this document, then the node SHOULD consider the message as normal PathTear. The node SHOULD propagate normal PathTear downstream and delete LSP state.

#### 4.3.3. CONDITIONS object

As any implementation that does not support Conditional PathTear SHOULD ignore the new object but process the message as normal PathTear without generating any error, the Class-Num of the new object SHOULD be 10bbbbbb where 'b' represents a bit (from Section 3.10 of [RFC2205]).

The new object is called as "CONDITIONS" object that will specify the conditions under which default processing rules of the RSVP-TE message SHOULD be invoked.

The object has the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|          Length          |   Class   |   C-type   |
+-----+-----+-----+-----+-----+-----+-----+
|                               Reserved                               |M|
+-----+-----+-----+-----+-----+-----+-----+

```

Length

This contains the size of the object in bytes and should be set to eight.

Class

To be assigned

C-type

1

M bit

If M-bit is set to 1, then the PathTear message SHOULD be processed based on the condition if the receiver router is a Merge Point or not.

If M-bit is set to 0, then the PathTear message SHOULD be processed as normal PathTear message.

#### 4.4. Remote State Teardown

If the Ingress wants to tear down the LSP because of a management event while the LSP is being locally repaired at a transit PLR, it would not be desirable to wait till backup LSP signaling to perform state cleanup. To enable LSP state cleanup when the LSP is being locally repaired, the PLR SHOULD send "remote" PathTear message instructing the MP to delete PSB and RSB states corresponding to the LSP. The TTL in "remote" PathTear message SHOULD be set to 255.

Consider node C in example topology (Figure 1) has gone down and B locally repairs the LSP.

1. Ingress A receives a management event to tear down the LSP.
2. A sends normal PathTear to B.
3. To enable LSP state cleanup, B SHOULD send "remote" PathTear with destination IP address set to that of D used in Node-ID signaling adjacency with D, and RSVP\_HOP object containing local address used in Node-ID signaling adjacency.
4. B then deletes PSB and RSB states corresponding to the LSP.
5. On D there would be a remote signaling adjacency with B and so D SHOULD accept the remote PathTear and delete PSB and RSB states corresponding to the LSP.

##### 4.4.1. PLR Behavior on Local Repair Failure

If local repair fails on the PLR after a failure, then this should be considered as a case for cleaning up LSP state from PLR to the Egress. PLR would achieve this using "remote" PathTear to clean up state from MP. If MP has retained state, then it would propagate PathTear downstream thereby achieving state cleanup. Note that in



the case of link protection, the PathTear would be directed to LP-MP node IP address rather than the Nhop interface address.

#### 4.4.2. PLR Behavior on Resv RRO Change

When a router that has already made NP available detects a change in the RRO carried in RESV message, and if the RRO change indicates that the router's former NP-MP is no longer present in the LSP path, then the router SHOULD send "Remote" PathTear directly to its former NP-MP.

In the example topology in Figure 1, assume A has made node protection available and C has concluded it is NP-MP. When the B-C link fails then implementing the procedure specified in Section 4.2.4 of this document, C will retain state till: remote NodeID control plane adjacency with A goes down, or PathTear or ResvTear is received for PSB or RSB respectively. If B also has made node protection available, B will eventually complete backup LSP signaling with its NP-MP D and trigger RESV to A with RRO changed. The new RRO of the LSP carried in RESV will not contain C. When A processes the RESV with a new RRO not containing C - its former NP-MP, A SHOULD send "Remote" PathTear to C. When C receives a "Remote" PathTear for its PSB state, C will send normal PathTear downstream to D and delete both PSB and RSB states corresponding to the LSP. As D has already received backup LSP signaling from B, D will retain control plane and forwarding states corresponding to the LSP.

#### 4.4.3. LSP Preemption during Local Repair

If an LSP is preempted when there is no failure along the path of the LSP, the node on which preemption occurs would send PathErr and ResvTear upstream and only delete the forwarding state and RSB state corresponding to the LSP. But if the LSP is being locally repaired upstream of the node on which the LSP is preempted, then the node SHOULD delete both PSB and RSB states corresponding to the LSP and send normal PathTear downstream.

##### 4.4.3.1. Preemption on LP-MP after Phop Link failure

If an LSP is preempted on LP-MP after its Phop or incoming link has already failed but the backup LSP has not been signaled yet, then the node SHOULD send normal PathTear and delete both PSB and RSB states corresponding to the LSP. As the LP-MP has retained LSP state because the PLR would signal the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be LP-MP any more requiring the node to clean up LSP state.

#### 4.4.3.2. Preemption on NP-MP after Phop Link failure

If an LSP is preempted on NP-MP after its Phop link has already failed but the backup LSP has not been signaled yet, then the node SHOULD send normal PathTear and delete PSB and RSB states corresponding to the LSP. As the NP-MP has retained LSP state because the PLR would signal the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be NP-MP any more requiring the node to clean up LSP state.

Consider B-C link goes down on the same example topology (Figure 1). As C is NP-MP for PLR A, C will retain LSP state.

1. The LSP is preempted on C.
  2. C will delete RSB state corresponding to the LSP. But C cannot send PathErr or ResvTear to PLR A because backup LSP has not been signaled yet.
  3. As the only reason for C having retained state after Phop node failure was that it was NP-MP, C SHOULD send normal PathTear to D and delete PSB state also. D would also delete PSB and RSB states on receiving PathTear from C.
  4. B starts backup LSP signaling to D. But as D does not have the LSP state, it will reject backup LSP PATH and send PathErr to B.
  5. B will delete its reservation and send ResvTear to A.
- #### 4.5. Backward Compatibility Procedures

The "Refresh interval Independent FRR" or RI-RSVP-FRR referred below in this section refers to the changes that have been proposed in previous sections. Any implementation that does not support them has been termed as "non-RI-RSVP-FRR implementation". The extensions proposed in [SUMMARY-FRR] are applicable to implementations that do not support RI-RSVP-FRR. On the other hand, changes proposed relating to LSP state cleanup namely Conditional and remote PathTear require support from one-hop and two-hop neighboring nodes along the LSP path. So procedures that fall under LSP state cleanup category SHOULD be turned on only if all nodes involved in the node protection FRR i.e. PLR, MP and intermediate node in the case of NP, support the extensions. Note that for LSPs requesting only link protection, the PLR and the LP-MP should support the extensions.

#### 4.5.1. Detecting Support for Refresh interval Independent FRR

An implementation supporting the extensions specified in previous sections (called RI-RSVP-FRR here after) SHOULD set the flag "Refresh interval Independent RSVP" or RI-RSVP in CAPABILITY object in Hello messages. The RI-RSVP flag is specified in [TE-SCALE-REC].

- As nodes supporting the extensions SHOULD initiate Node Hellos with adjacent nodes, a node on the path of protected LSP can determine whether its PPhop or Nhop neighbor supports RI-RSVP-FRR enhancements from the Hello messages sent by the neighbor.
- If a node attempts to make node protection available, then the PLR SHOULD initiate remote Node-ID signaling adjacency with NNhop. If the NNhop (a) does not reply to remote node Hello message or (b) does not set "Enhanced facility protection" flag in CAPABILITY object in the reply, then the PLR can conclude that NNhop does not support RI-RSVP-FRR extensions.
- If node protection is requested for an LSP and if (a) PPhop node has not included a matching Bypass Summary FRR Association object in PATH or (b) PPhop node has not initiated remote node Hello messages, then the node SHOULD conclude that PLR does not support RI-RSVP-FRR extensions. The details are described in the "Procedures for backward compatibility" section below.

Any node that sets the I-bit is set in its CAPABILITY object MUST also set Refresh-Reduction-Capable bit in common header of all RSVP-TE messages.

#### 4.5.2. Procedures for backward compatibility

The procedures defined hereafter are performed on a subset of LSPs that traverse a node, rather than on all LSPs that traverse a node. This behavior is required to support backward compatibility for a subset of LSPs traversing nodes running non-RI-RSVP-FRR implementations.

##### 4.5.2.1. Lack of support on Downstream Node

- If the Nhop does not support the RI-RSVP-FRR extensions, then the node SHOULD reduce the "refresh period" in TIME\_VALUES object carried in PATH to default small refresh default value.
- If node protection is requested and the NNhop node does not support the enhancements, then the node SHOULD reduce the "refresh

period" in TIME\_VALUES object carried in PATH to a small refresh default value.

If the node reduces the refresh time from the above procedures, it SHOULD also not send remote PathTear or Conditional PathTear messages.

Consider the example topology in Figure 1. If C does not support the RI-RSVP-FRR extensions, then:

- A and B SHOULD reduce the refresh time to default value of 30 seconds and trigger PATH
- If B is not an MP and if Phop link of B fails, B cannot send Conditional PathTear to C but SHOULD time out PSB state from A normally. This would be accomplished if A would also reduce the refresh time to default value. So if C does not support the RI-RSVP-FRR extensions, then Phop B and PPhop A SHOULD reduce refresh time to a small default value.

#### 4.5.2.2. Lack of support on Upstream Node

- If Phop node does not support the RI-RSVP-FRR extensions, then the node SHOULD reduce the "refresh period" in TIME\_VALUES object carried in RESV to default small refresh time value.
- If node protection is requested and the Phop node does not support the RI-RSVP-FRR extensions, then the node SHOULD reduce the "refresh period" in TIME\_VALUES object carried in PATH to default value.
- If node protection is requested and PPhop node does not support the RI-RSVP-FRR extensions, then the node SHOULD reduce the "refresh period" in TIME\_VALUES object carried in RESV to default value.
- If the node reduces the refresh time from the above procedures, it SHOULD also not execute MP procedures specified in Section 4.2 of this document.

#### 4.5.2.3. Incremental Deployment

The backward compatibility procedures described in the previous subsections imply that a router supporting the RI-RSVP-FRR extensions specified in this document can apply the procedures specified in the document either in the downstream or upstream direction of an LSP,

depending on the capability of the routers downstream or upstream in the LSP path.

- RI-RSVP-FRR extensions and procedures are enabled for downstream Path, PathTear and ResvErr messages corresponding to an LSP if link protection is requested for the LSP and the Nhop node supports the extensions
- RI-RSVP-FRR extensions and procedures are enabled for downstream Path, PathTear and ResvErr messages corresponding to an LSP if node protection is requested for the LSP and both Nhop & NNhop nodes support the extensions
- RI-RSVP-FRR extensions and procedures are enabled for upstream PathErr, Resv and ResvTear messages corresponding to an LSP if link protection is requested for the LSP and the Phop node supports the extensions
- RI-RSVP-FRR extensions and procedures are enabled for upstream PathErr, Resv and ResvTear messages corresponding to an LSP if node protection is requested for the LSP and both Phop and PPhop nodes support the extensions

For example, if an implementation supporting the RI-RSVP-FRR extensions specified in this document is deployed on all routers in particular region of the network and if all the LSPs in the network request node protection, then the FRR extensions will only be applied for the LSP segments that traverse the particular region. This will aid incremental deployment of these extensions and also allow reaping the benefits of the extensions in portions of the network where it is supported.

## 5. Security Considerations

This security considerations pertaining to [RFC2205], [RFC3209] and [RFC5920] remain relevant.

This document extends the applicability of Node-ID based Hello session between immediate neighbors. The Node-ID based Hello session between PLR and NP-MP may require the two routers to exchange Hello messages with non-immediate neighbor. So, the implementations SHOULD provide the option to configure Node-ID neighbor specific or global authentication key to authentication messages received from Node-ID neighbors. The network administrator MAY utilize this option to enable RSVP-TE routers to authenticate Node-ID Hello messages received with TTL greater than 1. Implementations SHOULD also

provide the option to specify a limit on the number of Node-ID based Hello sessions that can be established on a router supporting the extensions defined in this document.

## 6. IANA Considerations

### 6.1. New Object - CONDITIONS

RSVP Change Guidelines [RFC3936] defines the Class-Number name space for RSVP objects. The name space is managed by IANA.

IANA registry: RSVP Parameters

Subsection: Class Names, Class Numbers, and Class Types

A new RSVP object using a Class-Number from 128-183 range called the "CONDITIONS" object is defined in Section 4.3 of this document. The Class-Number from 128-183 range will be allocated by IANA.

## 7. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3209] Awduche, D., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.
- [RFC4090] Pan, P., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.
- [RFC2961] Berger, L., "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.
- [RFC2205] Braden, R., "Resource Reservation Protocol (RSVP)", RFC 2205, September 1997.
- [RFC4558] Ali, Z., "Node-ID Based Resource Reservation (RSVP) Hello: A Clarification Statement", RFC 4558, June 2006.
- [RFC3473] Berger, L., "Generalized Multi-Protocol Label Switching Signaling Resource Reservation Protocol-Traffic Engineering Extensions", RFC 3473, January 2003.
- [RFC5063] Satyanarayana, A., "Extensions to GMPLS Resource Reservation Protocol Graceful Restart", RFC5063, October 2007.

[RFC3936] Kompella, K. and J. Lang, "Procedures for Modifying the Resource reSerVation Protocol (RSVP)", BCP 96, RFC 3936, October 2004.

[TE-SCALE-REC] Vishnu Pavan Beeram et. al, "Implementation Recommendations to improve scalability of RSVP-TE Deployments", draft-ietf-teas-rsvp-te-scaling-rec (work in progress)

[SUMMARY-FRR] Mike Tallion et. al, "RSVP-TE Summary Fast Reroute Extensions for LSP Tunnels", draft-mtaillon-mpls-summary-frr-rsvpte (work in progress)

## 8. Informative References

[RFC5439] Yasukawa, S., "An Analysis of Scaling Issues in MPLS-TE Core Networks", RFC 5439, February 2009.

[RFC5920] Fang, L., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

## 9. Acknowledgments

We are very grateful to Yakov Rekhter for his contributions to the development of the idea and thorough review of content of the draft. Thanks to Raveendra Torvi and Yimin Shen for their comments and inputs.

## 10. Contributors

Markus Jork  
Juniper Networks  
Email: mjork@juniper.net

Harish Sitaraman  
Juniper Networks  
Email: hsitaraman@juniper.net

Vishnu Pavan Beeram  
Juniper Networks  
Email: vbeeram@juniper.net

Ebben Aries  
Juniper Networks

Email: [exa@juniper.net](mailto:exa@juniper.net)

Mike Tallion  
Cisco Systems Inc.  
Email: [mtallion@cisco.com](mailto:mtallion@cisco.com)

#### 11. Authors' Addresses

Chandra Ramachandran  
Juniper Networks  
Email: [csekar@juniper.net](mailto:csekar@juniper.net)

Ina Minei  
Google, Inc  
[inaminei@google.com](mailto:inaminei@google.com)

Dante Pacella  
Verizon  
Email: [dante.j.pacella@verizon.com](mailto:dante.j.pacella@verizon.com)

Tarek Saad  
Cisco Systems Inc.  
Email: [tsaad@cisco.com](mailto:tsaad@cisco.com)





MPLS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 18 August 2022

G. Mirsky  
Ericsson  
J. Tantsura  
Juniper Networks  
I. Varlashkin  
Google  
M. Chen  
Huawei  
14 February 2022

Bidirectional Forwarding Detection (BFD) Directed Return Path for MPLS  
Label Switched Paths (LSPs)  
draft-ietf-mpls-bfd-directed-19

Abstract

Bidirectional Forwarding Detection (BFD) is expected to be able to monitor a wide variety of encapsulations of paths between systems. When a BFD session monitors an explicitly routed unidirectional path there may be a need to direct egress BFD peer to use a specific path for the reverse direction of the BFD session.

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 18 August 2022.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Conventions used in this document . . . . .	3
1.1.1. Requirements Language . . . . .	3
2. Problem Statement . . . . .	3
3. Control of the Reverse BFD Path . . . . .	3
3.1. BFD Reverse Path TLV . . . . .	3
3.2. Return Codes . . . . .	5
4. Use Case Scenario . . . . .	5
5. Operational Considerations . . . . .	6
6. IANA Considerations . . . . .	6
6.1. BFD Reverse Path TLV . . . . .	6
6.2. Return Code . . . . .	7
7. Implementation Status . . . . .	7
8. Security Considerations . . . . .	8
9. Normative References . . . . .	8
Appendix A. Acknowledgments . . . . .	9
Authors' Addresses . . . . .	9

## 1. Introduction

[RFC5880], [RFC5881], and [RFC5883] established the BFD protocol for IP networks. [RFC5884] and [RFC7726] set rules for using BFD asynchronous mode over IP/MPLS LSPs, while not defining means to control the path an egress BFD system uses to send BFD control packets towards the ingress BFD system.

For the case when BFD is used to detect defects of the traffic engineered LSP the path the BFD control packets transmitted by the egress BFD system toward the ingress may be disjoint from the LSP in the forward direction. The fact that BFD control packets are not guaranteed to follow the same links and nodes in both forward and reverse directions may be one of the factors contributing to producing false positive defect notifications, i.e., false alarms, at the ingress BFD peer. Ensuring that both directions of the BFD session use co-routed paths may, in some environments, improve the determinism of the failure detection and localization.

This document defines the BFD Reverse Path TLV as an extension to LSP Ping [RFC8029] and proposes that it is to be used to instruct the egress BFD system to use an explicit path for its BFD control packets

associated with a particular BFD session. The TLV will be allocated from the TLV and sub-TLV registry defined in [RFC8029]. As a special case, forward and reverse directions of the BFD session can form a bi-directional co-routed associated channel.

### 1.1. Conventions used in this document

#### 1.1.1. 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.

### 2. Problem Statement

When BFD is used to monitor explicitly routed unidirectional path, e.g., MPLS-TE LSP, BFD control packets in forward direction would be in-band using the mechanism defined in [RFC5884]. But the reverse direction of the BFD session would follow the shortest path route and that might lead to the problem in detecting failures on an explicit unidirectional path, as described below:

- \* detection by an ingress node of a failure on the reverse path may not be unambiguously interpreted as the failure of the path in the forward direction.

To address this scenario, the egress BFD peer would be instructed to use a specific path for BFD control packets.

### 3. Control of the Reverse BFD Path

To bootstrap a BFD session over an MPLS LSP, LSP ping, defined in [RFC8029], MUST be used with BFD Discriminator TLV [RFC5884]. This document defines a new TLV, BFD Reverse Path TLV, that MAY contain none, one or more sub-TLVs that can be used to carry information about the reverse path for the BFD session that is specified by the value in BFD Discriminator TLV.

#### 3.1. BFD Reverse Path TLV

The BFD Reverse Path TLV is an optional TLV within the LSP ping [RFC8029]. However, if used, the BFD Discriminator TLV MUST be included in an Echo Request message as well. If the BFD Discriminator TLV is not present when the BFD Reverse Path TLV is included; then it MUST be treated as malformed Echo Request, as described in [RFC8029].

The BFD Reverse Path TLV carries information about the path onto which the egress BFD peer of the BFD session referenced by the BFD Discriminator TLV MUST transmit BFD control packets. The format of the BFD Reverse Path TLV is as presented in Figure 1.

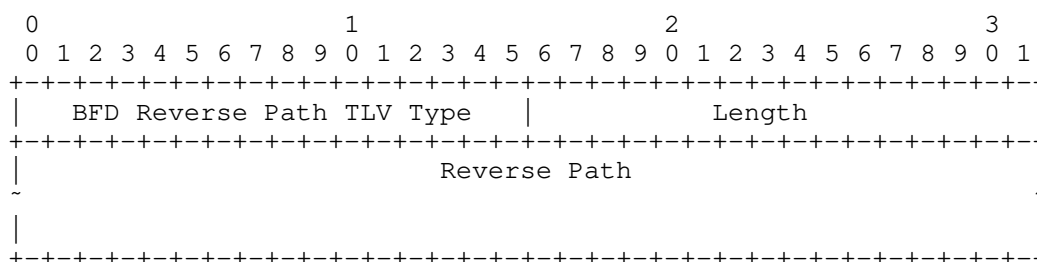


Figure 1: BFD Reverse Path TLV

BFD Reverse Path TLV Type is two octets in length and has a value of TBD1 (to be assigned by IANA as requested in Section 6).

Length field is two octets long and defines the length in octets of the Reverse Path field.

Reverse Path field contains none, one or more sub-TLVs. Any non-multicast Target FEC Stack sub-TLV (already defined, or to be defined in the future) for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be used in this field. Multicast Target FEC Stack sub-TLVs, i.e., p2mp and mp2mp, SHOULD NOT be included in Reverse Path field. If the egress LSR finds multicast Target Stack sub-TLV, it MUST send echo reply with the received Reverse Path TLV, BFD Discriminator TLV and set the Return Code to "Inappropriate Target FEC Stack sub-TLV present" Section 3.2. None, one or more sub-TLVs MAY be included in the BFD Reverse Path TLV. If no sub-TLVs are found in the BFD Reverse Path TLV, the egress BFD peer MUST revert to using the local policy-based decision as described in Section 7 [RFC5884], i.e., routed over IP network.

If the egress LSR cannot find the path specified in the Reverse Path TLV it MUST send Echo Reply with the received BFD Discriminator TLV, Reverse Path TLV and set the Return Code to "Failed to establish the BFD session. The specified reverse path was not found" Section 3.2. An implementation MAY provide configuration options to define action at the egress BFD peer. For example, if the egress LSR cannot find the path specified in the Reverse Path TLV, it MAY establish the BFD session over an IP network, as defined in [RFC5884].

The BFD Reverse Path TLV MAY be used in the bootstrapping of a BFD session process described in Section 6 [RFC5884]. A system that supports this specification MUST support using the BFD Reverse Path TLV after the BFD session has been established. If a system that supports this specification receives an LSP Ping with the BFD Discriminator TLV and no BFD Reverse Path TLV even though the reverse path for the specified BFD session has been established according to the previously received BFD Reverse Path TLV, the egress LSR MUST transition to transmitting periodic BFD Control messages as defined in Section 7 [RFC5884].

### 3.2. Return Codes

This document defines the following Return Codes for MPLS LSP Echo Reply:

- \* "Inappropriate Target FEC Stack sub-TLV present", (TBD3). When multicast Target FEC Stack sub-TLV found in the received Echo Request by the egress BFD peer, an Echo Reply with the return code set to "Inappropriate Target FEC Stack sub-TLV present" MUST be sent to the ingress BFD peer Section 3.1.
- \* "Failed to establish the BFD session. The specified reverse path was not found", (TBD4). When a specified reverse path is not available at the egress BFD peer, an Echo Reply with the return code set to "Failed to establish the BFD session. The specified reverse path was not found" MUST be sent back to the ingress BFD peer Section 3.1.

### 4. Use Case Scenario

In the network presented in Figure 2 node A monitors two tunnels to node H: A-B-C-D-G-H and A-B-E-F-G-H. To bootstrap a BFD session to monitor the first tunnel, node A MUST include a BFD Discriminator TLV with Discriminator value (e.g., foobar-1) and MAY include a BFD Reverse Path TLV that references H-G-D-C-B-A tunnel. To bootstrap a BFD session to monitor the second tunnel, node A MUST include a BFD Discriminator TLV with a different Discriminator value (e.g., foobar-2) [RFC7726] and MAY include a BFD Reverse Path TLV that references H-G-F-E-B-A tunnel.

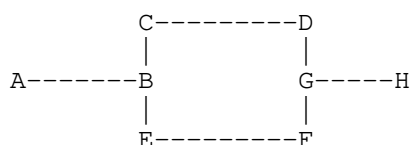


Figure 2: Use Case for BFD Reverse Path TLV

If an operator needs node H to monitor a path to node A, e.g. H-G-D-C-B-A tunnel, then by looking up the list of known Reverse Paths it MAY find and use the existing BFD session.

## 5. Operational Considerations

When an explicit path is set either as Static or RSVP-TE LSP, corresponding sub-TLVs, defined in [RFC7110], MAY be used to identify the explicit reverse path for the BFD session. If any of defined in [RFC7110] sub-TLVs used in BFD Reverse Path TLV, then the periodic verification of the control plane against the data plane, as recommended in Section 4 [RFC5884], MUST use the Return Path TLV, as per [RFC7110], with that sub-TLV. By using the LSP Ping with Return Path TLV, an operator monitors whether at the egress BFD node the reverse LSP is mapped to the same FEC as the BFD session. Selection and control of the rate of LSP Ping with Return Path TLV follows the recommendation of [RFC5884]: "The rate of generation of these LSP Ping Echo request messages SHOULD be significantly less than the rate of generation of the BFD Control packets. An implementation MAY provide configuration options to control the rate of generation of the periodic LSP Ping Echo request messages."

Suppose an operator planned network maintenance activity that possibly affects FEC used in the BFD Reverse Path TLV. In that case, the operator MUST avoid the unnecessary disruption using the LSP Ping with a new FEC in the BFD Reverse Path TLV. But in some scenarios, proactive measures cannot be taken. Because the frequency of LSP Ping messages will be lower than the defect detection time provided by the BFD session. As a result, a change in the reverse-path FEC will first be detected as the BFD session's failure. In such a case, the ingress BFD node SHOULD immediately transmit the LSP Ping Echo request with Return Path TLV to verify whether the FEC is still valid. If the failure was caused by the change in the FEC used for the reverse direction of the BFD session, the ingress BFD node SHOULD bootstrap a new BFD session using another FEC in BFD Reverse Path TLV.

## 6. IANA Considerations

### 6.1. BFD Reverse Path TLV

The IANA is requested to assign a new value for BFD Reverse Path TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "TLVs and sub-TLVs" sub-registry.

Value	Description	Reference
(TBD1)	BFD Reverse Path TLV	This document

Table 1: New BFD Reverse Type TLV

## 6.2. Return Code

The IANA is requested to assign a new Return Code value from the "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, "Return Codes" sub-registry, as follows using a Standards Action value.

Value	Description	Reference
(TBD3)	Inappropriate Target FEC Stack sub-TLV present.	This document
(TBD4)	Failed to establish the BFD session. The specified reverse path was not found.	This document

Table 2: New Return Code

## 7. Implementation Status

- The organization responsible for the implementation: ZTE Corporation.
  - The implementation's name ROSng empowers traditional routers, e.g., ZXCTN 6000.
  - A brief general description: A Return Path can be specified for a BFD session over RSVP tunnel or LSP. The same can be specified for a backup RSVP tunnel/LSP.
- The implementation's level of maturity: production.
- Coverage: RSVP LSP (no support for Static LSP)
  - Version compatibility: draft-ietf-mpls-bfd-directed-10.
  - Licensing: proprietary.



- Implementation experience: simple once you support RFC 7110.
- Contact information: Qian Xin qian.xin2@zte.com.cn
- The date when information about this particular implementation was last updated: 12/16/2019

Note to RFC Editor: This section MUST be removed before publication of the document.

## 8. Security Considerations

Security considerations discussed in [RFC5880], [RFC5884], [RFC7726], [RFC8029], and [RFC7110] apply to this document.

## 9. 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>>.
- [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>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, DOI 10.17487/RFC7110, January 2014, <<https://www.rfc-editor.org/info/rfc7110>>.

- [RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <<https://www.rfc-editor.org/info/rfc7726>>.
- [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>>.

#### Appendix A. Acknowledgments

The authors greatly appreciate a thorough review and the most helpful comments from Eric Gray and Carlos Pignataro. The authors much appreciate the help of Qian Xin, who provided information about the implementation of this specification.

#### Authors' Addresses

Greg Mirsky  
Ericsson

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Jeff Tantsura  
Juniper Networks

Email: [jefftant.ietf@gmail.com](mailto:jefftant.ietf@gmail.com)

Ilya Varlashkin  
Google

Email: [Ilya@nobulus.com](mailto:Ilya@nobulus.com)

Mach(Guoyi) Chen  
Huawei

Email: [mach.chen@huawei.com](mailto:mach.chen@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 14, 2017

W. Cheng  
L. Wang  
H. Li  
China Mobile  
H. Helvoort  
Hai Gaoming BV  
J. Dong  
Huawei Technologies  
June 12, 2017

Shared-Ring protection (MSRP) mechanism for ring topology  
draft-ietf-mpls-tp-shared-ring-protection-06

Abstract

This document describes requirements, architecture and solutions for MPLS-TP Shared Ring Protection (MSRP) in a ring topology for point-to-point (P2P) services. The MSRP mechanism is described to meet the ring protection requirements as described in RFC 5654. This document defines the Ring Protection Switch (RPS) Protocol that is used to coordinate the protection behavior of the nodes on MPLS ring.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 14, 2017.

## 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 (<http://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 and Notation . . . . .	3
3. MPLS-TP Ring Protection Criteria and Requirements . . . . .	4
4. Shared Ring Protection Architecture . . . . .	5
4.1. Ring Tunnel . . . . .	5
4.1.1. Establishment of Ring Tunnel . . . . .	6
4.1.2. Label Assignment and Distribution . . . . .	8
4.1.3. Forwarding Operation . . . . .	8
4.2. Failure Detection . . . . .	9
4.3. Ring Protection . . . . .	10
4.3.1. Wrapping . . . . .	11
4.3.2. Short Wrapping . . . . .	13
4.3.3. Steering . . . . .	16
4.4. Interconnected Ring Protection . . . . .	19
4.4.1. Interconnected Ring Topology . . . . .	19
4.4.2. Interconnected Ring Protection Mechanisms . . . . .	21
4.4.3. Ring Tunnels in Interconnected Rings . . . . .	22
4.4.4. Interconnected Ring Switching Procedure . . . . .	24
4.4.5. Interconnected Ring Detection Mechanism . . . . .	25
5. Ring Protection Coordination Protocol . . . . .	26
5.1. RPS and PSC Comparison on Ring Topology . . . . .	26
5.2. RPS Protocol . . . . .	27
5.2.1. Transmission and Acceptance of RPS Requests . . . . .	29
5.2.2. RPS PDU Format . . . . .	29
5.2.3. Ring Node RPS States . . . . .	31
5.2.4. RPS State Transitions . . . . .	33
5.3. RPS State Machine . . . . .	35
5.3.1. Switch Initiation Criteria . . . . .	35
5.3.2. Initial States . . . . .	37
5.3.3. State transitions When Local Request is Applied . . . . .	38
5.3.4. State Transitions When Remote Request is Applied . . . . .	42

5.3.5. State Transitions When Request Addresses to Another Node is Received . . . . .	45
6. IANA Considerations . . . . .	47
6.1. G-ACh Channel Type . . . . .	48
6.2. RPS Request Codes . . . . .	48
7. Operational Considerations . . . . .	48
8. Security Considerations . . . . .	49
9. Contributing Authors . . . . .	50
10. Acknowledgements . . . . .	51
11. References . . . . .	51
11.1. Normative References . . . . .	52
11.2. Informative References . . . . .	52
Authors' Addresses . . . . .	53

## 1. Introduction

As described in section 2.5.6.1 of [RFC5654], several service providers have expressed much interest in operating MPLS-TP in ring topologies and require a high-level survivability function in these topologies. In operational transport network deployment, MPLS-TP networks are often constructed using ring topologies. This calls for an efficient and optimized ring protection mechanism to achieve simple operation and fast, sub 50 ms, recovery performance.

This document specifies an MPLS-TP Shared-Ring Protection mechanisms that meets the criteria for ring protection and the ring protection requirements described in section 2.5.6.1 of [RFC5654].

The basic concept and architecture of the Shared-Ring protection mechanism are specified in this document. This document describes the solutions for point-to-point transport paths. While the basic concept may also apply to point-to-multipoint transport paths, the solution for point-to-multipoint transport paths is out of the scope of this document.

## 2. Terminology and Notation

Terminology:

Ring Node: All nodes in the ring topology are Ring Nodes and they MUST actively participate in the ring protection.

Ring tunnel: A ring tunnel provides a server layer for the LSPs traversing the ring. The notation used for a ring tunnel is: R<d><p><X> where <d> = c (clockwise) or a (anticlockwise), <p> = W (working) or P (protecting), and <X> = the node name.

Ring map: A ring map is present in each ring node. The ring-map contains the ring topology information, i.e. the nodes in the ring, the adjacency of the ring nodes and the status of the links between ring nodes (Intact or Severed). The ring map is used by every ring node to determine the switchover behavior of the ring tunnels.

Notation:

The following syntax will be used to describe the contents of the label stack:

1. The label stack will be enclosed in square brackets ("[]").
2. Each level in the stack will be separated by the '|' character. It should be noted that the label stack may contain additional layers. However, we only present the layers that are related to the protection mechanism.
3. If the Label is assigned by Node X, the Node Name is enclosed in parentheses ("()").

### 3. MPLS-TP Ring Protection Criteria and Requirements

The generic requirements for MPLS-TP protection are specified in [RFC5654]. The requirements specific for ring protection are specified in section 2.5.6.1 of [RFC5654]. This section describes how the criteria for ring protection are met:

- a. The number of OAM entities needed to trigger protection

Each ring node requires only one instance of the RPS protocol per ring. The OAM of the links connected to the adjacent ring-nodes has to be forwarded to only this instance in order to trigger protection. For detailed information, see section 5.2.

- b. The number of elements of recovery in the ring

Each ring-node requires only one instance of the RPS protocol and is independent of the number of LSPs that are protected. For detailed information, see section 5.2.

- c. The required number of labels required for the protection paths

The RPS protocol uses ring tunnels and each tunnel has a set of labels. The number of ring tunnel labels is related to the number of ring-nodes and is independent of the number of protected LSPs. For detailed information, see section 4.1.2.

d. The amount of control and management-plane transactions

Each ring-node requires only one instance of the RPS protocol per ring. This means that only one maintenance operation is required per ring-node. For detailed information, see section 5.2.

e. Minimize the signaling and routing information exchange during protection

Information exchange during a protection switch is using the in-band RPS and OAM messages. No control plane interactions are required. For detailed information, see section 5.2.

#### 4. Shared Ring Protection Architecture

##### 4.1. Ring Tunnel

This document introduces a new logical layer of the ring for shared ring protection in MPLS-TP networks. As shown in Figure 1, the new logical layer consists of ring tunnels which provides a server layer for the LSPs traversing the ring. Once a ring tunnel is established, the forwarding and protection switching of the ring are all performed at the ring tunnel level. A port can carry multiple ring tunnels, and a ring tunnel can carry multiple LSPs.

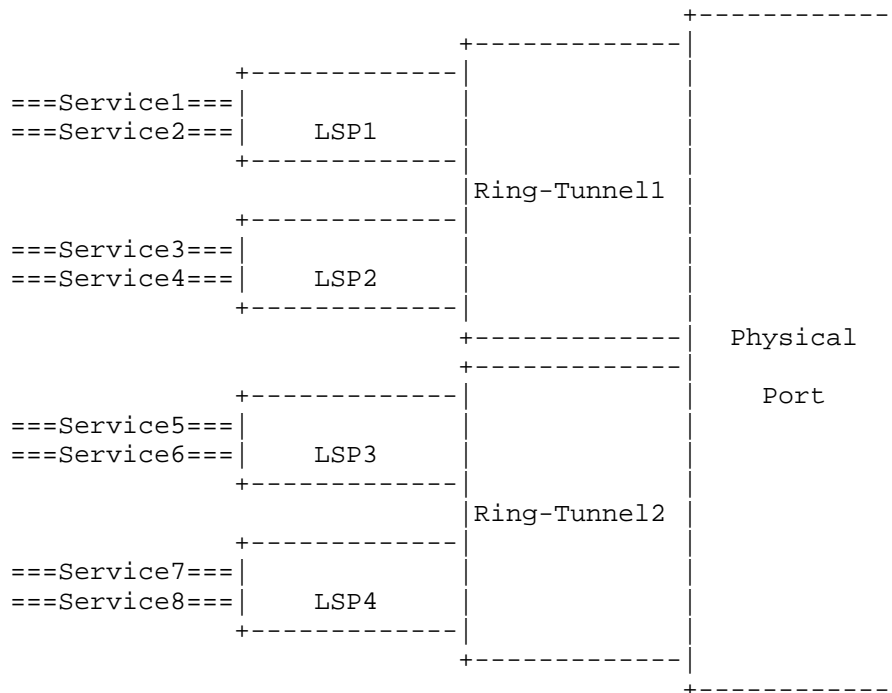


Figure 1. The logical layers of the ring

The label stack used in MPLS-TP Shared Ring Protection mechanism is [Ring Tunnel Label|LSP Label|service Label](Payload) as illustrated in figure 2.

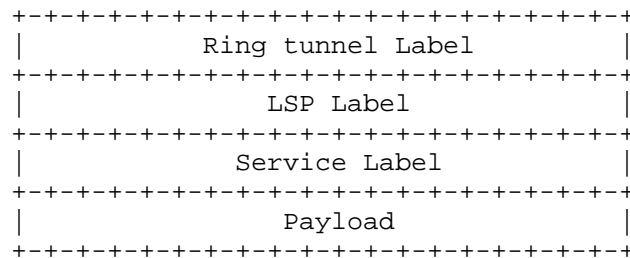


Figure 2. Label stack used in MPLS-TP Shared Ring Protection

#### 4.1.1. Establishment of Ring Tunnel

The Ring tunnels are established based on the egress nodes. The egress node is the node where traffic leaves the ring. LSPs which have the same egress node on the ring and travels along the ring in the same direction (clockwise or anticlockwise) share the same ring



tunnels. In other words, all the LSPs that traverse the ring in the same direction and exit from the same node share the same working ring tunnel and protection ring tunnel. For each egress node, four ring tunnels are established:

- o one clockwise working ring tunnel, which is protected by the anticlockwise protection ring tunnel
- o one anticlockwise protection ring tunnel
- o one anticlockwise working ring tunnel, which is protected by the clockwise protection ring tunnel
- o one clockwise protection ring tunnel

The structure of the protection tunnels is determined by the selected protection mechanism. This will be detailed in subsequent sections.

As shown in Figure 3, LSP1, LSP2 and LSP3 enter the ring from Node E, Node A and Node B respectively, and all leave the ring at Node D. To protect these LSPs that traverse the ring, a clockwise working ring tunnel (RcW\_D) via E->F->A->B->C->D, and its anticlockwise protection ring tunnel (RaP\_D) via D->C->B->A->F->E->D are established. Also, an anti-clockwise working ring tunnel (RaW\_D) via C->B->A->F->E->D, and its clockwise protection ring tunnel (RcP\_D) via D->E->F->A->B->C->D are established. For simplicity Figure 3 only shows RcW\_D and RaP\_D. A similar provisioning should be applied for any other node on the ring. In summary, for each node in Figure 3 when acting as egress node, the ring tunnels are created as follows:

- o To Node A: RcW\_A, RaW\_A, RcP\_A, RaP\_A
- o To Node B: RcW\_B, RaW\_B, RcP\_B, RaP\_B
- o To Node C: RcW\_C, RaW\_C, RcP\_C, RaP\_C
- o To Node D: RcW\_D, RaW\_D, RcP\_D, RaP\_D
- o To Node E: RcW\_E, RaW\_E, RcP\_E, RaP\_E
- o To Node F: RcW\_F, RaW\_F, RcP\_F, RaP\_F

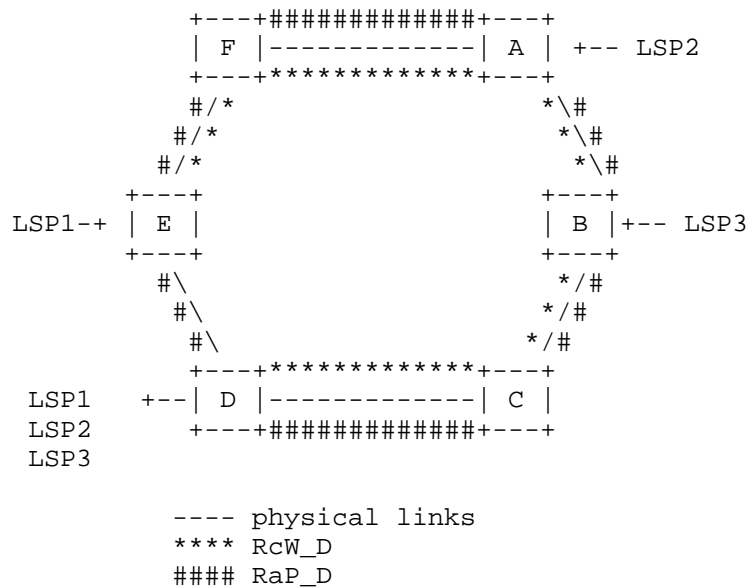


Figure 3. Ring tunnels in MSRP

Through these working and protection ring tunnels, LSPs which enter the ring from any node can reach any egress nodes on the ring, and are protected from failures on the ring.

#### 4.1.2. Label Assignment and Distribution

The ring tunnel labels are downstream-assigned labels as defined in [RFC3031]. The ring tunnel labels on each hop of the ring tunnel can be either configured statically, provisioned by a controller, or distributed dynamically via a control protocol. For an LSP which traverses the ring tunnel, the ingress ring node and the egress ring node are considered adjacent at the LSP layer, and LSP label needs to be allocated at these two ring nodes. The control plane for label distribution is outside the scope of this document.

#### 4.1.3. Forwarding Operation

When an MPLS-TP transport path, i.e. an LSP, enters the ring, the ingress node on the ring pushes the working ring tunnel label which is used to reach the specific egress node and sends the traffic to the next hop. The transit nodes on the working ring tunnel swap the ring tunnel labels and forward the packets to the next hop. When the packet arrives at the egress node, the egress node pops the ring tunnel label and forwards the packets based on the inner LSP label and service label. Figure 4 shows the label operation in the MPLS-TP

shared ring protection mechanism. Assume that LSP1 enters the ring at Node A and exits from Node D, and the following label operations are executed.

1. Ingress node: Packets of LSP1 arrive at Node A with a label stack [LSP1] and are supposed to be forwarded in the clockwise direction of the ring. The label of the clockwise working ring tunnel RcW\_D will be pushed at Node A, the label stack for the forwarded packet at Node A is changed to [RcW\_D(B)|LSP1].
2. Transit nodes: In this case, Node B and Node C forward the packets by swapping the working ring tunnel labels. For example, the label [RcW\_D(B)|LSP1] is swapped to [RcW\_D(C)|LSP1] at Node B.
3. Egress node: When the packet arrives at Node D (i.e. the egress node) with label stack [RcW\_D(D)|LSP1], Node D pops RcW\_D(D), and subsequently deals with the inner labels of LSP1.

```

+---+#####[RaP_D(F)]#####+---+
| F |-----| A | +-- LSP1
+---+*****[RcW_D(A)]*****+---+
#/*                               *\#
[RaP_D(E)]#/*[RcW_D(F)]          [RcW_D(B)]*\#[RaP_D(A)]
#/*                               *\#
+---+                               +---+
| E |                               | B |
+---+                               +---+
#\\                               */#
[RaP_D(D)]#\\                     [RxW_D(C)]*/#[RaP_D(B)]
#\\                               */#
+---+*****[RcW_D(D)]*****+---+
LSP1 +-- | D |-----| C |
+---+#####[RaP_D(C)]#####+---+

-----physical links          ***** RcW_D          ##### RaP_D

```

Figure 4. Label operation of MSRP

#### 4.2. Failure Detection

The MPLS-TP section layer OAM is used to monitor the connectivity between each two adjacent nodes on the ring using the mechanisms defined in [RFC6371]. Protection switching is triggered by the failure detected on the ring by the OAM mechanisms.

Two ports of a link form a Maintenance Entity Group (MEG), and an MEG end point (MEP) function is installed in each ring port. CC OAM

packets are periodically exchanged between each pair of MEPS to monitor the link health. Three consecutive lost CC packets MUST be interpreted as a link failure.

A node failure is regarded as the failure of two links attached to that node. The two nodes adjacent to the failed node detect the failure in the links that are connected to the failed node.

#### 4.3. Ring Protection

This section specifies the ring protection mechanisms in detail. In general, the description uses the clockwise working ring tunnel and the corresponding anti-clockwise protection ring tunnel as an example, but the mechanism is applicable in the same way to the anti-clockwise working and clockwise protection ring tunnels.

In a ring network, each working ring tunnel is associated with a protection ring tunnel in the opposite direction, and every node MUST obtain the ring topology either by configuration or via a topology discovery mechanism. The ring topology and the connectivity (Intact or Severed) between two adjacent ring nodes form the ring map. Each ring node maintains the ring map and uses it to perform ring protection switching.

Taking the topology in Figure 4 as an example, LSP1 enters the ring at Node A and leaves the ring at Node D. In normal state, LSP1 is carried by the clockwise working ring tunnel (RCW\_D) through the path A->B->C->D. The label operation is:

```
[LSP1](Payload) -> [RCW_D(B)|LSP1](NodeA) -> [RCW_D(C)|LSP1](NodeB)
-> [RCW_D(D)| LSP1](NodeC) -> [LSP1](Payload).
```

Then at node D the packet will be forwarded based on the label stack of LSP1.

Three typical ring protection mechanisms are described in this section: wrapping, short wrapping and steering. All nodes on the same ring MUST use the same protection mechanism. If the RPS protocol in any node detects RPS message with a protection switching mode that was not provisioned in that node a failure of protocol will be reported, and the protection mechanism will not be activated.

Wrapping ring protection: the node which detects a failure or accepts a switch request switches the traffic impacted by the failure or the switch request to the opposite direction (away from the failure). In this way, the impacted traffic is switched to the protection ring tunnel by the switching node upstream of the failure, then travels around the ring to the switching node downstream of the failure

through the protection ring tunnel, where it is switched back onto the working ring tunnel to reach the egress node.

Short wrapping ring protection provides some optimization to wrapping protection, in which the impacted traffic is only switched once to the protection ring tunnel by the switching node upstream to the failure. At the egress node, the traffic leave the ring from the protection ring tunnel. This can reduce the traffic detour of wrapping protection.

Steering ring protection implies that the node that detects a failure sends a request along the ring to the other node adjacent to the failure, and all nodes in the ring process this information. For the impacted traffic, the ingress node (which adds traffic to the ring) performs switching of the traffic from working to the protection ring tunnel, and the egress node will drop the traffic received from the protection ring tunnel.

The following sections describe these protection mechanisms in detail.

#### 4.3.1. Wrapping

With the wrapping mechanism, the protection ring tunnel is a closed ring identified by the egress node. As shown in Figure 4, the RaP\_D is the anticlockwise protection ring tunnel for the clockwise working ring tunnel RcW\_D. As specified in the following sections, the closed ring protection tunnel can protect both link failures and node failures. Wrapping can be applicable for the protection the p2mp LSPs on the ring, the details of which is outside the scope of this document.

##### 4.3.1.1. Wrapping for Link Failure

When a link failure between Node B and Node C occurs, if it is a bi-directional failure, both Node B and Node C can detect the failure via the OAM mechanism; if it is an uni-directional failure, one of the two nodes would detect the failure via the OAM mechanism. In both cases the node at the other side of the detected failure will be determined by the ring-map and informed using the Ring Protection Switch Protocol (RPS) which is specified in section 5. Then Node B switches the clockwise working ring tunnel (RcW\_D) to the anticlockwise protection ring tunnel (RaP\_D) and Node C switches anticlockwise protection ring tunnel(RaP\_D) back to the clockwise working ring tunnel (RcW\_D). The payload which enters the ring at Node A and leaves the ring at Node D follows the path A->B->A->F->E->D->C->D. The label operation is:

```
[LSP1](Payload) -> [RcW_D(B)|LSP1](Node A) -> [RaP_D(A)|LSP1](Node B)
-> [RaP_D(F)|LSP1](Node A) -> [RaP_D(E)|LSP1](Node F) ->
[RaP_D(D)|LSP1](Node E) -> [RaP_D(C)|LSP1](Node D) ->
[RcW_D(D)|LSP1](Node C) -> [LSP1](Payload).
```

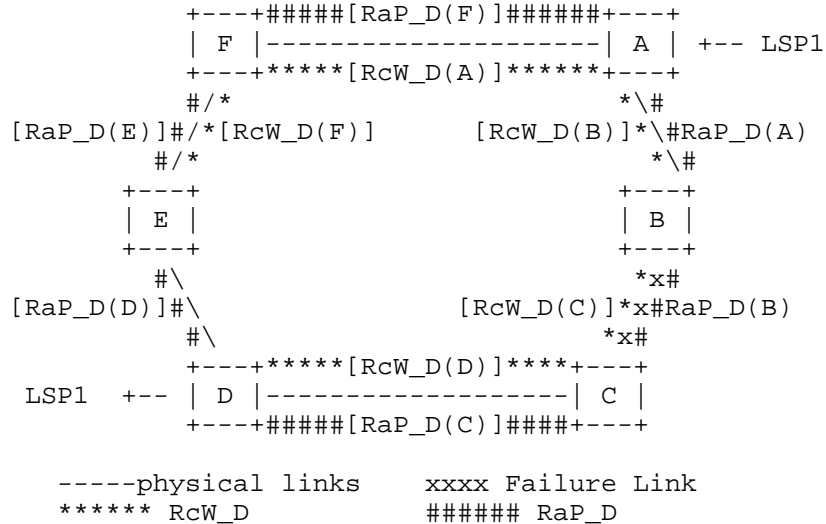


Figure 5. Wrapping for link failure

#### 4.3.1.2. Wrapping for Node Failure

As shown in Figure 6, when Node B fails, Node A detects the failure between A and B and switches the clockwise work ring tunnel (RcW\_D) to the anticlockwise protection ring tunnel (RaP\_D), Node C detects the failure between C and B and switches the anticlockwise protection ring tunnel (RaP\_D) to the clockwise working ring tunnel (RcW\_D). The node at the other side of the failed node will be determined by the ring-map and informed using the Ring Protection Switch Protocol (RPS) specified in section 5.

The payload which enters the ring at Node A and exits at Node D follows the path A->F->E->D->C->D. The label operation is:

```
[LSP1](Payload)-> [RaP_D(F)|LSP1](NodeA) -> [RaP_D(E)|LSP1](NodeF) ->
[RaP_D(D)|LSP1](NodeE) -> [RaP_D(C)|LSP1](NodeD) -> [RcW_D(D)|LSP1]
(NodeC) -> [LSP1](Payload).
```

In one special case where node D fails, all the ring tunnels with node D as egress will become unusable. The ingress node will update its ring map according to received RPS messages and determine that the egress node is not reachable thus it will not send traffic to

either the working or the protection tunnel. However, before the failure location information is propagated to all the ring nodes, the wrapping protection mechanism may cause temporary traffic loop: node C detects the failure and switches the traffic from the clockwise work ring tunnel (RcW\_D) to the anticlockwise protection ring tunnel (RaP\_D), node E also detects the failure and would switch the traffic from anticlockwise protection ring tunnel (RaP\_D) back to the clockwise work ring tunnel (RcW\_D). A possible mechanism to mitigate the temporary loop problem is: the TTL of the ring tunnel label is set to  $2*N$  by the ingress ring node of the traffic, where  $N$  is the number of nodes on the ring.

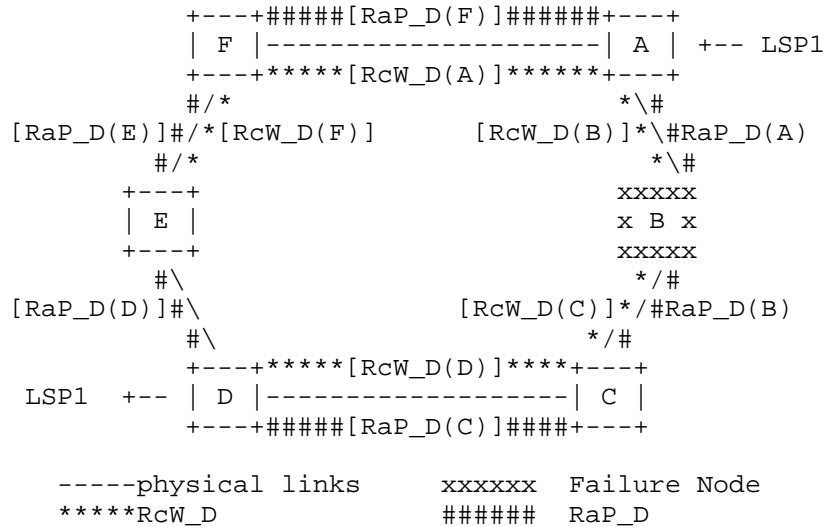


Figure 6. Wrapping for node failure

#### 4.3.2. Short Wrapping

With the wrapping protection scheme, protection switching is executed at both nodes adjacent to the failure, consequently the traffic will be wrapped twice. This mechanism will cause additional latency and bandwidth consumption when traffic is switched to the protection path.

With short wrapping protection, protection switching is executed only at the node upstream to the failure, and the packet leaves the ring in the protection ring tunnel at the egress node. This scheme can reduce the additional latency and bandwidth consumption when traffic is switched to the protection path. However the two directions of a protected bidirectional LSP are no longer co-routed under the protection switching conditions.

In the traditional wrapping solution, the protection ring tunnel is configured as a closed ring, while in the short wrapping solution, the protection ring tunnel is configured as ended at the egress node, which is similar to the working ring tunnel. Short wrapping is easy to implement in shared ring protection because both the working and protection ring tunnels are terminated on the egress nodes. Figure 7 shows the clockwise working ring tunnel and the anticlockwise protection ring tunnel with node D as the egress node.

#### 4.3.2.1. Short Wrapping for Link Failure

As shown in Figure 7, in normal state, LSP1 is carried by the clockwise working ring tunnel (RcW\_D) through the path A->B->C->D. When a link failure between Node B and Node C occurs, Node B switches the working ring tunnel RcW\_D to the protection ring tunnel RaP\_D in the opposite direction. The difference with wrapping occurs in the protection ring tunnel at the egress node. In short wrapping protection, RaP\_D ends in Node D and then traffic will be forwarded based on the LSP labels. Thus with the short wrapping mechanism, LSP1 will follow the path A->B->A->F->E->D when a link failure between Node B and Node C happens. The protection switch at node D is based on the information from its ring map and the information received via the RPS protocol.

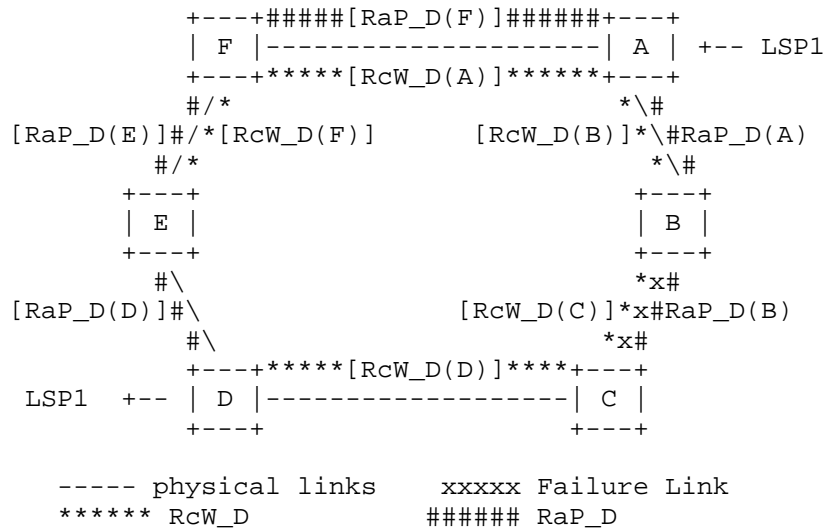


Figure 7. Short wrapping for link failure



#### 4.3.2.2. Short Wrapping for Node Failure

For the node failure which happens on a non-egress node, the short wrapping protection switching is similar to the link failure case as described in the previous section. This section specifies the scenario of an egress node failure.

As shown in Figure 8, LSP1 enters the ring on node A, and leaves the ring on node D. In normal state, LSP1 is carried by the clockwise working ring tunnel (RcW\_D) through the path A->B->C->D. When node D fails, traffic of LSP1 cannot be protected by any ring tunnels which use node D as the egress node. The ingress node will update its ring map according to received RPS messages and determine that the egress node is not reachable thus it will not send traffic to either the working or the protection tunnel. However, before the failure location information is propagated to all the ring nodes using the RPS protocol, node C switches all the traffic on the working ring tunnel RcW\_D to the protection ring tunnel RaP\_D in the opposite direction based on the information in the ring map. When the traffic arrives at node E which also detects the failure of node D, the protection ring tunnel RaP\_D cannot be used to forward traffic to node D. Since with short wrapping mechanism, protection switching can only be performed once from the working ring tunnel to the protection ring tunnel, thus node E MUST NOT switch the traffic which is already carried on the protection ring tunnel back to the working ring tunnel in the opposite direction. Instead, node E will discard the traffic received on RaP\_D locally. This can avoid the temporary traffic loop when the failure happens on the egress node of the ring tunnel. This also illustrates one of the benefits of having separate working and protection ring tunnels in each ring direction.

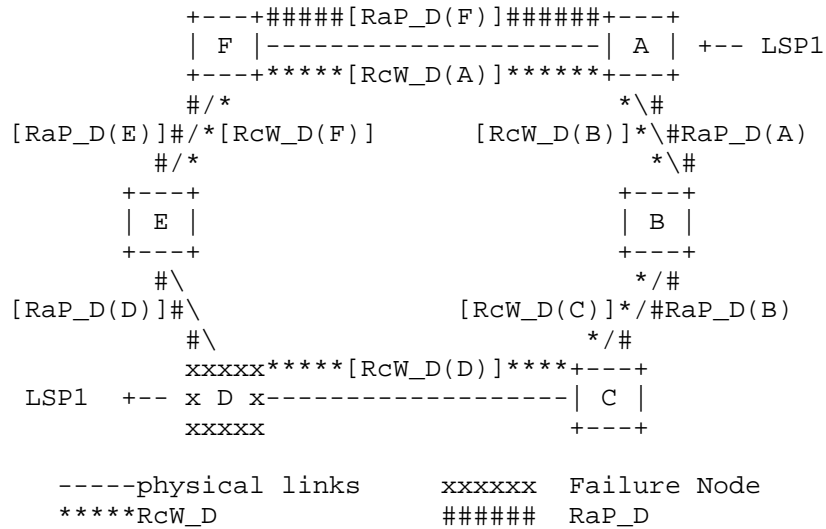


Figure 8. Short Wrapping for egress node failure

#### 4.3.3. Steering

With the steering protection mechanism, the ingress node (which adds traffic to the ring) perform switching from the working to the protection ring tunnel, and at the egress node the traffic leaves the ring from the protection ring tunnel.

When a failure occurs in the ring, the node which detects the failure with OAM mechanism sends the failure information in the opposite direction of the failure hop by hop along the ring using an RPS request message and the ring-map information. When a ring node receives the RPS message which identifies a failure, it can determine the location of the fault by using the topology information of the ring map and updates the ring map accordingly, then it can determine whether the LSPs entering the ring locally need to switchover or not. For LSPs that need to switchover, it will switch the LSPs from the working ring tunnels to their corresponding protection ring tunnels.

##### 4.3.3.1. Steering for Link Failure

```

Ring map of F                                +---LSP1
+---+---+---+---+---+   +----+ ###[RaP_D(F)]### +---/+ +---+---+---+---+---+
|F|A|B|C|D|E|F|         | F | ----- | A | |A|B|C|D|E|F|A|
+---+---+---+---+---+   +----+ ***[RcW_D(A)]*** +----+ +---+---+---+---+---+
|I|I|I|S|I|I|           +-----+ |I|I|S|I|I|I|I|
+---+---+---+---+---+   #/*                               *\\# +---+---+---+---+---+
[RaP_D(E)] #/* [RcW_D(B)] *\\# [RaP_D(A)]
#/* [RcW_D(F)] *\\#
+---+---+---+---+---+ #/* *\\#
|E|F|A|B|C|D|E| +---+ +-----+ --- LSP2
+---+---+---+---+---+ | E | +---+---+---+---+---+
|I|I|I|I|S|I| +---+ +-----+ |B|C|D|E|F|A|B|
+---+---+---+---+---+ #\\# */# +---+---+---+---+---+
[RaP_D(D)] #\\# [RcW_D(E)] [RcW_D(C)] */# |I|S|I|I|I|I|
#\\# */# +---+---+---+---+---+
#\\# */# [RaP_D(B)]
+---+---+---+---+---+ +----+ [RcW_D(D)] +----+ +---+---+---+---+---+
|D|E|F|A|B|C|D| +---+ | D | xxxxxxxxxxxxxxxxxxxxxxxx | C | |C|D|E|F|A|B|C|
+---+---+---+---+---+ LSP1 +----+ [RaP_D(C)] +----+ +---+---+---+---+---+
|I|I|I|I|I|S| LSP2 +-----+ |S|I|I|I|I|I|I|
+---+---+---+---+---+ +---+---+---+---+---+

----- physical links ***** RcW_D ##### RaP_D
I: Intact S: Severed

```

Figure 9. Steering operation and protection switching

As shown in Figure 9, LSP1 enters the ring from Node A while LSP2 enters the ring from Node B, and both of them have the same destination node D.

In normal state, LSP1 is carried by the clockwise working ring tunnel (RcW\_D) through the path A->B->C->D, the label operation is:  
 [LSP1](Payload) -> [RcW\_D(B)|LSP1](NodeA) -> [RcW\_D(C)| LSP1](NodeB)  
 -> [RcW\_D(D)|LSP1](NodeC) -> [LSP1](Payload) .

LSP2 is carried by the clockwise working ring tunnel (RcW\_D) through the path B->C->D, the label operation is: [LSP2](Payload) -> [RcW\_D(C)|LSP2](NodeB) -> [RcW\_D(D)|LSP2](NodeC) -> [LSP2](Payload) .

If the link between nodes C and D fails, according to the fault detection and distribution mechanisms, Node D will find out that there is a failure in the link between C and D, and it will update the link state of its ring topology, changing the link between C and D from normal to fault. In the direction that is opposite to the failure position, Node D will send the state report message to Node E, informing Node E of the fault between C and D, and E will update the link state of its ring topology accordingly, changing the link between C and D from normal to fault. In this way, the state report

message is sent hop by hop in the clockwise direction. Similar to Node D, Node C will send the failure information in the anti-clockwise direction.

When Node A receives the failure report message and updates the link state of its ring map, it is aware that there is a fault on the clockwise working ring tunnel to node D (RcW\_D), and LSP1 enters the ring locally and is carried by this ring tunnel, thus Node A will decide to switch the LSP1 onto the anticlockwise protection ring tunnel to node D (RaP\_D). After the switchover, LSP1 will follow the path A->F->E->D, the label operation is: [LSP1](Payload) -> [RaP\_D(F)|LSP1](NodeA) -> [RaP\_D(E)|LSP1](NodeF) -> [RaP\_D(D)|LSP1](NodeE) -> [LSP1](Payload).

The same procedure also applies to the operation of LSP2. When Node B updates the link state of its ring topology, and finds out that the working ring tunnel RcW\_D has failed, it will switch the LSP2 to the anticlockwise protection tunnel RaP\_D. After the switchover, LSP2 goes through the path B->A->F->E->D, and the label operation is: [LSP2](Payload) -> [RaP\_D(A)|LSP2](NodeB) -> [RaP\_D(F)|LSP2](NodeA) -> [RaP\_D(E)|LSP2](NodeF) -> [RaP\_D(D)|LSP2](NodeE) -> [LSP2](Payload).

Assume the link between nodes A and B breaks down, as shown in Figure 10. Similar to the above failure case, Node B will detect a fault in the link between A and B, and it will update its ring map, changing the link state between A and B from normal to fault. The state report message is sent hop by hop in the clockwise direction, notifying every node that there is a fault between node A and B, and every node updates the link state of its ring topology. As a result, Node A will detect a fault in the working ring tunnel to node D, and switch LSP1 to the protection ring tunnel, while Node B determine that the working ring tunnel for LSP2 still works fine, and will not perform the switchover.

```

                                     /-- LSP1
+-----+ +-----+ ##### [RaP_D(F)] ##### +---/ +-----+
|F|A|B|C|D|E|F| | F | ----- | A | |A|B|C|D|E|F|A|
+-----+ +-----+ ***[RcW_D(A)]*** +---+ +-----+
|I|S|I|I|I|I| #/* x |S|I|I|I|I|I|
+-----+ #/* x +-----+
[RaP_D(E)] #/*[RcW_D(F)] [RcW_D(B)]x [RaP_D(A)]
#/* x --- LSP2
+-----+ +-----+ +-----+
|E|F|A|B|C|D|E| | E | | B | |B|C|D|E|F|A|B|
+-----+ +-----+ +-----+
|I|I|S|I|I|I|I| #\* */# |I|I|I|I|I|S|
+-----+ #\*[RcW_D(E)] [RcW_D(C)] */# +-----+
[RaP_D(D)] #\* */# [RaP_D(B)]
+-----+ #\* */# +-----+
|D|E|F|A|B|C|D| +---+ ***[RcW_D(D)]*** +---+ |C|D|E|F|A|B|C|
+-----+ +--- | D | ----- | C | +-----+
|I|I|I|S|I|I| LSP1 +---+ ##### [RaP_D(C)] ##### +---+ |I|I|I|I|S|I|
+-----+ LSP2 +-----+
----- physical links ***** RcW_D ##### RaP_D

```

Figure 10. Steering operation and protection switching (2)

#### 4.3.3.2. Steering for Node Failure

For a node failure which happens on a non-egress node, steering protection switching is similar to the link failure case as described in the previous section.

If the failure occurs at the egress node of the LSP, the ingress node will update its ring map according to the received RPS messages, it will also determine that the egress node is not reachable after the failure, thus it will not send traffic to either the working or the protection tunnel, and a traffic loop can be avoided.

### 4.4. Interconnected Ring Protection

#### 4.4.1. Interconnected Ring Topology

Interconnected ring topology is widely used in MPLS-TP networks. For a given ring, the interconnection node acts as the egress node for that ring, meaning that all LSPs using the interconnection node as an egress from one specific ring to another will use the same group of ring tunnels within the ring. This document will discuss two typical interconnected ring topologies:

##### 1. Single-node interconnected rings

In single-node interconnected rings, the connection between the two rings is through a single node. Because the interconnection node is in fact a single point of failure, this topology should be avoided in real transport networks.

Figure 11 shows the topology of single-node interconnected rings. Node C is the interconnection node between Ring1 and Ring2.

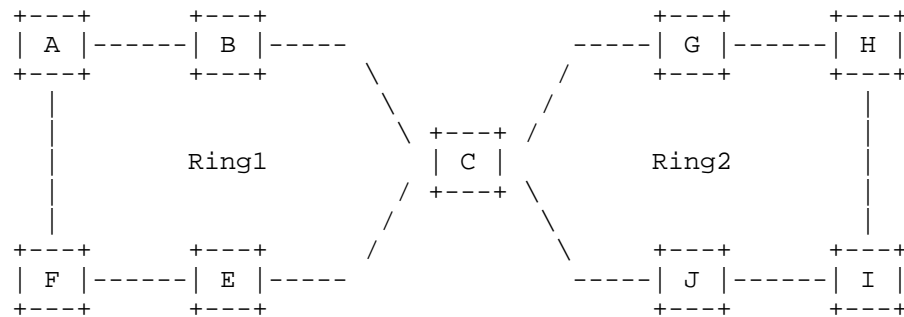


Figure 11. Single-node interconnected rings

## 2. Dual-node interconnected rings

In dual-node interconnected rings, the connection between the two rings is through two nodes. The two interconnection nodes belong to both interconnected rings. This topology can recover from one interconnection node failure.

Figure 12 shows the topology of dual-node interconnected rings. Nodes C and Node D are the interconnection nodes between Ring1 and Ring2.

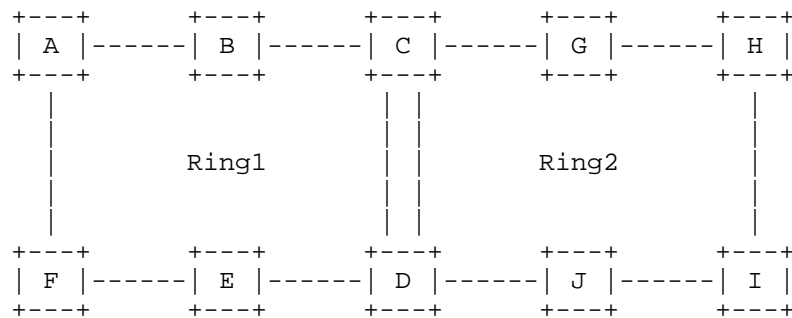


Figure 12. Dual-node interconnected rings

#### 4.4.2. Interconnected Ring Protection Mechanisms

Interconnected rings can be treated as two independent rings. Ring protection switching (RPS) protocol operates on each ring independently. A failure that happens in one ring only triggers protection switching in the ring itself and does not affect the other ring, unless the failure is on the interconnection node. In this way, protection switching on each ring is the same as the mechanisms described in section 4.3.

The service LSPs that traverse the interconnected rings use the ring tunnels in each ring, within a given ring, the tunnel is selected using normal ring selection procedures. The traversing LSPs are stitched on the interconnection node. On the interconnection node, the ring tunnel label of the source ring is popped, then LSP label is swapped, after that the ring tunnel label of the destination ring is pushed.

In the dual-node interconnected ring scenario, the two interconnection nodes can be managed as a virtual node group. In addition to the ring tunnels to each physical ring node, Each ring SHOULD assign the working and protection ring tunnels to the virtual interconnection node group. In addition, on both nodes in the virtual interconnection node group, the same LSP label is assigned for each traversed LSP. This way, any interconnection node in the virtual node group can terminate the working or protection ring tunnels targeted to the virtual node group, and stitch the service LSP from the source ring tunnel to the destination ring tunnel.

When the service LSP passes through the interconnected rings, the direction of the working ring tunnels used on both rings SHOULD be the same. In dual-node interconnected rings, this ensures that in normal state the traffic passes only one of the two interconnection nodes, and does not pass the link between the two interconnection

nodes. The traffic will then only be switched to the protection path if the interconnection node which is in working path fails. For example, if the service LSP uses the clockwise working ring tunnel on Ring1, when the service LSP leaves Ring1 and enters Ring2, the working ring tunnel used on Ring2 should also follow the clockwise direction.

#### 4.4.3. Ring Tunnels in Interconnected Rings

The same ring tunnels as described in section 4.1 are used in each ring of the interconnected rings. In addition, ring tunnels to the virtual interconnection node group are established on each ring of the interconnected rings, i.e.:

- o one clockwise working ring tunnel to the virtual interconnection node group
- o one anticlockwise protection ring tunnel to the virtual interconnection node group
- o one anticlockwise working ring tunnel to the virtual interconnection node group
- o one clockwise protection ring tunnel to the virtual interconnection node group

The ring tunnels to the virtual interconnection node group are shared by all LSPs that need to be forwarded to other rings. These ring tunnels can terminate at any node in the virtual interconnection node group.

For example, all the ring tunnels on Ring1 in Figure 13 are provisioned as follows:

- o To Node A: R1cW\_A, R1aW\_A, R1cP\_A, R1aP\_A
- o To Node B: R1cW\_B, R1aW\_B, R1cP\_B, R1aP\_B
- o To Node C: R1cW\_C, R1aW\_C, R1cP\_C, R1aP\_C
- o To Node D: R1cW\_D, R1aW\_D, R1cP\_D, R1aP\_D
- o To Node E: R1cW\_E, R1aW\_E, R1cP\_E, R1aP\_E
- o To Node F: R1cW\_F, R1aW\_F, R1cP\_F, R1aP\_F
- o To the virtual interconnection node group (including Node F and Node A): R1cW\_F&A, R1aW\_F&A, R1cP\_F&A, R1aP\_F&A



All the ring tunnels on Ring2 in Figure 13 are provisioned as follows:

- o To Node A: R2cW\_A, R2aW\_A, R2cP\_A, R2aP\_A
- o To Node F: R2cW\_F, R2aW\_F, R2cP\_F, R2aP\_F
- o To Node G: R2cW\_G, R2aW\_G, R2cP\_G, R2aP\_G
- o To Node H: R2cW\_H, R2aW\_H, R2cP\_H, R2aP\_H
- o To Node I: R2cW\_I, R2aW\_I, R2cP\_I, R2aP\_I
- o To Node J: R2cW\_J, R2aW\_J, R2cP\_J, R2aP\_J
- o To the virtual interconnection node group (including Node F and Node A): R2cW\_F&A, R2aW\_F&A, R2cP\_F&A, R2aP\_F&A

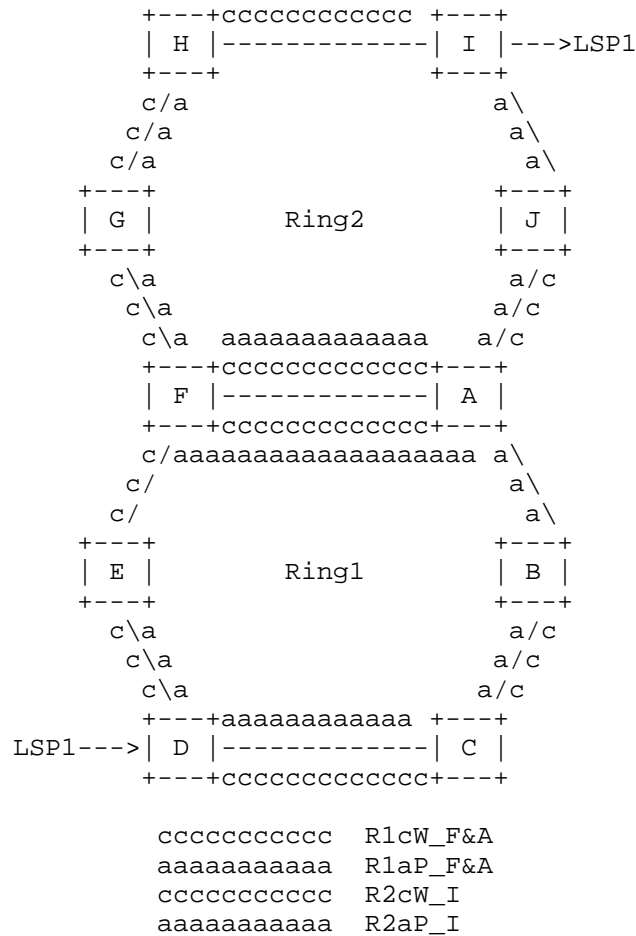


Figure 13. Ring tunnels for the interconnected rings

#### 4.4.4. Interconnected Ring Switching Procedure

As shown in Figure 13, for the service LSP1 which enters Ring1 at Node D and leaves Ring1 at Node F and continues to enter Ring2 at Node F and leaves Ring2 at Node I, the short wrapping protection scheme is described as below.

In normal state, LSP1 follows R1cW\_F&A in Ring1 and R2cW\_I in Ring2. At the interconnection node F, the label used for the working ring tunnel R1cW\_F&A in Ring1 is popped, the LSP label is swapped, and the label used for the working ring tunnel R2cW\_I in Ring2 will be pushed based the inner LSP label lookup. The working path that the service LSP1 follows is: LSP1->R1cW\_F&A (D->E->F)->R2cW\_I(F->G->H->I)->LSP1.

In case of link failure, for example, when a failure occurs on the link between Node F and Node E, Node E will detect the failure and execute protection switching as described in 4.3.2. The path that the service LSP1 follows after switching change to: LSP1->R1cW\_F&A(D->E)->R1aP\_F&A(E->D->C->B->A)->R2cW\_I(A->F->G->H->I)->LSP1.

In case of a non-interconnection node failure, for example, when the failure occurs at Node E in Ring1, Node D will detect the failure and execute protection switching as described in 4.3.2. The path that the service LSP1 follows after switching becomes: LSP1->R1aP\_F&A(D->C->B->A)->R2cW\_I(A->F->G->H->I)->LSP1.

In case of an interconnection node failure, for example, when the failure occurs at the interconnection Node F, Node E in Ring1 will detect the failure, and execute protection switching as described in 4.3.2. Node A in Ring2 will also detect the failure, and execute protection switching as described in 4.3.2. The path that the service traffic LSP1 follows after switching is: LSP1->R1cW\_F&A(D->E)->R1aP\_F&A(E->D->C->B->A)->R2aP\_I(A->J->I)->LSP1.

#### 4.4.5. Interconnected Ring Detection Mechanism

As shown in Figure 13, in normal state the service traffic LSP1 traverses D->E->F in Ring1 and F->G->H->I in Ring2. Node A and F are the interconnection nodes. When both the link between Node F and Node G and the link between Node F and Node A fail, the ring tunnel from Node F to Node I in Ring2 becomes unreachable. However, the other interconnection node A is still available, and LSP1 can still reach Node I via node A.

In order to achieve this, the interconnection nodes need to know the ring topology of each ring so that they can judge whether a node is reachable. This judgment is based on the knowledge of ring map and the fault location. The ring map can be obtained from the NMS or topology discovery mechanisms. The fault location can be obtained by transmitting the fault information around the ring. The nodes that detect the failure will transmit the fault information in the opposite direction hop by hop using the RPS protocol message. When the interconnection node receives the message that informs the failure, it will calculate the location of the fault according to the topology information that is maintained by itself and determines whether the LSPs entering the ring at itself can reach the destination. If the destination node is reachable, the LSP will leave the source ring and enter the destination ring. If the destination node is not reachable, the LSP will switch to the anticlockwise protection ring tunnel.

In Figure 13, Node F determines that the ring tunnel to Node I is unreachable, the service LSP1 for which the destination node on the ring2 is Node I MUST switch to the protection ring tunnel (R1aP\_F&A) and consequently the service traffic LSP1 traverses the interconnected rings at Node A. Node A will pop the ring tunnel label of Ring1 and push the ring tunnel label of Ring2 and send the traffic to Node I via ring tunnel (R2aW\_I).

## 5. Ring Protection Coordination Protocol

### 5.1. RPS and PSC Comparison on Ring Topology

This section provides comparison between RPS and PSC [RFC6378] [RFC6974] on ring topologies. This can be helpful to explain the reason of defining a new protocol for ring protection switching.

The PSC protocol [RFC6378] is designed for point-to-point LSPs, on which the protection switching can only be performed on one or both of the end points of the LSP. The RPS protocol is designed for ring tunnels, which consist of multiple ring nodes, and the failure could happen on any segment of the ring, thus RPS is capable of identifying and handling the different failures on the ring, and coordinating the protection switching behavior of all the nodes on the ring. As will be specified in the following sections, this is achieved with the introduction of the "Pass-Through" state for the ring nodes, and the location of the protection request is identified via the Node IDs in the RPS Request message.

Taking a ring topology with N nodes as example:

With the mechanism specified in [RFC6974], on every ring node, a linear protection configuration has to be provisioned with every other node in the ring, i.e. with (N-1) other nodes. This means that on every ring node there will be (N-1) instances of the PSC protocol. And in order to detect faults and to transport the PSC message, each instance shall have a MEP on the working path and a MEP on the protection path respectively. This means that every node on the ring needs to be configured with (N-1) \* 2 MEPs.

With the mechanism defined in this document, on every ring node there will only be a single instance of the RPS protocol. In order to detect faults and to transport the RPS message, each node only needs to have a MEP on the section to its adjacent nodes respectively. In this way, every ring node only needs to be configured with 2 MEPs.

As shown in the above example, RPS is designed for ring topologies and can achieve ring protection efficiently with minimum protection

instances and OAM entities, which meets the requirements on topology specific recovery mechanisms as specified in [RFC5654].

## 5.2. RPS Protocol

The Ring Protection Switch (RPS) Protocol defined in this section is used to coordinate the protection switching action of all the ring nodes in the same ring.

The protection operation of the ring tunnels is controlled with the help of the RPS protocol. The RPS processes in each of the individual ring nodes that form the ring MUST communicate using the G-ACh channel. The RPS protocol is applicable to all the three ring protection modes. This section takes the short-wrapping mechanism described in section 4.3.2 as an example.

The RPS protocol is used to distribute the ring status information and RPS requests to all the ring nodes. Changes in the ring status information and RPS requests can be initiated automatically based on link status or caused by external commands.

Each node on the ring is uniquely identified by assigning it a node ID. The node ID MUST be unique on each ring. The maximum number of nodes on the ring supported by the RPS protocol is 127. The node ID SHOULD be independent of the order in which the nodes appear on the ring. The node ID is used to identity the source and destination nodes of each RPS request.

Every node obtains the ring topology either by configuration or via some topology discovery mechanism. The ring map consists of the ring topology information, and connectivity status (Intact or Severed) between the adjacent ring nodes, which is determined via the OAM message exchanged between the adjacent nodes. The ring map is used by every ring node to determine the switchover behavior of the ring tunnels.

As shown in Figure 14, when no protection switching is active on the ring, each node MUST send RPS requests with No Request (NR) to its two adjacent nodes periodically. The transmission interval of RPS requests is specified in section 5.2.1.

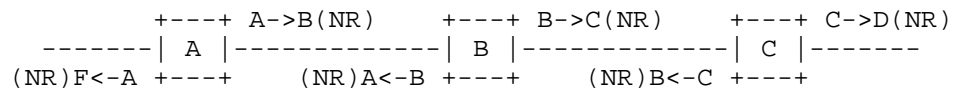


Figure 14. RPS communication between the ring nodes in case of no failure in the ring

As shown in Figure 15, When a node detects a failure and determines that protection switching is required, it MUST send the appropriate RPS request in both directions to the destination node. The destination node is the other node that is adjacent to the identified failure. When a node that is not the destination node receives an RPS request and it has no higher priority local request, it MUST transfer in the same direction the RPS request as received. In this way, the switching nodes can maintain RPS protocol communication in the ring. The RPS request MUST be terminated by the destination node of the message. If an RPS request with the node itself set as the source node is received, this message MUST be dropped and not be forwarded to next node.

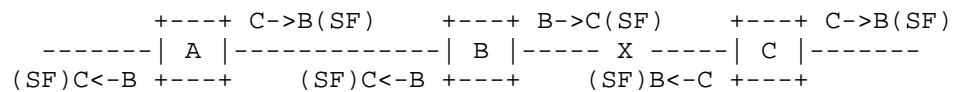


Figure 15. RPS communication between the ring nodes in case of failure between nodes B and C

Note that in the case of a bidirectional failure such as a cable cut, the two adjacent nodes detect the failure and send each other an RPS request in opposite directions.

- o In rings utilizing the wrapping protection, each node detects the failure or receives the RPS request as the destination node MUST perform the switch from/to the working ring tunnels to/from the protection ring tunnels if it has no higher priority active RPS request.
- o In rings utilizing the short wrapping protection, each node detects the failure or receives the RPS request as the destination node MUST perform the switch only from the working ring tunnels to the protection ring tunnels.
- o In rings utilizing the steering protection. When a ring switch is required, any node MUST perform the switches if its added/dropped traffic is affected by the failure. Determination of the affected traffic MUST be performed by examining the RPS requests (indicating the nodes adjacent to the failure or failures) and the stored ring map (indicating the relative position of the failure and the added traffic destined towards that failure).

When the failure has cleared and the Wait-to-Restore (WTR) timer has expired, the nodes which generate the RPS requests MUST drop their respective switches and MUST generate an RPS request carrying the NR code. The node receiving from both directions such an RPS request MUST drop its protection switches.

A protection switch MUST be initiated by one of the criteria specified in Section 5.3. A failure of the RPS protocol or controller MUST NOT trigger a protection switch.

Ring switches MUST be preempted by higher priority RPS requests. For example, consider a protection switch that is active due to a manual switch request on the given link, and another protection switch is required due to a failure on another link. Then an RPS request MUST be generated, the former protection switch MUST be dropped, and the latter protection switch established.

The shared ring protection mechanism supports multiple protection switches in the ring, resulting in the ring being segmented into two or more separate segments. This may happen when several RPS requests of the same priority exist in the ring due to multiple failures or external switch commands.

Proper operation of the MSRP mechanism relies on all nodes using their ring map to determine the state of the ring (nodes and links). In order to accommodate ring state knowledge, during a protection switch the RPS requests MUST be sent in both directions.

#### 5.2.1. Transmission and Acceptance of RPS Requests

A new RPS request MUST be transmitted immediately when a change in the transmitted status occurs.

The first three RPS protocol messages carrying new RPS request MUST be transmitted as fast as possible. For fast protection switching within 50 ms, the interval of the first three RPS protocol messages SHOULD be 3.3 ms. The successive RPS requests SHOULD be transmitted with the interval of 5 seconds. A ring node which is not the destination of the received RPS message MUST forward it to the next node along the ring immediately.

#### 5.2.2. RPS PDU Format

Figure 16 depicts the format of an RPS packet that is sent on the G-ACh. The Channel Type field is set to indicate that the message is an RPS message.

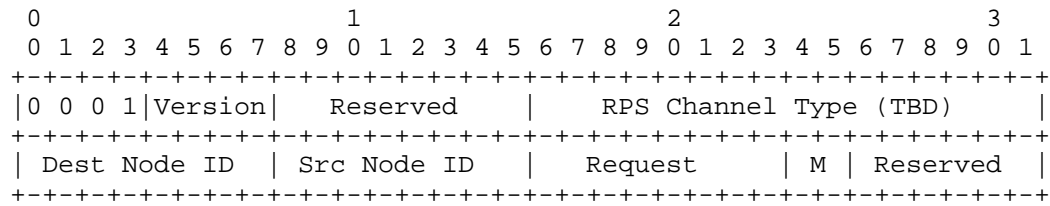


Figure 16. G-ACh RPS Packet Format

The following fields MUST be provided:

- o Destination Node ID: The destination node ID MUST always be set to value of the node ID of the adjacent node. The Node ID MUST be unique on each ring. Valid destination node ID values are 1-127.
- o Source Node ID: The source node ID MUST always be set to the ID value of the node generating the RPS request. The Node ID MUST be unique on each ring. Valid source node ID values are 1-127.
- o Protection Switching Mode (M): This 2-bit field indicates the protection switching mode used by the sending node of the RPS message. This can be used to check that the ring nodes on the same ring use the same protection switching mechanism. The defined values of the M field are listed as below:

Bits (MSB-LSB)	Protecton Switching Mode
0 0	Reserved
0 1	Wrapping
1 0	Short Wrapping
1 1	Steering

- o RPS request code: A code consisting of eight bits as specified below:



Bits (MSB - LSB)	Condition, State or external Request	Priority
0 0 0 0 1 1 1 1	Lockout of Protection (LP)	highest
0 0 0 0 1 1 0 1	Forced Switch (FS)	
0 0 0 0 1 0 1 1	Signal Fail (SF)	
0 0 0 0 0 1 1 0	Manual Switch (MS)	
0 0 0 0 0 1 0 1	Wait-To-Restore (WTR)	
0 0 0 0 0 0 1 1	Exercise (EXER)	
0 0 0 0 0 0 0 1	Reverse Request (RR)	
0 0 0 0 0 0 0 0	No Request (NR)	lowest

### 5.2.3. Ring Node RPS States

Idle state: A node is in the idle state when it has no RPS request and is sending and receiving NR code to/from both directions.

Switching state: A node not in the idle or pass-through states is in the switching state.

Pass-through state: A node is in the pass-through state when its highest priority RPS request is a request not destined to it or generated by it. The pass-through is bidirectional.

#### 5.2.3.1. Idle State

A node in the idle state MUST generate the NR request in both directions.

A node in the idle state MUST terminate RPS requests flow in both directions.

A node in the idle state MUST block the traffic flow on protection ring tunnels in both directions.

#### 5.2.3.2. Switching State

A node in the switching state MUST generate RPS request to its adjacent node with its highest RPS request code in both directions when it detects a failure or receives an external command.

In bidirectional failure condition, both of the nodes adjacent to the failure detect the failure and send the RPS request in both directions with the destination set to each other, while each node can only receive the RPS request via the long path, the message sent via the short path will get lost due to the bidirectional failure.

Here the short path refers to the shorter path on the ring between the source and destination node of the RPS request, and the long path refers to the longer path on the ring between the source and destination node of the RPS request. Upon receipt of the RPS request on the long path, the destination node of the RPS request MUST send RPS request with its highest request code periodically along the long path to the other node adjacent to the failure.

In unidirectional failure condition, the node which detects the failure MUST send the RPS request in both directions with the destination node set to the other node adjacent to the failure. The destination node of the RPS request cannot detect the failure itself but will receive RPS request from both the short path and the long path. The destination node MUST acknowledge the received RPS request by replying an RPS request with the RR code on the short path, and an RPS request with the received RPS request code on the long path. Accordingly, when the node which detects the failure receives RPS request with RR code on the short path, then the RPS request received from the same node along the long path SHOULD be ignored.

A node in the switching state MUST terminate the received RPS requests in both directions and not forward it further along the ring.

The following switches as defined in section 5.3.1 MUST be allowed to coexist:

- o LP and LP
- o FS and FS
- o SF and SF
- o FS and SF

When multiple MS RPS requests exist at the same time addressing different links and there is no higher priority request on the ring, no switch SHOULD be executed and existing switches MUST be dropped. The nodes MUST still signal RPS request with the MS code.

Multiple EXER requests MUST be allowed to coexist in the ring.

A node in a ring switching state that receives the external command LP for the affected link MUST drop its switch and MUST signal NR for the locked link if there is no other RPS request on another link. The node still SHOULD signal relevant RPS request for another link.

#### 5.2.3.3. Pass-through State

When a node is in a pass-through state, it MUST transfer the received RPS Request unchanged in the same direction.

When a node is in a pass-through state, it MUST enable the traffic flow on protection ring tunnels in both directions.

#### 5.2.4. RPS State Transitions

All state transitions are triggered by an incoming RPS request change, a WTR expiration, an externally initiated command, or locally detected MPLS-TP section failure conditions.

RPS requests due to a locally detected failure, an externally initiated command, or received RPS request shall preempt existing RPS requests in the prioritized order given in Section 5.2.2, unless the requests are allowed to coexist.

##### 5.2.4.1. Transitions Between Idle and Pass-through States

The transition from the idle state to pass-through state MUST be triggered by a valid RPS request change, in any direction, from the NR code to any other code, as long as the new request is not destined to the node itself. Both directions move then into a pass-through state, so that, traffic entering the node through the protection ring tunnels are transferred transparently through the node.

A node MUST revert from pass-through state to the idle state when RPS request with NR code is received in both directions. Then both directions revert simultaneously from the pass-through state to the idle state.

##### 5.2.4.2. Transitions Between Idle and Switching States

Transition of a node from the Idle state to the Switching state MUST be triggered by one of the following conditions:

- o A valid RPS request change from the NR code to any code received on either the long or the short path and is destined to this node
- o An externally initiated command for this node
- o The detection of an MPLS-TP section layer failure at this node

Actions taken at a node in the Idle state upon transition to the switching state are:

- o For all protection switch requests, except EXER and LP, the node MUST execute the switch
- o For EXER, and LP, the node MUST signal appropriate request but not execute the switch

In one of the following conditions, transition from the Switching state to the Idle state MUST be triggered:

- o On node which triggers the protection switching, when the WTR time expires or an externally initiated command is cleared, the node MUST transit from Switching state to Idle State and signal the NR code using RPS message in both directions.
- o On node which enters the switching state due to the received RPS request: Upon reception of the NR code from both directions, the head-end node MUST drop its switch, transition to Idle State and signal the NR code in both directions.

#### 5.2.4.3. Transitions Between Switching States

When a node that is currently executing any protection switch receives a higher priority RPS request (due to a locally detected failure, an externally initiated command, or a ring protection switch request destined to it) for the same link, it MUST update the priority of the switch it is executing to the priority of the received RPS request.

When a failure condition clears at a node, the node MUST enter WTR condition and remain in it for the appropriate time-out interval, unless:

- o A different RPS request with a higher priority than WTR is received
- o Another failure is detected
- o An externally initiated command becomes active

The node MUST send out a WTR code on both the long and short paths.

When a node that is executing a switch in response to incoming SF RPS request (not due to a locally detected failure) receives a WTR code (unidirectional failure case), it MUST send out RR code on the short path and the WTR on the long path.

#### 5.2.4.4. Transitions Between Switching and Pass-through States

When a node that is currently executing a switch receives an RPS request for a non-adjacent link of higher priority than the switch it is executing, it **MUST** drop its switch immediately and enter the pass-through state.

The transition of a node from pass-through to switching state **MUST** be triggered by:

- o An equal priority, a higher priority, or an allowed coexisting externally initiated command
- o The detection of an equal priority, a higher priority, or an allowed coexisting automatic initiated command
- o The receipt of an equal, a higher priority, or an allowed coexisting RPS request destined to this node

### 5.3. RPS State Machine

#### 5.3.1. Switch Initiation Criteria

##### 5.3.1.1. Administrative Commands

Administrative commands can be initiated by the network operator through the Network Management System (NMS). The operator command may be transmitted to the appropriate node via the MPLS-TP RPS message.

The following commands can be transferred by the RPS message:

- o Lockout of Protection (LP): This command prevents any protection activity and prevents using ring switches anywhere in the ring. If any ring switches exist in the ring, this command causes the switches to drop.
- o Forced Switch to protection (FS): This command performs the ring switch of normal traffic from the working entity to the protection entity for the link between the node at which the command is initiated and the adjacent node to which the command is directed. This switch occurs regardless of the state of the MPLS-TP section for the requested link, unless a higher priority switch request exists.
- o Manual Switch to protection (MS): This command performs the ring switch of the normal traffic from the working entity to the protection entity for the link between the node at which the

command is initiated and the adjacent node to which the command is directed. This occurs if the MPLS-TP section for the requested link is not satisfying an equal or higher priority switch request.

- o Exercise - Ring (EXER): This command exercises ring protection switching on the addressed link without completing the actual switch. The command is issued and the responses (RR) are checked, but no normal traffic is affected.

The following commands are not transferred by the RPS message:

- o Clear: This command clears the administrative command and Wait-To-Restore timer (WTR) at the node to which the command was addressed. The node to node signaling after the removal of the externally initiated commands is performed using the no-request code (NR).
- o Lockout of Working (LW): This command prevents the normal traffic transported over the addressed link from being switched to the protection entity by disabling the node's capability of requesting switch for this link in case of failure. If any normal traffic is already switched on the protection entity, the switch is dropped. If no other switch requests are active on the ring, the no-request code (NR) is transmitted. This command has no impact on any other link. If the node receives the switch request from the adjacent node from any side it will perform the requested switch. If the node receives the switch request addressed to the other node, it will enter the pass-through state.

#### 5.3.1.2. Automatically Initiated Commands

Automatically initiated commands can be initiated based on MPLS-TP section layer OAM indication and the received switch requests.

The node can initiate the following switch requests automatically:

- o Signal Fail (SF): This command is issued when the MPLS-TP section layer OAM detects signal failure condition.
- o Wait-To-Restore (WTR): This command is issued when MPLS-TP section detects that the SF condition has cleared. It is used to maintain the state during the WTR period unless it is preempted by a higher priority switch request. The WTR time may be configured by the operator in 1 minute steps between 0 and 12 minutes; the default value is 5 minutes.

- o Reverse Request (RR): This command is transmitted to the source node of the received RPS message over the short path as an acknowledgment for receiving the switch request.

#### 5.3.2. Initial States

This section describes the possible states of a ring node, the corresponding action of the working and protection ring tunnels on the node, and the RPS request which should be generated in that state.

State		Signaled RPS
A	Idle Working: no switch Protection: no switch	NR
B	Pass-through Working: no switch Protection: pass through	N/A
C	Switching - LP Working: no switch Protection: no switch	LP
D	Idle - LW Working: no switch Protection: no switch	NR
E	Switching - FS Working: switched Protection: switched	FS
F	Switching - SF Working: switched Protection: switched	SF
G	Switching - MS Working: switched Protection: switched	MS
H	Switching - WTR Working: switched Protection: switched	WTR
I	Switching - EXER Working: no switch Protection: no switch	EXER

### 5.3.3. State transitions When Local Request is Applied

In the state description below 'O' means that new local request will be rejected because of exiting request.

```
=====
Initial state      New request      New state
-----
```



A (Idle)	LP	C (Switching - LP)
	LW	D (Idle - LW)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	Recover from SF	N/A
	MS	G (Switching - MS)
	Clear	N/A
	WTR expires	N/A
	EXER	I (Switching - EXER)
Initial state	New request	New state
-----	-----	-----
B (Pass-through)	LP	C (Switching - LP)
	LW	B (Pass-through)
	FS	O - if current state is due to LP sent by another node
		E (Switching - FS) - otherwise
	SF	O - if current state is due to LP sent by another node
		F (Switching - SF) - otherwise
	Recover from SF	N/A
	MS	O - if current state is due to LP, SF or FS sent by another node
		G (Switching - MS) - otherwise
	Clear	N/A
	WTR expires	N/A
	EXER	O
Initial state	New request	New state
-----	-----	-----
C (Switching - LP)	LP	N/A
	LW	O
	FS	O
	SF	O
	Recover from SF	N/A
	MS	O
	Clear	A (Idle) - if there is no failure in the ring
		F (Switching - SF) - if there is a failure at this node
		B (Pass-through) - if there is a failure at another node
	WTR expires	N/A
	EXER	O
Initial state	New request	New state
-----	-----	-----

D (Idle - LW)	LP	C (Switching - LP)
	LW	N/A - if on the same link
		D (Idle - LW) - if on another link
	FS	O - if on the same link
		E (Switching - FS) - if on another link
	SF	O - if on the addressed link
		F (Switching - SF) - if on another link
	Recover from SF	N/A
	MS	O - if on the same link
		G (Switching - MS) - if on another link
	Clear	A (Idle) - if there is no failure on addressed link
		F (Switching - SF) - if there is a failure on this link
	WTR expires	N/A
	EXER	O
=====		
Initial state	New request	New state
-----	-----	-----
E (Switching - FS)	LP	C (Switching - LP)
	LW	O - if on another link
		D (Idle - LW) - if on the same link
	FS	N/A - if on the same link
		E (Switching - FS) - if on another link
	SF	O - if on the addressed link
		E (Switching - FS) - if on another link
	Recover from SF	N/A
	MS	O
	Clear	A (Idle) - if there is no failure in the ring
		F (Switching - SF) - if there is a failure at this node
		B (Pass-through) - if there is a failure at another node
	WTR expires	N/A
	EXER	O
=====		
Initial state	New request	New state
-----	-----	-----
F (Switching - SF)	LP	C (Switching - LP)
	LW	O - if on another link

		D (Idle - LW) - if on the same link
	FS	E (Switching - FS)
	SF	N/A - if on the same link
		F (Switching - SF) - if on another link
	Recover from SF	H (Switching - WTR)
	MS	O
	Clear	N/A
	WTR expires	N/A
	EXER	O
=====		
Initial state	New request	New state
-----	-----	-----
G (Switching - MS)	LP	C (Switching - LP)
	LW	O - if on another link
		D (Idle - LW) - if on the same link
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	Recover from SF	N/A
	MS	N/A - if on the same link
		G (Switching - MS) - if on another link release the switches but signal MS
	Clear	A
	WTR expires	N/A
	EXER	O
=====		
Initial state	New request	New state
-----	-----	-----
H (Switching - WTR)	LP	C (Switching - LP)
	LW	D (Idle - W)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	Recover from SF	N/A
	MS	G (Switching - MS)
	Clear	A
	WTR expires	A
	EXER	O
=====		
Initial state	New request	New state
-----	-----	-----
I (Switching - EXER)	LP	C (Switching - LP)
	LW	D (idle - W)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	Recover from SF	N/A

MS	G (Switching - MS)
Clear	A
WTR expires	N/A
EXER	N/A - if on the same link
	I (Switching - EXER)

=====

#### 5.3.4. State Transitions When Remote Request is Applied

The priority of a remote request does not depend on the side from which the request is received.

Initial state	New request	New state
A (Idle)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	A (Idle)

Initial state	New request	New state
B (Pass-through)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	E (Switching - FS) - otherwise N/A - cannot happen when there is LP request in the ring
	MS	F (Switching - SF) - otherwise N/A - cannot happen when there is LP, FS or SF request in the ring
	WTR	G (Switching - MS) - otherwise N/A - cannot happen when there is LP, FS, SF or MS request in the ring
	EXER	N/A - cannot happen when there is LP, FS, SF, MS or WTR request in the ring
		I (Switching - EXER) - otherwise
	RR	N/A
	NR	A (Idle) - if received from

both sides		
Initial state	New request	New state
C (Switching - LP)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	N/A - cannot happen when there is LP request in the ring
	MS	N/A - cannot happen when there is LP request in the ring
	WTR	N/A
	EXER	N/A - cannot happen when there is LP request in the ring
	RR	C (Switching - LP)
	NR	N/A
Initial state	New request	New state
D (Idle - LW)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	D (Idle - LW)
Initial state	New request	New state
E (Switching - FS)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	E (Switching - FS)
	MS	N/A - cannot happen when there is FS request in the ring
	WTR	N/A
	EXER	N/A - cannot happen when there is FS request in the ring
	RR	E (Switching - FS)
	NR	N/A
Initial state	New request	New state
F (Switching - SF)	LP	C (Switching - LP)
	FS	F (Switching - SF)
	SF	F (Switching - SF)
	MS	N/A - cannot happen when there is SF request in the ring

	WTR	N/A
	EXER	N/A - cannot happen when there is SF request in the ring
	RR	F (Switching - SF)
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
G (Switching - MS)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS) - release the switches but signal MS
	WTR	N/A
	EXER	N/A - cannot happen when there is MS request in the ring
	RR	G (Switching - MS)
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
H (Switching - WTR)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	H (Switching - WTR)
	EXER	N/A - cannot happen when there is WTR request in the ring
	RR	H (Switching - WTR)
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
I (Switching - EXER)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	I (Switching - EXER)
	NR	N/A
=====		

### 5.3.5. State Transitions When Request Addresses to Another Node is Received

The priority of a remote request does not depend on the side from which the request is received.

=====		
Initial state	New request	New state
-----	-----	-----
A (Idle)	LP	B (Pass-through)
	FS	B (Pass-through)
	SF	B (Pass-through)
	MS	B (Pass-through)
	WTR	B (Pass-through)
	EXER	B (Pass-through)
	RR	N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
B (Pass-through)	LP	B (Pass-through)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	B (Pass-through) - otherwise
		N/A - cannot happen when there is LP request in the ring
	MS	B (Pass-through) - otherwise
		N/A - cannot happen when there is LP, FS or SF request in the ring
	WTR	B (Pass-through) - otherwise
		N/A - cannot happen when there is LP, FS, SF or MS request in the ring
	EXER	B (Pass-through) - otherwise
		N/A - cannot happen when there is LP, FS, SF, MS or WTR request in the ring
	RR	B (Pass-through) - otherwise
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
C (Switching - LP)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	N/A - cannot happen when there

	MS	is LP request in the ring N/A - cannot happen when there
	WTR	is LP request in the ring N/A - cannot happen when there
	EXER	is LP in the ring N/A - cannot happen when there
	RR	is LP request in the ring N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
D (Idle - LW)	LP	B (Pass-through)
	FS	B (Pass-through)
	SF	B (Pass-through)
	MS	B (Pass-through)
	WTR	B (Pass-through)
	EXER	B (Pass-through)
	RR	N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
E (Switching - FS)	LP	B (Pass-through)
	FS	E (Switching - FS)
	SF	E (Switching - FS)
	MS	N/A - cannot happen when there
		is FS request in the ring
	WTR	N/A - cannot happen when there
		is FS request in the ring
	EXER	N/A - cannot happen when there
		is FS request in the ring
	RR	N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
F (Switching - SF)	LP	B (Pass-through)
	FS	F (Switching - SF)
	SF	F (Switching - SF)
	MS	N/A - cannot happen when there
		is SF request in the ring
	WTR	N/A - cannot happen when there
		is SF request in the ring
	EXER	N/A - cannot happen when there
		is SF request in the ring
	RR	N/A
	NR	N/A



=====		
Initial state	New request	New state
-----	-----	-----
G (Switching - MS)	LP	B (Pass-through)
	FS	B (Pass-through)
	SF	B (Pass-through)
	MS	G (Switching - MS) - release the switches but signal MS
	WTR	N/A - cannot happen when there is MS request in the ring
	EXER	N/A - cannot happen when there is MS request in the ring
	RR	N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
H (Switching - WTR)	LP	B (Pass-through)
	FS	B (Pass-through)
	SF	B (Pass-through)
	MS	B (Pass-through)
	WTR	N/A
	EXER	N/A - cannot happen when there is WTR request in the ring
	RR	N/A
	NR	N/A
=====		
Initial state	New request	New state
-----	-----	-----
I (Switching - EXER)	LP	B (Pass-through)
	FS	B (Pass-through)
	SF	B (Pass-through)
	MS	B (Pass-through)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	N/A
=====		

## 6. IANA Considerations

IANA is requested to administer the assignment of new values defined in this document and listed in the sections below.

### 6.1. G-ACh Channel Type

The Channel Types for the Generic Associated channel (GACH) are allocated from the IANA PW Associated Channel Type registry defined in [RFC4446] and updated by [RFC5586].

IANA is requested to allocate a new GACH Channel Type as follows:

Value	Description	Reference
TBD	Ring Protection Switching Protocol (RPS)	this document

### 6.2. RPS Request Codes

IANA is requested to create a new sub-registry under the "Multiprotocol Label Switching (MPLS) Operations, Administration, and Management (OAM) Parameters" registry called the "MPLS RPS Request Code Registry". All code points within this registry shall be allocated according to the "Specification Required" procedure as specified in [RFC5226].

The RPS Request Field is 8 bits, the allocated values are as follows:

Value	Description	Reference
0	No Request (NR)	this document
1	Reverse Request (RR)	this document
2	unassigned	
3	Exercise (EXER)	this document
4	unassigned	
5	Wait-To-Restore (WTR)	this document
6	Manual Switch (MS)	this document
7-10	unassigned	
11	Signal Fail (SF)	this document
12	unassigned	
13	Forced Switch (FS)	this document
14	unassigned	
15	Lockout of Protection (LP)	this document
16-254	unassigned	
255	Reserved	

## 7. Operational Considerations

This document describes three protection modes of the RPS protocol. Operators could choose the appropriate protection mode according to their network and service requirement.

Wrapping mode provides a ring protection mechanism in which the protected traffic will reach every node of the ring, and is applicable to protect both the point-to-point LSPs and LSPs which needs to be dropped in several ring nodes, i.e. the point-to-multipoint applications. When protection is in active, the protected traffic is switched (wrapped) to/from the protection ring tunnel at both sides of the defective link/node. Due to the wrapping the additional propagation delay and bandwidth consumption of the protection tunnel are considerable. For bidirectional LSP, the protected traffic in both directions is co-routed.

Short wrapping mode provides a ring protection mechanism which can be used to protect only point-to-point LSPs. When protection is in active, the protected traffic wrapped to the protection ring tunnel at the defective link/node and leaves the ring when the protection ring tunnel reach the egress node. Compared with wrapping mode, short wrapping can reduce the propagation latency and bandwidth consumption of the protection tunnel. However the two directions of a protected bidirectional LSP are not totally co-routed.

Steering mode provides a ring protection mechanism that can be used to protect only point-to-point LSPs. When protection is in active, the protected traffic is switched to the protection ring tunnel at the ingress node and leaves the ring when the protection ring tunnel reach the egress node. Steering mode has the least propagation delay and bandwidth consumption of the three modes, and the two directions of a protected bidirectional LSP can be kept co-routed.

Note that only one protection mode can be provisioned in the whole ring for all protected traffic.

## 8. Security Considerations

MPLS-TP is a subset of MPLS and so builds upon many of the aspects of the security model of MPLS. Please refer to [RFC5920] for generic MPLS security issues and methods for securing traffic privacy and integrity.

The RPS message defined in this document is used for protection coordination on the ring, if it is injected or modified by an attacker, the ring nodes might not agree on the protection action, and the improper protection switching action may cause temporary break to services traversing the ring. It is important that the RPS message is used within a trusted MPLS-TP network domain as described in [RFC6941].

The RPS message is carried in the G-ACh [RFC5586], so it is dependent on the security of the G-ACh itself. The G-ACh is a generalization of the Associated Channel defined in [RFC4385]. Thus, this document relies on the security mechanisms provided for the Associated Channel as described in those two documents.

As described in the security considerations of [RFC6378], the G-ACh is essentially connection oriented so injection or modification of control messages requires the subversion of a transit node. Such subversion is generally considered hard in connection oriented MPLS networks and impossible to protect against at the protocol level. Management level techniques are more appropriate. The procedures and protocol extensions defined in this document do not affect the security model of MPLS-TP linear protection as defined in [RFC6378].

## 9. Contributing Authors

Kai Liu  
Huawei Technologies  
Email: alex.liukai@huawei.com

Jia He  
Huawei Technologies  
Email: hejia@huawei.com

Fang Li  
China Academy of Telecommunication Research MIIT., China  
Email: lifang@catr.cn

Jian Yang  
ZTE Corporation P.R.China  
Email: yang.jian90@zte.com.cn

Junfang Wang  
Fiberhome Telecommunication Technologies Co., LTD.  
Email: wjf@fiberhome.com.cn

Wen Ye  
China Mobile  
Email: yewen@chinamobile.com

Minxue Wang  
China Mobile  
Email: wangminxue@chinamobile.com

Sheng Liu  
China Mobile  
Email: liusheng@chinamobile.com

Guanghui Sun  
Huawei Technologies  
Email: sunguanghui@huawei.com

## 10. Acknowledgements

The authors would like to thank Gregory Mirsky, Yimin Shen, Eric Osborne, Spencer Jackson and Eric Gray for their valuable comments and suggestions.

## 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, <<http://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, <<http://www.rfc-editor.org/info/rfc3031>>.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<http://www.rfc-editor.org/info/rfc4385>>.
- [RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <<http://www.rfc-editor.org/info/rfc4446>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<http://www.rfc-editor.org/info/rfc5586>>.
- [RFC5654] Niven-Jenkins, B., Ed., Brungard, D., Ed., Betts, M., Ed., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, DOI 10.17487/RFC5654, September 2009, <<http://www.rfc-editor.org/info/rfc5654>>.

## 11.2. Informative References

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <<http://www.rfc-editor.org/info/rfc5920>>.
- [RFC6371] Busi, I., Ed. and D. Allan, Ed., "Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks", RFC 6371, DOI 10.17487/RFC6371, September 2011, <<http://www.rfc-editor.org/info/rfc6371>>.

- [RFC6378] Weingarten, Y., Ed., Bryant, S., Osborne, E., Sprecher, N., and A. Fulignoli, Ed., "MPLS Transport Profile (MPLS-TP) Linear Protection", RFC 6378, DOI 10.17487/RFC6378, October 2011, <<http://www.rfc-editor.org/info/rfc6378>>.
- [RFC6941] Fang, L., Ed., Niven-Jenkins, B., Ed., Mansfield, S., Ed., and R. Graveman, Ed., "MPLS Transport Profile (MPLS-TP) Security Framework", RFC 6941, DOI 10.17487/RFC6941, April 2013, <<http://www.rfc-editor.org/info/rfc6941>>.
- [RFC6974] Weingarten, Y., Bryant, S., Ceccarelli, D., Caviglia, D., Fondelli, F., Corsi, M., Wu, B., and X. Dai, "Applicability of MPLS Transport Profile for Ring Topologies", RFC 6974, DOI 10.17487/RFC6974, July 2013, <<http://www.rfc-editor.org/info/rfc6974>>.

## Authors' Addresses

Weiqliang Cheng  
China Mobile

Email: [chengweiqliang@chinamobile.com](mailto:chengweiqliang@chinamobile.com)

Lei Wang  
China Mobile

Email: [wangleiyj@chinamobile.com](mailto:wangleiyj@chinamobile.com)

Han Li  
China Mobile

Email: [lihan@chinamobile.com](mailto:lihan@chinamobile.com)

Huub van Helvoort  
Hai Gaoming BV

Email: [huubatwork@gmail.com](mailto:huubatwork@gmail.com)

Jie Dong  
Huawei Technologies

Email: [jie.dong@huawei.com](mailto:jie.dong@huawei.com)

TEAS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 13 July 2022

V.P. Beeram  
T. Saad  
Juniper Networks  
R. Gandhi  
Cisco Systems, Inc.  
X. Liu  
Volta Networks  
I. Bryskin  
Individual  
9 January 2022

A YANG Data Model for Resource Reservation Protocol (RSVP)  
draft-ietf-teas-yang-rsvp-17

## Abstract

This document defines a YANG data model for the configuration and management of the RSVP protocol. The YANG data model covers the building blocks that may be augmented by other RSVP extension data models such as RSVP Traffic-Engineering (RSVP-TE). It is divided into two modules that cover the basic and extended RSVP features.

## 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 13 July 2022.

## Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.



Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Requirements Language . . . . .	3
2.1. Prefixes in Data Node Names . . . . .	3
2.2. Model Tree Diagram . . . . .	4
3. Model Overview . . . . .	4
3.1. Module(s) Relationship . . . . .	5
3.2. Core Features . . . . .	5
3.3. Optional Features . . . . .	6
3.4. Data Model Structure . . . . .	6
3.5. Model Notifications . . . . .	8
4. RSVP Base YANG Model . . . . .	9
4.1. Tree Diagram . . . . .	9
4.2. YANG Module . . . . .	13
5. RSVP Extended YANG Model . . . . .	34
5.1. Tree Diagram . . . . .	34
5.2. YANG Module . . . . .	36
6. IANA Considerations . . . . .	45
7. Security Considerations . . . . .	46
8. Acknowledgement . . . . .	47
9. Appendix A . . . . .	47
10. Contributors . . . . .	53
11. References . . . . .	54
11.1. Normative References . . . . .	54
11.2. Informative References . . . . .	56
Authors' Addresses . . . . .	57

## 1. Introduction

YANG [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]. YANG has proved relevant beyond its initial confines, as bindings to other interfaces (e.g. RESTCONF [RFC8040]) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model for the configuration and management of the RSVP protocol [RFC2205]. The data model is divided into two modules: a base and extended RSVP YANG modules. The RSVP

base YANG 'ietf-rsvp' module covers the data that is core to the function of the RSVP protocol and MUST be supported by vendors that support RSVP protocol [RFC2205]. The RSVP extended 'ietf-rsvp-extended' module covers the data that is optional, or provides ability to tune RSVP protocol base functionality. The support for RSVP extended module features by vendors is considered optional.

The RSVP YANG model provides the building blocks needed to allow augmentation by other models that extend the RSVP protocol- such as using RSVP extensions to signal Label Switched Paths (LSPs) as defined in [RFC3209].

The YANG module(s) defined in this document are compatible with the Network Management Datastore Architecture (NMDA) [RFC7950].

## 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.

The terminology for describing YANG data models is found in [RFC7950].

### 2.1. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
inet	ietf-inet-types	[RFC6991]
yang	ietf-yang-types	[RFC6991]
key-chain	ietf-key-chain	[RFC8177]

Table 1: Prefixes and corresponding YANG modules

## 2.2. Model Tree Diagram

A full tree diagram of the module(s) defined in this document is given in subsequent sections as per the syntax defined in [RFC8340].

## 3. Model Overview

The RSVP YANG module augments the "control-plane-protocol" entry from the 'ietf-routing' module defined in [RFC8349]. It also defines the identity "rsvp" of base type "rt:routing-protocol" to identify the RSVP routing protocol.

The 'ietf-rsvp' model defines a single instance of the RSVP protocol. The top 'rsvp' container encompasses data for one such RSVP protocol instance. Multiple instances can be defined as multiple control-plane protocols instances as described in [RFC8349].

The YANG data model defined has the common building blocks for the operation of the base RSVP protocol for the session type defined in [RFC2205]. The augmentation of this model by other models (e.g. to support RSVP Traffic Engineering (TE) extensions for signaling Label Switched Paths (LSPs)) are outside the scope of this document and are discussed in separate document(s).

### 3.1. Module(s) Relationship

This RSVP YANG data model defined in this document is divided into two modules: a base and extended modules. The RSVP data covered in 'ietf-rsvp' module are categorized as core to the function of the protocol and MUST be supported by vendors claiming the support for RSVP protocol [RFC2205].

The RSVP extended features that are covered in 'ietf-rsvp-extended' module are categorized as either optional or providing ability to better tune the basic functionality of the RSVP protocol. The support for RSVP extended features by all vendors is considered optional.

The relationship between the base and RSVP extended YANG modules and the IETF routing YANG model is shown in Figure 1.

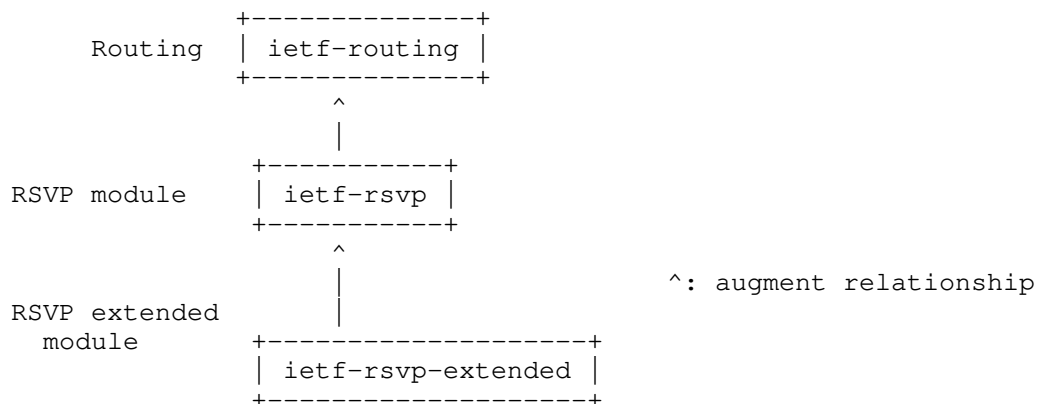


Figure 1: Relationship of RSVP and RSVP extended modules with other protocol modules

### 3.2. Core Features

The RSVP data covered in the 'ietf-rsvp' YANG module provides the common building blocks that are required to configure, operate and manage the RSVP protocol and MUST be supported by vendors that claim the support for base RSVP protocol defined in [RFC2205].

In addition, the following standard RSVP core features are modeled under the 'ietf-rsvp' module:

- \* Basic operational statistics, including protocol messages, packets and errors.

- \* Basic RSVP authentication feature as defined in [RFC2747]) using string based authentication key.
- \* Basic RSVP Refresh Reduction feature as defined in ([RFC2961]).
- \* Basic RSVP Hellos feature as defined in ([RFC3209])
- \* Basic RSVP Graceful Restart feature as defined in [RFC3473], [RFC5063], and [RFC5495].

### 3.3. Optional Features

Optional features are beyond the basic configuration, and operation of the RSVP protocol. The decision whether to support these RSVP features on a particular device is left to the vendor that supports the RSVP core features.

The following optional features that are covered in the 'ietf-rsvp-extended' YANG module:

- \* Advanced operational statistics, including protocol messages, packets and errors.
- \* Advanced RSVP authentication features as defined in [RFC2747]) using various authentication key types including those defined in [RFC8177].
- \* Advanced RSVP Refresh Reduction features defined in ([RFC2961]).
- \* Advanced RSVP Hellos features as defined in [RFC3209], and [rfc4558].
- \* Advanced RSVP Graceful Restart features as defined in [RFC3473], [RFC5063], and [RFC5495].

### 3.4. Data Model Structure

The RSVP YANG data model defines the 'rsvp' top-level container that contains the configuration and operational state for the RSVP protocol. The presence of this container enables the RSVP protocol functionality.

The 'rsvp' top-level container also includes data that has router level scope (i.e. applicable to all objects modeled under rsvp). It also contains configuration and state data about the following types of RSVP objects:

- \* interfaces

\* neighbors

\* sessions

The derived state data is contained in "read-only" nodes directly under the intended object as shown in Figure 2.

```

module: ietf-rsvp
  +--rw rsvp!
    +--rw <<router-level scope data>>
      .
      .
    +--rw interfaces
      .
      +-- ro <<derived state associated with interfaces>>
      .
      .
    +--rw neighbors
      .
      +-- ro <<derived state associated with the LSP Tunnel>>
      .
      .
    +--rw sessions
      .
      +-- ro <<derived state associated with the LSP Tunnel>>
      .
  rpcs:
    +--x clear-session
    +--x clear-neighbor
    +--x clear-authentication

```

Figure 2: RSVP high-level tree model view

The following

'router-level':

The router-level scope configuration and state data are applicable to all modeled objects under the top-level 'rsvp' container, and MAY affect the RSVP protocol behavior.

'interfaces':

The 'interfaces' container includes a list of RSVP enabled interfaces. It also includes RSVP configuration and state data that is applicable to all interfaces. An entry in the interfaces list MAY carry its own configuration or state data. Any data or state under the "interfaces" container level is equally applicable to all interfaces unless it is explicitly overridden by configuration or state under a specific interface.

'neighbors' :

The 'neighbors' container includes a list of RSVP neighbors. An entry in the RSVP neighbor list MAY carry its own configuration and state relevant to the specific RSVP neighbor. The RSVP neighbors can be dynamically discovered using RSVP signaling, or can be explicitly configured.

'sessions' :

The 'sessions' container includes a list RSVP sessions. An entry in the RSVP session list MAY carry its own configuration and state relevant to a specific RSVP session. RSVP sessions are usually derived state that are created as result of signaling. This model defines attributes related to IP RSVP sessions as defined in [RFC2205].

The defined YANG data model supports configuration inheritance for neighbors, and interfaces. Data nodes defined under the main container (e.g. the container that encompasses the list of interfaces, or neighbors) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element (e.g. interface).

### 3.5. Model Notifications

Modeling notifications data is key in any defined YANG data model. [RFC8639] and [RFC8641] define a subscription and push mechanism for YANG datastores. This mechanism currently allows the user to:

- \* Subscribe notifications on a per client basis
- \* Specify subtree filters [RFC6241] or XPath filters [RFC8639] so that only interested contents will be sent.
- \* Specify either periodic or on-demand notifications.

#### 4. RSVP Base YANG Model

The RSVP base module includes the core features and building blocks for modeling the RSVP protocol as described in Section 3.2.

##### 4.1. Tree Diagram

Figure 3 shows the YANG tree representation for configuration, state data and RPCs that are covered in 'ietf-rsvp' YANG module:

module: ietf-rsvp

```

augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +--rw rsvp!
      +--rw interfaces
        +--rw refresh-reduction
          | +--rw enabled?    boolean
        +--rw hellos
          | +--rw enabled?    boolean
        +--rw authentication
          | +--rw enabled?      boolean
          | +--rw authentication-key? string
          | +--rw crypto-algorithm identityref
        +--ro statistics
          +--ro messages
            +--ro ack-sent?          yang:counter64
            +--ro ack-received?      yang:counter64
            +--ro bundle-sent?       yang:counter64
            +--ro bundle-received?   yang:counter64
            +--ro hello-sent?        yang:counter64
            +--ro hello-received?    yang:counter64
            +--ro integrity-challenge-sent? yang:counter64
            +--ro integrity-challenge-received? yang:counter64
            +--ro integrity-response-sent? yang:counter64
            +--ro integrity-response-received? yang:counter64
            +--ro notify-sent?        yang:counter64
            +--ro notify-received?    yang:counter64
            +--ro path-sent?          yang:counter64
            +--ro path-received?      yang:counter64
            +--ro path-err-sent?      yang:counter64
            +--ro path-err-received?  yang:counter64
            +--ro path-tear-sent?     yang:counter64
            +--ro path-tear-received? yang:counter64
            +--ro resv-sent?          yang:counter64
            +--ro resv-received?      yang:counter64
            +--ro resv-confirm-sent?  yang:counter64
            +--ro resv-confirm-received? yang:counter64

```



```

+---ro resv-err-sent?                yang:counter64
+---ro resv-err-received?            yang:counter64
+---ro resv-tear-sent?               yang:counter64
+---ro resv-tear-received?           yang:counter64
+---ro srefresh-sent?                yang:counter64
+---ro srefresh-received?            yang:counter64
+---ro unknown-messages-received?    yang:counter64
+---ro packets
|   +---ro sent?                     yang:counter64
|   +---ro received?                 yang:counter64
+---ro errors
|   +---ro authenticate?             yang:counter64
|   +---ro checksum?                 yang:counter64
|   +---ro packet-length?            yang:counter64
+---rw interface* [name]
|   +---rw name                      if:interface-ref
|   +---rw refresh-reduction
|   |   +---rw enabled?              boolean
+---rw hellos
|   +---rw enabled?                  boolean
+---rw authentication
|   +---rw enabled?                  boolean
|   +---rw authentication-key?       string
|   +---rw crypto-algorithm           identityref
+---ro statistics
|   +---ro messages
|   |   +---ro ack-sent?
|   |   |   yang:counter64
|   |   +---ro ack-received?
|   |   |   yang:counter64
|   |   +---ro bundle-sent?
|   |   |   yang:counter64
|   |   +---ro bundle-received?
|   |   |   yang:counter64
|   |   +---ro hello-sent?
|   |   |   yang:counter64
|   |   +---ro hello-received?
|   |   |   yang:counter64
|   |   +---ro integrity-challenge-sent?
|   |   |   yang:counter64
|   |   +---ro integrity-challenge-received?
|   |   |   yang:counter64
|   |   +---ro integrity-response-sent?
|   |   |   yang:counter64
|   |   +---ro integrity-response-received?
|   |   |   yang:counter64
|   |   +---ro notify-sent?
|   |   |   yang:counter64

```

```

+--ro notify-received?
|   yang:counter64
+--ro path-sent?
|   yang:counter64
+--ro path-received?
|   yang:counter64
+--ro path-err-sent?
|   yang:counter64
+--ro path-err-received?
|   yang:counter64
+--ro path-tear-sent?
|   yang:counter64
+--ro path-tear-received?
|   yang:counter64
+--ro resv-sent?
|   yang:counter64
+--ro resv-received?
|   yang:counter64
+--ro resv-confirm-sent?
|   yang:counter64
+--ro resv-confirm-received?
|   yang:counter64
+--ro resv-err-sent?
|   yang:counter64
+--ro resv-err-received?
|   yang:counter64
+--ro resv-tear-sent?
|   yang:counter64
+--ro resv-tear-received?
|   yang:counter64
+--ro srefresh-sent?
|   yang:counter64
+--ro srefresh-received?
|   yang:counter64
+--ro unknown-messages-received?
|   yang:counter64
+--ro packets
|   +--ro sent?          yang:counter64
|   +--ro received?     yang:counter64
+--ro errors
|   +--ro authenticate?  yang:counter64
|   +--ro checksum?      yang:counter64
|   +--ro packet-length? yang:counter64
+--rw sessions
|   +--ro session-ip*
|       [destination protocol-id destination-port]
|       +--ro destination-port  uint16
|       +--ro protocol-id       uint8

```

```

    +--ro source?                inet:ip-address
    +--ro destination            inet:ip-address
    +--ro session-name?          string
    +--ro session-status?        enumeration
    +--ro session-type            identityref
    +--ro psbs
      +--ro psb* []
        +--ro source-port?       inet:port-number
        +--ro expires-in?        uint32
    +--ro rsbs
      +--ro rsb* []
        +--ro source-port?       inet:port-number
        +--ro reservation-style  identityref
        +--ro expires-in?        uint32
+--rw neighbors
  +--rw neighbor* [address]
    +--rw address                inet:ip-address
    +--rw epoch?                 uint32
    +--rw expiry-time?           uint32
    +--rw graceful-restart
      +--ro neighbor-restart-time?  uint32
      +--ro neighbor-recovery-time? uint32
      +--rw helper-mode
        +--ro neighbor-restart-time-remaining?  uint32
        +--ro neighbor-recovery-time-remaining? uint32
    +--ro hello-status?          enumeration
    +--rw interface?             if:interface-ref
    +--ro neighbor-status?       enumeration
    +--rw refresh-reduction-capable? boolean
    +--ro restart-count?         yang:counter32
    +--ro restart-time?          yang:date-and-time
+--rw graceful-restart
  +--rw enabled?                 boolean
  +--rw local-restart-time?      uint32
  +--rw local-recovery-time?     uint32
  +--rw helper-mode
    +--rw enabled?               boolean
    +--rw max-helper-restart-time?  uint32
    +--rw max-helper-recovery-time? uint32

rpcs:
  +---x clear-session
    +---w input
      +---w routing-protocol-instance-name  leafref
      +---w (filter-type)
        +---:(match-all)
          +---w all                          empty
        +---:(match-one)

```

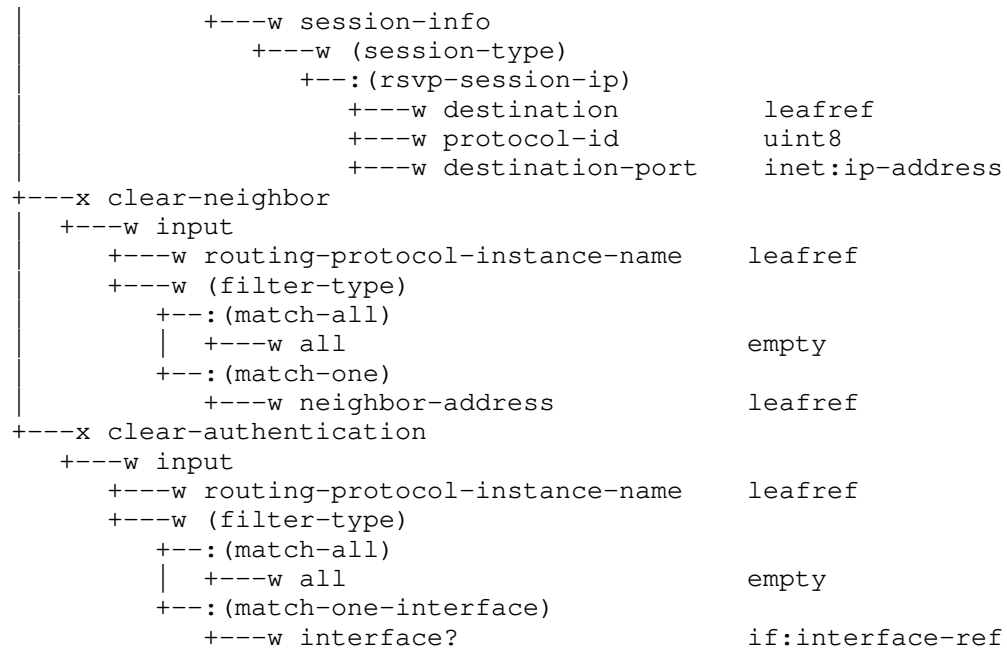


Figure 3: RSVP model tree diagram

#### 4.2. YANG Module

The ietf-rsvp module imports from the following modules:

- \* ietf-interfaces defined in [RFC8343]
- \* ietf-yang-types and ietf-inet-types defined in [RFC6991]
- \* ietf-routing defined in [RFC8349]
- \* ietf-key-chain defined in [RFC8177]
- \* ietf-netconf-acm defined in [RFC8341]

This module also references the following documents: [RFC2205], [RFC5495], [RFC3473], [RFC5063], [RFC2747], [RFC3209], and [RFC2961].

```

<CODE BEGINS> file "ietf-rsvp@2021-12-02.yang"
module ietf-rsvp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp";

  /* Replace with IANA when assigned */

```

```
prefix rsvp;

import ietf-interfaces {
  prefix if;
  reference
    "RFC8343: A YANG Data Model for Interface Management";
}
import ietf-inet-types {
  prefix inet;
  reference
    "RFC6991: Common YANG Data Types";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC6991: Common YANG Data Types";
}
import ietf-routing {
  prefix rt;
  reference
    "RFC8349: A YANG Data Model for Routing Management
    (NMDA Version)";
}
import ietf-key-chain {
  prefix key-chain;
  reference
    "RFC8177: YANG Data Model for Key Chains";
}
import ietf-netconf-acm {
  prefix nacm;
  reference
    "RFC8341: Network Configuration Access Control Model";
}
organization
  "IETF Traffic Engineering Architecture and Signaling (TEAS)
  Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/teas/>
  WG List:    <mailto:teas@ietf.org>

  Editor:     Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
               <mailto:tsaad@juniper.net>

  Editor:     Rakesh Gandhi
               <mailto:rgandhi@cisco.com>
```

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

Editor:  Igor Bryskin
        <mailto:i_bryskin@yahoo.com>";
description
  "This module contains the RSVP YANG data model.
  The model fully conforms to the Network Management Datastore
  Architecture (NMDA).

  Copyright (c) 2019 IETF Trust and the persons
  identified as authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).
  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision 2021-12-02 {
  description
    "Initial version.";
  reference
    "RFCXXXX: A YANG Data Model for Resource Reservation Protocol
    (RSVP)";
}

identity rsvp {
  base rt:routing-protocol;
  description
    "RSVP protocol";
}

identity rsvp-session-type {
  description
    "Base RSVP session type";
}

identity rsvp-session-ip {
  base rsvp-session-type;
```

```
    description
      "RSVP IP session type";
  }

  identity reservation-style {
    description
      "Base identity for reservation style.";
  }

  identity reservation-wildcard-filter {
    base reservation-style;
    description
      "Wildcard-Filter (WF) Style.";
    reference
      "RFC2205";
  }

  identity reservation-fixed-filter {
    base reservation-style;
    description
      "Fixed-Filter (FF) Style.";
    reference
      "RFC2205";
  }

  identity reservation-shared-explicit {
    base reservation-style;
    description
      "Shared Explicit (SE) Style.";
    reference
      "RFC2205";
  }

  grouping graceful-restart {
    description
      "RSVP graceful restart local parameters grouping.";
    container graceful-restart {
      description
        "Graceful restart local information.";
      leaf enabled {
        type boolean;
        default "false";
        description
          "'true' if RSVP Graceful Restart is enabled.
          'false' if RSVP Graceful Restart is disabled.";
        reference "RFC5495";
      }
      leaf local-restart-time {
```

```
    type uint32;
    units "seconds";
    default "120";
    description
        "Time it takes the local node to restart its RSVP-TE
        component (to the point where it can exchange RSVP
        Hello with its neighbors). A value of 0xffffffff
        indicates that the restart of the neighbor's control plane
        may occur over an indeterminate interval and that the
        operation of its data plane is unaffected by control plane
        failures.";
    reference "RFC3473";
}
leaf local-recovery-time {
    type uint32;
    units "seconds";
    default "120";
    description
        "The period of time, in seconds, that the local
        node requires to re-synchronize RSVP and MPLS
        forwarding state with its neighbor. A value of zero (0)
        indicates that MPLS forwarding state was not preserved
        across a particular reboot.";
    reference "RFC3473";
}
container helper-mode {
    description
        "Helper mode information. In this mode, the node
        resynchronize its stored states with a neighbor whose
        control plane has restarted. The helper mode term is
        borrowed from RFC3623 and adopted by several vendors
        vendors in their implementation of RSVP graceful restart.";
    leaf enabled {
        type boolean;
        default "true";
        description
            "'true' if helper mode is enabled.";
    }
    leaf max-helper-restart-time {
        type uint32;
        units "seconds";
        default "20";
        description
            "The maximum time the router or switch waits after it
            discovers that the neighboring router has gone down
            before it declares the neighbor down.";
        reference "RFC5063";
    }
}
```



```
    leaf max-helper-recovery-time {
      type uint32;
      units "seconds";
      default "180";
      description
        "The maximum amount of time the router retains the state
         of its RSVP neighbors while they undergo a graceful
         restart.";
      reference "RFC5063";
    }
  }
}
grouping neighbor-graceful-restart {
  description
    "RSVP graceful restart neighbor parameters grouping.";
  container graceful-restart {
    description
      "Graceful restart information.";
    leaf neighbor-restart-time {
      type uint32;
      units "seconds";
      default "120";
      config false;
      description
        "Time it takes the neighbor node to restart its RSVP-TE
         component (to the point where it can exchange RSVP
         Hello with its neighbors). A value of 0xffffffff
         indicates that the restart of the neighbor's control plane
         may occur over an indeterminate interval and that the
         operation of its data plane is unaffected by control plane
         failures.";
      reference "RFC3473";
    }
    leaf neighbor-recovery-time {
      type uint32;
      units "seconds";
      default "120";
      config false;
      description
        "The period of time, in milliseconds, that the neighbor
         node requires to re-synchronize RSVP and MPLS
         forwarding state with its neighbor. A value of zero (0)
         indicates that MPLS forwarding state was not preserved
         across a particular reboot.";
      reference "RFC3473";
    }
  }
  container helper-mode {
```

```
    description
      "Helper mode information.";
    leaf neighbor-restart-time-remaining {
      type uint32;
      units "seconds";
      config false;
      description
        "Number of seconds remaining for neighbor to send Hello
         message after restart.";
      reference "RFC5063";
    }
    leaf neighbor-recovery-time-remaining {
      type uint32;
      units "seconds";
      config false;
      description
        "Number of seconds remaining for neighbor to refresh.";
      reference "RFC5063";
    }
  }
  // helper-mode
}

grouping refresh-reduction {
  description
    "Top level grouping for RSVP refresh reduction parameters.";
  container refresh-reduction {
    description
      "Top level container for RSVP refresh reduction parameters.";
    leaf enabled {
      type boolean;
      default "true";
      description
        "'true' if RSVP Refresh Reduction is enabled.
         'false' if RSVP Refresh Reduction is disabled.";
    }
    reference
      "RFC2961 RSVP Refresh Overhead Reduction Extensions";
  }
}

grouping authentication {
  description
    "Top level grouping for RSVP authentication parameters.";
  container authentication {
    description
      "Top level container for RSVP authentication parameters.";
  }
}
```

```
    leaf enabled {
      type boolean;
      default "false";
      description
        "'true' if RSVP Authentication is enabled.
        'false' if RSVP Authentication is disabled.";
    }
    leaf authentication-key {
      type string;
      default "";
      description
        "An authentication key string.";
      reference
        "RFC2747: RSVP Cryptographic Authentication";
    }
    leaf crypto-algorithm {
      type identityref {
        base key-chain:crypto-algorithm;
      }
      mandatory true;
      description
        "Cryptographic algorithm associated with key.";
    }
  }
}

grouping hellos {
  description
    "Top level grouping for RSVP hellos parameters.";
  container hellos {
    description
      "Top level container for RSVP hello parameters.";
    leaf enabled {
      type boolean;
      default "true";
      description
        "'true' if RSVP Hello is enabled.
        'false' if RSVP Hello is disabled.";
    }
    reference
      "RFC3209: RSVP-TE: Extensions to RSVP for LSP Tunnels.
      RFC5495: Description of the Resource Reservation Protocol -
      Traffic-Engineered (RSVP-TE) Graceful Restart Procedures.";
  }
}

grouping session-attributes {
  description
```

```
    "Top level grouping for RSVP session properties.";
  leaf destination-port {
    type uint16;
    description
      "RSVP destination port.";
    reference
      "RFC2205";
  }
  leaf protocol-id {
    type uint8;
    description
      "The IP protocol ID.";
    reference
      "RFC2205, section 3.2";
  }
  leaf source {
    type inet:ip-address;
    description
      "RSVP source address.";
    reference
      "RFC2205";
  }
  leaf destination {
    type inet:ip-address;
    description
      "RSVP destination address.";
    reference
      "RFC2205";
  }
  leaf session-name {
    type string;
    default "";
    description
      "The signaled name of this RSVP session.";
  }
  leaf session-status {
    type enumeration {
      enum up {
        description
          "RSVP session is up.";
      }
      enum down {
        description
          "RSVP session is down.";
      }
    }
    default "down";
    description
```

```
        "Enumeration of RSVP session states.";
    }
    leaf session-type {
        type identityref {
            base rsvp-session-type;
        }
        mandatory "true";
        description
            "RSVP session type.";
    }
    container psbs {
        description
            "Path State Block (PSB) container.";
        list psb {
            description
                "List of Path State Blocks.";
            leaf source-port {
                type inet:port-number;
                description
                    "RSVP source port.";
                reference
                    "RFC2205";
            }
            leaf expires-in {
                type uint32;
                units "seconds";
                default "180";
                description
                    "Time to expiry (in seconds).";
            }
        }
    }
    container rsbs {
        description
            "Reservation State Block (RSB) container.";
        list rsb {
            description
                "List of Reservation State Blocks.";
            leaf source-port {
                type inet:port-number;
                description
                    "RSVP source port.";
                reference
                    "RFC2205";
            }
            leaf reservation-style {
                type identityref {
                    base reservation-style;
                }
            }
        }
    }
}
```

```
    }
    mandatory "true";
    description
      "RSVP reservation style.";
  }
  leaf expires-in {
    type uint32;
    units "seconds";
    default "180";
    description
      "Time to expiry (in seconds).";
  }
}

grouping neighbor-attributes {
  description
    "Top level grouping for RSVP neighbor properties.";
  leaf address {
    type inet:ip-address;
    description
      "Address of the RSVP neighbor.";
  }
  leaf epoch {
    type uint32;
    default "0";
    description
      "Neighbor epoch.";
    reference "RFC5063";
  }
  leaf expiry-time {
    type uint32;
    units "seconds";
    default "180";
    description
      "Neighbor expiry time after which the neighbor state is
      purged if no states associated with it.";
  }
  uses neighbor-graceful-restart {
    description
      "Allows configuration applicable to all
      neighbors";
  }
  leaf hello-status {
    type enumeration {
      enum enabled {
        description
```

```
        "RSVP Hellos enabled.";
    }
    enum disabled {
        description
            "RSVP Hellos disabled.";
    }
    enum restarting {
        description
            "RSVP restarting.";
    }
}
config false;
description
    "RSVP Hello status.";
}
leaf interface {
    type if:interface-ref;
    description
        "Interface where RSVP neighbor was detected.";
}
leaf neighbor-status {
    type enumeration {
        enum up {
            description
                "Neighbor state up.";
        }
        enum down {
            description
                "Neighbor state down.";
        }
        enum hello-disable {
            description
                "RSVP Hellos disabled.";
        }
        enum restarting {
            description
                "RSVP neighbor restarting.";
        }
    }
}
config false;
description
    "RSVP neighbor state.";
}
leaf refresh-reduction-capable {
    type boolean;
    default "true";
    description
        "Enables all RSVP refresh reduction message bundling, RSVP
```

```
        message ID, reliable message delivery and Srefresh
        messages.";
    reference
        "RFC2961 RSVP Refresh Overhead Reduction Extensions";
}
leaf restart-count {
    type yang:counter32;
    config false;
    description
        "Number of times this RSVP neighbor has restarted.";
}
leaf restart-time {
    type yang:date-and-time;
    config false;
    description
        "Last restart time of the RSVP neighbor.";
    reference "RFC3473";
}
}

grouping packet-statistics {
    description
        "Packet statistics grouping.";
    container packets {
        description
            "Packet statistics container.";
        leaf sent {
            type yang:counter64;
            description
                "RSVP packet sent count.";
        }
        leaf received {
            type yang:counter64;
            description
                "RSVP packet received count.";
        }
    }
}

grouping message-statistics {
    description
        "RSVP protocol statistics grouping.";
    container messages {
        description
            "RSVP protocol statistics container.";
        leaf ack-sent {
            type yang:counter64;
            description
```



```
        "RSVP Hello sent count.";
    }
    leaf ack-received {
        type yang:counter64;
        description
            "RSVP Hello received count.";
    }
    leaf bundle-sent {
        type yang:counter64;
        description
            "RSVP Bundle message sent count.";
    }
    leaf bundle-received {
        type yang:counter64;
        description
            "RSVP Bundle message received count.";
    }
    leaf hello-sent {
        type yang:counter64;
        description
            "RSVP Hello message sent count.";
    }
    leaf hello-received {
        type yang:counter64;
        description
            "RSVP Hello message received count.";
    }
    leaf integrity-challenge-sent {
        type yang:counter64;
        description
            "RSVP Integrity Challenge message sent count.";
    }
    leaf integrity-challenge-received {
        type yang:counter64;
        description
            "RSVP Integrity Challenge message received count.";
    }
    leaf integrity-response-sent {
        type yang:counter64;
        description
            "RSVP Integrity Response message sent count.";
    }
    leaf integrity-response-received {
        type yang:counter64;
        description
            "RSVP Integrity Response message received count.";
    }
    leaf notify-sent {
```

```
    type yang:counter64;
    description
      "RSVP Notify message sent count.";
  }
  leaf notify-received {
    type yang:counter64;
    description
      "RSVP Notify message received count.";
  }
  leaf path-sent {
    type yang:counter64;
    description
      "RSVP Path message sent count.";
  }
  leaf path-received {
    type yang:counter64;
    description
      "RSVP Path message received count.";
  }
  leaf path-err-sent {
    type yang:counter64;
    description
      "RSVP Path error message sent count.";
  }
  leaf path-err-received {
    type yang:counter64;
    description
      "RSVP Path error message received count.";
  }
  leaf path-tear-sent {
    type yang:counter64;
    description
      "RSVP Path tear message sent count.";
  }
  leaf path-tear-received {
    type yang:counter64;
    description
      "RSVP Path tear message received count.";
  }
  leaf resv-sent {
    type yang:counter64;
    description
      "RSVP Resv message sent count.";
  }
  leaf resv-received {
    type yang:counter64;
    description
      "RSVP Resv message received count.";
```

```
    }
    leaf resv-confirm-sent {
      type yang:counter64;
      description
        "RSVP Confirm message sent count.";
    }
    leaf resv-confirm-received {
      type yang:counter64;
      description
        "RSVP Confirm message received count.";
    }
    leaf resv-err-sent {
      type yang:counter64;
      description
        "RSVP Resv error message sent count.";
    }
    leaf resv-err-received {
      type yang:counter64;
      description
        "RSVP Resv error message received count.";
    }
    leaf resv-tear-sent {
      type yang:counter64;
      description
        "RSVP Resv tear message sent count.";
    }
    leaf resv-tear-received {
      type yang:counter64;
      description
        "RSVP Resv tear message received count.";
    }
    leaf srefresh-sent {
      type yang:counter64;
      description
        "RSVP Srefresh message sent count.";
    }
    leaf srefresh-received {
      type yang:counter64;
      description
        "RSVP Srefresh message received count.";
    }
    leaf unknown-messages-received {
      type yang:counter64;
      description
        "Unknown messages received count.";
    }
  }
}
```

```
grouping errors-statistics {
  description
    "Error statistics grouping.";
  container errors {
    description
      "Error statistics container.";
    leaf authenticate {
      type yang:counter64;
      description
        "The total number of RSVP packets received with an
        authentication failure.";
    }
    leaf checksum {
      type yang:counter64;
      description
        "The total number of RSVP packets received with an invalid
        checksum value.";
    }
    leaf packet-length {
      type yang:counter64;
      description
        "The total number of packets received with an invalid
        packet length.";
    }
  }
}

grouping statistics {
  description
    "RSVP statistic attributes.";
  container statistics {
    config false;
    description
      "RSVP statistics container.";
    uses message-statistics;
    uses packet-statistics;
    uses errors-statistics;
  }
}

grouping intf-attributes {
  description
    "Top level grouping for RSVP interface properties.";
  uses refresh-reduction;
  uses hellos;
  uses authentication;
  uses statistics;
}
```

```
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol" {
    when "rt:type = 'rsvp:rsvp'" {
      description
        "This augment is only valid when routing protocol instance
        type is RSVP.";
    }
    description
      "RSVP protocol augmentation.";
    container rsvp {
      presence "Enable RSVP feature";
      description
        "RSVP feature container";
      container interfaces {
        description
          "RSVP interfaces container.";
        uses intf-attributes;
        list interface {
          key "name";
          description
            "RSVP interfaces.";
          leaf name {
            type if:interface-ref;
            description
              "RSVP interface.";
          }
          uses intf-attributes;
        }
      }
    }
    container sessions {
      description
        "RSVP sessions container.";
      list session-ip {
        key "destination protocol-id destination-port";
        config false;
        description
          "List of RSVP sessions.";
        uses session-attributes;
      }
    }
    container neighbors {
      description
        "RSVP neighbors container";
      list neighbor {
        key "address";
        description
          "List of RSVP neighbors";
        uses neighbor-attributes;
      }
    }
  }
```

```
    }
  }
  uses graceful-restart;
}

grouping session-ref {
  description
    "Session reference information";
  leaf destination {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/rsvp:rsvp"
        + "/rsvp:sessions/rsvp:session-ip/destination";
    }
    mandatory true;
    description
      "The RSVP session destination.";
  }
  leaf protocol-id {
    type uint8;
    mandatory true;
    description
      "The RSVP session protocol ID.";
  }
  leaf destination-port {
    type inet:ip-address;
    mandatory true;
    description
      "The RSVP session destination port.";
  }
}

rpc clear-session {
  nacm:default-deny-all;
  description
    "Clears RSVP sessions RPC";
  input {
    leaf routing-protocol-instance-name {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/rt:name";
      }
      mandatory true;
      description
        "Name of the RSVP protocol instance whose session
        is being cleared."
    }
  }
}
```

```
        If the corresponding RSVP instance doesn't exist,
        then the operation will fail with an error-tag of
        'data-missing' and an error-app-tag of
        'routing-protocol-instance-not-found'.";
    }
    choice filter-type {
        mandatory true;
        description
            "Filter choice";
        case match-all {
            leaf all {
                type empty;
                mandatory true;
                description
                    "Match all RSVP sessions.";
            }
        }
        case match-one {
            container session-info {
                description
                    "Specifies the specific session to invoke the operation
                    on.";
                choice session-type {
                    mandatory true;
                    description
                        "The RSVP session type.";
                    case rsvp-session-ip {
                        uses session-ref;
                    }
                }
            }
        }
    }
}

rpc clear-neighbor {
    nacm:default-deny-all;
    description
        "RPC to clear the RSVP Hello session to a neighbor.";
    input {
        leaf routing-protocol-instance-name {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                    + "rt:control-plane-protocol/rt:name";
            }
            mandatory true;
            description

```

```
        "Name of the RSVP protocol instance whose session
        is being cleared.

        If the corresponding RSVP instance doesn't exist,
        then the operation will fail with an error-tag of
        'data-missing' and an error-app-tag of
        'routing-protocol-instance-not-found'.";
    }
    choice filter-type {
        mandatory true;
        description
            "The Filter choice.";
        case match-all {
            leaf all {
                type empty;
                mandatory true;
                description
                    "Match all RSVP neighbor sessions.";
            }
        }
        case match-one {
            leaf neighbor-address {
                type leafref {
                    path "/rt:routing/rt:control-plane-protocols"
                        + "/rt:control-plane-protocol/rsvp:rsvp"
                        + "/rsvp:neighbors/rsvp:neighbor/address";
                }
                mandatory true;
                description
                    "Match the specific RSVP neighbor session.";
            }
        }
    }
}

rpc clear-authentication {
    nacm:default-deny-all;
    description
        "Clears the RSVP Security Association (SA) before the
        lifetime expires.";
    input {
        leaf routing-protocol-instance-name {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                    + "rt:control-plane-protocol/rt:name";
            }
            mandatory true;
        }
    }
}
```



```

description
    "Name of the RSVP protocol instance whose session
    is being cleared.

    If the corresponding RSVP instance doesn't exist,
    then the operation will fail with an error-tag of
    'data-missing' and an error-app-tag of
    'routing-protocol-instance-not-found'.";
}
choice filter-type {
    mandatory true;
    description
        "Filter choice";
    case match-all {
        leaf all {
            type empty;
            mandatory true;
            description
                "Match all RSVP security associations.";
        }
    }
    case match-one-interface {
        leaf interface {
            type if:interface-ref;
            description
                "Interface where RSVP security association(s) to be
                detected.";
        }
    }
}
}
}
}
<CODE ENDS>
```

The RSVP extended module augments the RSVP base module with optional feature data as described in Section 3.3.

Figure 4 shows the YANG tree representation for configuration and state data that are covered in 'ietf-rsvp-extended' YANG module:

```

module: ietf-rsvp-extended

augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp
    /rsvp:graceful-restart:
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces:
    +--rw refresh-interval?          uint32
    +--rw refresh-misses?            uint32
    +--rw checksum-enable?           empty
    +--rw patherr-state-removal?     empty
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:statistics/rsvp:packets:
    +--ro discontinuity-time?        yang:date-and-time
    +--ro out-dropped?               yang:counter64
    +--ro in-dropped?               yang:counter64
    +--ro out-errors?               yang:counter64
    +--ro in-errors?               yang:counter64
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:refresh-reduction:
    +--rw bundle-message-max-size?   uint32
    +--rw ack-hold-time?             uint32
    +--rw ack-max-size?             uint32
    +--rw ack-retransmit-time?       uint32
    +--rw srefresh-ack-desired?      empty
    +--rw srefresh-max-size?         uint32
    +--rw srefresh-relative-period?  uint8
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:hellos:
    +--rw interface-based?          empty
    +--rw hello-interval?           uint32
    +--rw hello-misses?            uint32
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:authentication:
    +--rw lifetime?                 uint32
    +--rw window-size?             uint32
    +--rw challenge?               empty
    +--rw retransmits?             uint32
    +--rw key-chain?               key-chain:key-chain-ref
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:interface:
    +--rw refresh-interval?          uint32
    +--rw refresh-misses?           uint32

```

```

    +--rw checksum-enable?          empty
    +--rw patherr-state-removal?    empty
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:interface/rsvp:statistics/rsvp:packets:
    +--ro discontinuity-time?      yang:date-and-time
    +--ro out-dropped?             yang:counter64
    +--ro in-dropped?             yang:counter64
    +--ro out-errors?             yang:counter64
    +--ro in-errors?             yang:counter64
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:interface/rsvp:refresh-reduction:
    +--rw bundle-message-max-size? uint32
    +--rw ack-hold-time?           uint32
    +--rw ack-max-size?           uint32
    +--rw ack-retransmit-time?    uint32
    +--rw srefresh-ack-desired?   empty
    +--rw srefresh-max-size?      uint32
    +--rw srefresh-relative-period? uint8
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:interface/rsvp:hellos:
    +--rw interface-based?        empty
    +--rw hello-interval?         uint32
    +--rw hello-misses?          uint32
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces
    /rsvp:interface/rsvp:authentication:
    +--rw lifetime?              uint32
    +--rw window-size?          uint32
    +--rw challenge?            empty
    +--rw retransmits?          uint32
    +--rw key-chain?            key-chain:key-chain-ref

```

Figure 4: RSVP extended module tree diagram

## 5.2. YANG Module

The 'ietf-rsvp-extended' module imports from the following modules:

- \* ietf-rsvp defined in this document
- \* ietf-routing defined in [RFC8349]
- \* ietf-yang-types and ietf-inet-types defined in [RFC6991]
- \* ietf-key-chain defined in [RFC8177]

Figure 5 shows the RSVP extended YANG module:

This module also references the following documents: [RFC3473], [RFC2747], [RFC3209], [RFC2205], [RFC2961], and [RFC5495].

```
<CODE BEGINS> file "ietf-rsvp-extended@2021-12-02.yang"
module ietf-rsvp-extended {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-extended";
  prefix rsvp-extended;

  import ietf-rsvp {
    prefix rsvp;
    reference
      "RFCXXXX: A YANG Data Model for Resource Reservation Protocol
      (RSVP)";
  }
  import ietf-routing {
    prefix rt;
    reference
      "RFC8349: A YANG Data Model for Routing Management
      (NMDA Version)";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC6991: Common YANG Data Types";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC8177: YANG Data Model for Key Chains";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
    Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/teas/>
    WG List:  <mailto:teas@ietf.org>

    Editor:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

    Editor:   Tarek Saad
              <mailto:tsaad@juniper.net>

    Editor:   Rakesh Gandhi
```

```
<mailto:rgandhi@cisco.com>

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

Editor:   Igor Bryskin
         <mailto:i_bryskin@yahoo.com>";

description
  "This module contains the Extended RSVP YANG data model.
  The model fully conforms to the Network Management Datastore
  Architecture (NMDA).

  Copyright (c) 2019 IETF Trust and the persons
  identified as authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).
  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision 2021-12-02 {
  description
    "Initial version.";
  reference
    "RFCXXXX: A YANG Data Model for Resource Reservation Protocol
    (RSVP)";
}

grouping graceful-restart-extended {
  description
    "Configuration parameters relating to RSVP Graceful-Restart.";
}

grouping authentication-extended {
  description
    "Configuration parameters relating to RSVP authentication.";
  leaf lifetime {
    type uint32 {
      range "30..86400";
    }
  }
}
```

```
    }
    units "seconds";
    default "30";
    description
      "Life time for each security association.";
    reference
      "RFC2747: RSVP Cryptographic Authentication";
  }
  leaf window-size {
    type uint32 {
      range "1..64";
    }
    default "2";
    description
      "Window-size to limit number of out-of-order messages.";
    reference
      "RFC2747: RSVP Cryptographic Authentication";
  }
  leaf challenge {
    type empty;
    description
      "Enable challenge messages.";
    reference
      "RFC2747: RSVP Cryptographic Authentication";
  }
  leaf retransmits {
    type uint32 {
      range "1..10000";
    }
    default "1";
    description
      "Number of retransmits when messages are dropped.";
    reference
      "RFC2747: RSVP Cryptographic Authentication";
  }
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
      "Key chain name to authenticate RSVP
       signaling messages.";
    reference
      "RFC2747: RSVP Cryptographic Authentication";
  }
}

grouping hellos-extended {
  description
    "Configuration parameters relating to RSVP hellos";
```

```
leaf interface-based {
  type empty;
  description
    "Enable interface-based Hello adjacency if present.";
}
leaf hello-interval {
  type uint32;
  units "milliseconds";
  default "9000";
  description
    "Configure interval between successive Hello messages in
    milliseconds.";
  reference
    "RFC3209: RSVP-TE: Extensions to RSVP for LSP Tunnels.
    RFC5495: Description of the Resource Reservation Protocol -
    Traffic-Engineered (RSVP-TE) Graceful Restart Procedures.";
}
leaf hello-misses {
  type uint32 {
    range "1..10";
  }
  default "3";
  description
    "Configure max number of consecutive missed Hello messages.";
  reference
    "RFC3209: RSVP-TE: Extensions to RSVP for LSP Tunnels.
    RFC5495: Description of the Resource Reservation Protocol -
    Traffic- Engineered (RSVP-TE) Graceful Restart Procedures.";
}
}

grouping signaling-parameters-extended {
  description
    "Configuration parameters relating to RSVP signaling";
  leaf refresh-interval {
    type uint32;
    units "seconds";
    default "30";
    description
      "Set interval between successive refreshes";
    reference "RFC2205";
  }
  leaf refresh-misses {
    type uint32;
    default "9";
    description
      "Set max number of consecutive missed messages for state
      expiry";
  }
}
```

```
        reference "RFC2205";
    }
    leaf checksum-enable {
        type empty;
        description
            "Enable RSVP message checksum computation";
        reference "RFC2205";
    }
    leaf patherr-state-removal {
        type empty;
        description
            "State-Removal flag in Path Error message if present.";
        reference "RFC3473";
    }
}

grouping refresh-reduction-extended {
    description
        "Configuration parameters relating to RSVP refresh reduction.";
    leaf bundle-message-max-size {
        type uint32 {
            range "512..65000";
        }
        default "1500";
        description
            "Configure maximum size (bytes) of a single RSVP Bundle
            message.";
        reference "RFC2961";
    }
    leaf ack-hold-time {
        type uint32;
        units "milliseconds";
        default "9000";
        description
            "Configure hold time in milliseconds for sending RSVP ACK
            message(s).";
        reference "RFC2961";
    }
    leaf ack-max-size {
        type uint32;
        default "1500";
        description
            "Configure max size of a single RSVP ACK message.";
        reference "RFC2961";
    }
    leaf ack-retransmit-time {
        type uint32;
        units "milliseconds";
    }
}
```



```
    default "500";
    description
      "Configure min delay in milliseconds to wait for an
       acknowledgment before being retransmitted.";
    reference "RFC2961";
  }
  leaf srefresh-ack-desired {
    type empty;
    description
      "Enables the sending of MESSAGE_ID with ACK_Desired
       set with Srefresh messages.";
    reference "RFC2961";
  }
  leaf srefresh-max-size {
    type uint32 {
      range "20..65000";
    }
    default "1500";
    description
      "Configure max size (bytes) of a single RSVP Srefresh
       message.";
    reference "RFC2961";
  }
  leaf srefresh-relative-period {
    type uint8 {
      range "10..100";
    }
    description
      "Configures the period of Srefreshes relative to standard
       refresh message period in percentage.";
  }
}

grouping packets-extended-statistics {
  description
    "Packet statistics.";
  leaf discontinuity-time {
    type yang:date-and-time;
    description
      "The time on the most recent occasion at which any one or
       more of the statistic 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 out-dropped {
    type yang:counter64;
  }
}
```

```
        description
            "Out RSVP packet drop count.";
    }
    leaf in-dropped {
        type yang:counter64;
        description
            "In RSVP packet drop count.";
    }
    leaf out-errors {
        type yang:counter64;
        description
            "Out RSVP packet errors count.";
    }
    leaf in-errors {
        type yang:counter64;
        description
            "In RSVP packet rx errors count.";
    }
}

/**
 * RSVP extensions augmentations
 */
/* RSVP graceful restart*/
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/"
    + "rsvp:graceful-restart" {
    description
        "RSVP graceful restart configuration extensions";
    uses graceful-restart-extended;
}

/**
 * RSVP all interfaces extensions
 */

/* RSVP interface signaling extensions */
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
    description
        "RSVP signaling all interfaces configuration extensions";
    uses signaling-parameters-extended;
}

/* Packet statistics extension */
augment "/rt:routing/rt:control-plane-protocols/"
```

```
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
    + "rsvp:statistics/rsvp:packets" {
  description
    "RSVP packets all interfaces configuration extensions";
  uses packets-extended-statistics;
}

/* RSVP refresh reduction extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:refresh-reduction" {
  description
    "RSVP refresh-reduction all interface configuration
    extensions";
  uses refresh-reduction-extended;
}

/* RSVP hellos extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:hellos" {
  description
    "RSVP hello all interfaces configuration extensions";
  uses hellos-extended;
}

/* RSVP authentication extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:authentication" {
  description
    "RSVP authentication all interfaces configuration extensions";
  uses authentication-extended;
}

/**
 * RSVP per interface extensions
 */
/* RSVP interface signaling extensions */

augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:interface" {
  description
    "RSVP signaling interface configuration extensions";
  uses signaling-parameters-extended;
}
```

```
/* Packet statistics extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:interface/rsvp:statistics/rsvp:packets" {
  description
    "RSVP packet stats extensions";
  uses packets-extended-statistics;
}

/* RSVP refresh reduction extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:interface/rsvp:refresh-reduction" {
  description
    "RSVP refresh-reduction interface configuration extensions";
  uses refresh-reduction-extended;
}

/* RSVP hellos extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:interface/rsvp:hellos" {
  description
    "RSVP hello interface configuration extensions";
  uses hellos-extended;
}

/* RSVP authentication extension */
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
  + "rsvp:interface/rsvp:authentication" {
  description
    "RSVP authentication interface configuration extensions";
  uses authentication-extended;
}
}
<CODE ENDS>
```

Figure 5: RSVP extended YANG module

## 6. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

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

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

name: ietf-rsvp  
namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp  
prefix: rsvp  
reference: RFCXXXX

name: ietf-rsvp-extended  
namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended  
prefix: rsvp-extended  
reference: RFCXXXX

## 7. Security Considerations

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

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

There are a number of data nodes defined in the YANG module(s) defined in this document that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
rsvp:rsvp/ /rsvp:globals /rsvp:interfaces /rsvp:sessions

All of which are considered sensitive and if access to either of these is compromised, it can result in temporary network outages or be employed to mount DoS attacks.

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:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
rsvp:rsvp/ /rsvp:globals /rsvp:interfaces /rsvp:sessions
```

Additional information from these state data nodes can be inferred with respect to the network topology, and device location and subsequently be used to mount other attacks in the network.

For RSVP authentication, the configuration supported is via the specification of key-chains [RFC8177] or the direct specification of key and authentication algorithm, and hence security considerations of [RFC8177] are inherited. This includes the considerations with respect to the local storage and handling of authentication keys.

Some of the RPC operations defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. The RSVP YANG module support the "clear-session" and "clear-neighbor" RPCs. If access to either of these is compromised, they can result in temporary network outages be employed to mount DoS attacks.

The security considerations spelled out in the YANG 1.1 specification [RFC7950] apply for this document as well.

## 8. Acknowledgement

The authors would like to thank Tom Petch for reviewing and providing useful feedback about the document. The authors would also like to thank Lou Berger for reviewing and providing valuable feedback on this document.

## 9. Appendix A

A simple network setup is shown in {fig-example title}. R1 runs the RSVP routing protocol on both interfaces 'ge0/0/0/1', and 'ge0/0/0/2'.

State on R1:

Sessions:

=====

Destination	Protocol-ID	Dest-port
198.51.100.1	10	10

Neighbors:

=====

Neighbor Address	Interface
192.0.2.6	ge0/0/0/1

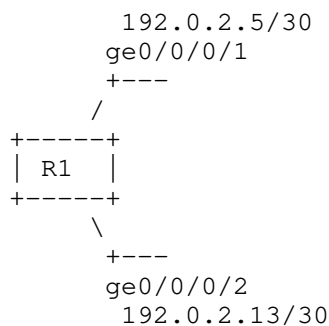


Figure 6: Example of network configuration.

The instance data tree could then be as follows:

```

{
  "ietf-routing:routing": {
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "rt:routing-protocol",
          "name": "rsvp:rsvp",
          "ietf-rsvp:rsvp": {
            "interfaces": {
              "refresh-reduction": {
                "enabled": true,
                "ietf-rsvp-extended:bundle-message-max-size": 2000,
                "ietf-rsvp-extended:reliable-ack-hold-time": 180,
                "ietf-rsvp-extended:reliable-ack-max-size": 2000,
                "ietf-rsvp-extended:reliable-retransmit-time": 180,
                "ietf-rsvp-extended:reliable-srefresh": [
                  null
                ],
                "ietf-rsvp-extended:summary-max-size": 2000
              }
            }
          }
        }
      ]
    }
  }
}

```

```
"hellos": {
  "enabled": true,
  "ietf-rsvp-extended:interface-based": [
    null
  ],
  "ietf-rsvp-extended:hello-interval": 27000,
  "ietf-rsvp-extended:hello-misses": 3
},
"statistics": {
  "messages": {
    "ack-sent": "777",
    "ack-received": "4840",
    "bundle-sent": "2195",
    "bundle-received": "293",
    "hello-sent": "2516",
    "hello-received": "3535",
    "integrity-challenge-sent": "2737",
    "integrity-challenge-received": "2330",
    "integrity-response-sent": "895",
    "integrity-response-received": "1029",
    "path-sent": "1197",
    "path-received": "3568",
    "path-err-sent": "4658",
    "path-err-received": "695",
    "path-tear-sent": "3706",
    "path-tear-received": "2604",
    "resv-sent": "3353",
    "resv-received": "3129",
    "resv-err-sent": "1787",
    "resv-err-received": "3205",
    "resv-tear-sent": "4465",
    "resv-tear-received": "3056",
    "summary-refresh-sent": "655",
    "summary-refresh-received": "3856"
  },
  "packets": {
    "sent": "2147",
    "received": "4374",
    "ietf-rsvp-extended:discontinuity-time":
      "2015-10-24T17:11:27+02:00",
    "ietf-rsvp-extended:out-dropped": "2696",
    "ietf-rsvp-extended:in-dropped": "941",
    "ietf-rsvp-extended:out-errors": "19",
    "ietf-rsvp-extended:in-errors": "2732"
  },
  "errors": {
    "authenticate": "2540",
    "checksum": "2566",
```



```
        "packet-length": "267"
      }
    },
    "interface": [
      {
        "interface": "ge0/0/0/1",
        "statistics": {
          "messages": {
            "ack-sent": "2747",
            "ack-received": "4934",
            "bundle-sent": "1618",
            "bundle-received": "3668",
            "hello-sent": "4288",
            "hello-received": "1194",
            "integrity-challenge-sent": "4850",
            "integrity-challenge-received": "3979",
            "integrity-response-sent": "479",
            "integrity-response-received": "1773",
            "path-sent": "2230",
            "path-received": "1793",
            "path-err-sent": "465",
            "path-err-received": "1859",
            "path-tear-sent": "923",
            "path-tear-received": "3924",
            "resv-sent": "3203",
            "resv-received": "2507",
            "resv-err-sent": "1259",
            "resv-err-received": "2445",
            "resv-tear-sent": "3045",
            "resv-tear-received": "4676",
            "summary-refresh-sent": "365",
            "summary-refresh-received": "2129"
          },
          "packets": {
            "sent": "847",
            "received": "3114",
            "ietf-rsvp-extended:discontinuity-time":
              "2015-10-24T17:11:27+02:00",
            "ietf-rsvp-extended:out-dropped": "1841",
            "ietf-rsvp-extended:in-dropped": "4832",
            "ietf-rsvp-extended:out-errors": "1334",
            "ietf-rsvp-extended:in-errors": "3900"
          },
          "errors": {
            "authenticate": "3494",
            "checksum": "4374",
            "packet-length": "2456"
          }
        }
      }
    ]
  }
}
```

```
    }  
  },  
  {  
    "interface": "ge0/0/0/2",  
    "statistics": {  
      "messages": {  
        "ack-sent": "1276",  
        "ack-received": "2427",  
        "bundle-sent": "4053",  
        "bundle-received": "3509",  
        "hello-sent": "3261",  
        "hello-received": "2863",  
        "integrity-challenge-sent": "4744",  
        "integrity-challenge-received": "3554",  
        "integrity-response-sent": "3155",  
        "integrity-response-received": "169",  
        "path-sent": "3853",  
        "path-received": "409",  
        "path-err-sent": "4227",  
        "path-err-received": "2830",  
        "path-tear-sent": "1742",  
        "path-tear-received": "3344",  
        "resv-sent": "3154",  
        "resv-received": "3492",  
        "resv-err-sent": "3112",  
        "resv-err-received": "3974",  
        "resv-tear-sent": "3657",  
        "resv-tear-received": "533",  
        "summary-refresh-sent": "4036",  
        "summary-refresh-received": "2123"  
      },  
      "packets": {  
        "sent": "473",  
        "received": "314",  
        "ietf-rsvp-extended:discontinuity-time":  
          "2015-10-24T17:11:27+02:00",  
        "ietf-rsvp-extended:out-dropped": "2042",  
        "ietf-rsvp-extended:in-dropped": "90",  
        "ietf-rsvp-extended:out-errors": "1210",  
        "ietf-rsvp-extended:in-errors": "1361"  
      },  
      "errors": {  
        "authenticate": "543",  
        "checksum": "2241",  
        "packet-length": "480"  
      }  
    }  
  }  
}
```

```
    ],
    "ietf-rsvp-extended:refresh-interval": 30,
    "ietf-rsvp-extended:refresh-misses": 5,
    "ietf-rsvp-extended:checksum_enabled": true,
    "ietf-rsvp-extended:patherr-state-removal": [
      null
    ]
  },
  "sessions": {
    "session-ip": [
      {
        "destination-port": 10,
        "protocol-id": 10,
        "destination": "198.51.100.1",
        "psbs": {
          "psb": [
            {
              "source-port": 10,
              "expires-in": 100
            }
          ]
        },
        "rsbs": {
          "rsb": [
            {
              "source-port": 10,
              "reservation-style":
                "rsvp:reservation-wildcard-filter",
              "expires-in": 100
            }
          ]
        }
      ]
    ]
  },
  "neighbors": {
    "neighbor": [
      {
        "address": "192.0.2.6",
        "epoch": 130,
        "expiry-time": 260,
        "graceful-restart": {
          "enabled": true,
          "local-restart-time": 271,
          "local-recovery-time": 138,
          "neighbor-restart-time": 341,
          "neighbor-recovery-time": 342
        }
      }
    ]
  }
}
```



Himanshu Shah  
Ciena

Email: hshah@ciena.com

Xia Chen  
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones  
Brocade

Email: raqib@Brocade.com

Bin Wen  
Comcast

Email: Bin\_Wen@cable.comcast.com

## 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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [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>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [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>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [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>>.
- [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>>.

- [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>>.
- [RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <<https://www.rfc-editor.org/info/rfc8639>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.

## 11.2. Informative References

- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <<https://www.rfc-editor.org/info/rfc2747>>.
- [RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, DOI 10.17487/RFC2961, April 2001, <<https://www.rfc-editor.org/info/rfc2961>>.
- [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>>.
- [rfc4558] Ali, Z., Rahman, R., Prairie, D., and D. Papadimitriou, "Node-ID Based Resource Reservation Protocol (RSVP) Hello: A Clarification Statement", RFC 4558, DOI 10.17487/RFC4558, June 2006, <<https://www.rfc-editor.org/info/rfc4558>>.

- [RFC5063] Satyanarayana, A., Ed. and R. Rahman, Ed., "Extensions to GMPLS Resource Reservation Protocol (RSVP) Graceful Restart", RFC 5063, DOI 10.17487/RFC5063, October 2007, <<https://www.rfc-editor.org/info/rfc5063>>.
- [RFC5495] Li, D., Gao, J., Satyanarayana, A., and S. Bardalai, "Description of the Resource Reservation Protocol - Traffic-Engineered (RSVP-TE) Graceful Restart Procedures", RFC 5495, DOI 10.17487/RFC5495, March 2009, <<https://www.rfc-editor.org/info/rfc5495>>.

## Authors' Addresses

Vishnu Pavan Beeram  
Juniper Networks

Email: [vbeeram@juniper.net](mailto:vbeeram@juniper.net)

Tarek Saad  
Juniper Networks

Email: [tsaad@juniper.net](mailto:tsaad@juniper.net)

Rakesh Gandhi  
Cisco Systems, Inc.

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

Xufeng Liu  
Volta Networks

Email: [xufeng.liu.ietf@gmail.com](mailto:xufeng.liu.ietf@gmail.com)

Igor Bryskin  
Individual

Email: [i\\_bryskin@yahoo.com](mailto:i_bryskin@yahoo.com)



TEAS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 11 August 2022

T. Saad  
Juniper Networks  
R. Gandhi  
Cisco Systems Inc  
X. Liu  
Volta Networks  
V.P. Beeram  
Juniper Networks  
I. Bryskin  
Individual  
O. Gonzalez de Dios  
Telefonica  
7 February 2022

A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths  
and Interfaces  
draft-ietf-teas-yang-te-29

Abstract

This document defines a YANG data model for the provisioning and management of Traffic Engineering (TE) tunnels, Label Switched Paths (LSPs), and interfaces. The model is divided into YANG modules that classify data into generic, device-specific, technology agnostic, and technology-specific elements.

This model covers data for configuration, operational state, remote procedural calls, and event notifications.

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 11 August 2022.

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Requirements Language . . . . .	3
2.1. Prefixes in Data Node Names . . . . .	4
2.2. Model Tree Diagrams . . . . .	4
3. Design Considerations . . . . .	5
3.1. State Data Organization . . . . .	5
4. Model Overview . . . . .	6
4.1. Module Relationship . . . . .	6
5. TE YANG Model . . . . .	7
5.1. Module Structure . . . . .	7
5.1.1. TE Globals . . . . .	9
5.1.2. TE Tunnels . . . . .	12
5.1.3. TE LSPs . . . . .	19
5.2. Tree Diagram . . . . .	19
5.3. YANG Module . . . . .	60
6. TE Device YANG Model . . . . .	98
6.1. Module Structure . . . . .	99
6.1.1. TE Interfaces . . . . .	99
6.2. Tree Diagram . . . . .	100
6.3. YANG Module . . . . .	102
7. Notifications . . . . .	116
8. TE Generic and Helper YANG Modules . . . . .	117
9. IANA Considerations . . . . .	117
10. Security Considerations . . . . .	117
11. Acknowledgement . . . . .	119
12. Contributors . . . . .	119
13. Appendix A: Data Tree Examples . . . . .	119
13.1. Basic Tunnel Setup . . . . .	120
13.2. Global Named Path Constraints . . . . .	121
13.3. Tunnel with Global Path Constraint . . . . .	121
13.4. Tunnel with Per-tunnel Path Constraint . . . . .	122
13.5. Tunnel State . . . . .	123

14. References	124
14.1. Normative References	124
14.2. Informative References	127
Authors' Addresses	128

## 1. Introduction

YANG [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]. YANG has proved relevant beyond its initial confines, as bindings to other interfaces (e.g. RESTCONF [RFC8040]) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document describes YANG data model for Traffic Engineering (TE) tunnels, Label Switched Paths (LSPs), and interfaces. The model covers data applicable to generic or device-independent, device-specific, and Multiprotocol Label Switching (MPLS) technology specific.

The document describes a high-level relationship between the modules defined in this document, as well as other external protocol YANG modules. The TE generic YANG data model does not include any data specific to a signaling protocol. It is expected other data plane technology model(s) will augment the TE generic YANG data model.

Also, it is expected other YANG module(s) that model TE signaling protocols, such as RSVP-TE ([RFC3209], [RFC3473]), or Segment-Routing TE (SR-TE) [I-D.ietf-spring-segment-routing-policy] will augment the generic TE YANG module.

## 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.

The following terms are defined in [RFC6241] and are used in this specification:

- \* client
- \* configuration data

\* state data

This document also makes use of the following terminology introduced in the YANG Data Modeling Language [RFC7950]:

\* augment

\* data model

\* data node

## 2.1. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt-types	ietf-routing-types	[RFC8294]
te-types	ietf-te-types	[RFC8776]
te-packet-types	ietf-te-packet-types	[RFC8776]
te	ietf-te	this document
te-dev	ietf-te-device	this document

Table 1: Prefixes and corresponding YANG modules

## 2.2. Model Tree Diagrams

The tree diagrams extracted from the module(s) defined in this document are given in subsequent sections as per the syntax defined in [RFC8340].

### 3. Design Considerations

This document describes a generic TE YANG data model that is independent of any dataplane technology. One of the design objectives is to allow specific data plane technology models to reuse the TE generic data model and possibly augment it with technology specific data.

The elements of the generic TE YANG data model, including TE Tunnels, LSPs, and interfaces have leaf(s) that identify the technology layer where they reside. For example, the LSP encoding type can identify the technology associated with a TE Tunnel or LSP.

Also, the generic TE YANG data model does not cover signaling protocol data. The signaling protocol used to instantiate TE LSPs are outside the scope of this document and expected to be covered by augmentations defined in other document(s).

The following other design considerations are taken into account with respect data organization:

- \* The generic TE YANG data model 'ietf-te' contains device independent data and can be used to model data off a device (e.g. on a TE controller). The device-specific TE data is defined in module 'ietf-te-device' as shown in Figure 1,
- \* In general, minimal elements in the model are designated as "mandatory" to allow freedom to vendors to adapt the data model to their specific product implementation.
- \* Suitable defaults are specified for all configurable elements.
- \* The model declares a number of TE functions as features that can be optionally supported.

#### 3.1. State Data Organization

The Network Management Datastore Architecture (NMDA) [RFC8342] addresses modeling state data for ephemeral objects. This document adopts the NMDA model for configuration and state data representation as per IETF guidelines for new IETF YANG models.

#### 4. Model Overview

The data models defined in this document cover the core TE features that are commonly supported by different vendor implementations. The support of extended or vendor specific TE feature(s) is expected to be in either augmentations, or deviations to the model defined in this document.

##### 4.1. Module Relationship

The generic TE YANG data model that is defined in "ietf-te.yang" covers the building blocks that are device independent and agnostic of any specific technology or control plane instances. The TE device model defined in "ietf-te-device.yang" augments the generic TE YANG data model and covers data that is specific to a device - for example, attributes of TE interfaces, or TE timers that are local to a TE node.

The TE data model for specific instances of data plane technology exist in a separate YANG module(s) that augment the generic TE YANG data model. For example, the MPLS-TE module "ietf-te-mpls.yang" is defined in another document and augments the TE generic model as shown in Figure 1.

The TE data model for specific instances of signaling protocol are outside the scope of this document and are defined in other documents. For example, the RSVP-TE YANG model augmentation of the TE model is covered in [I-D.ietf-teas-yang-rsvp].

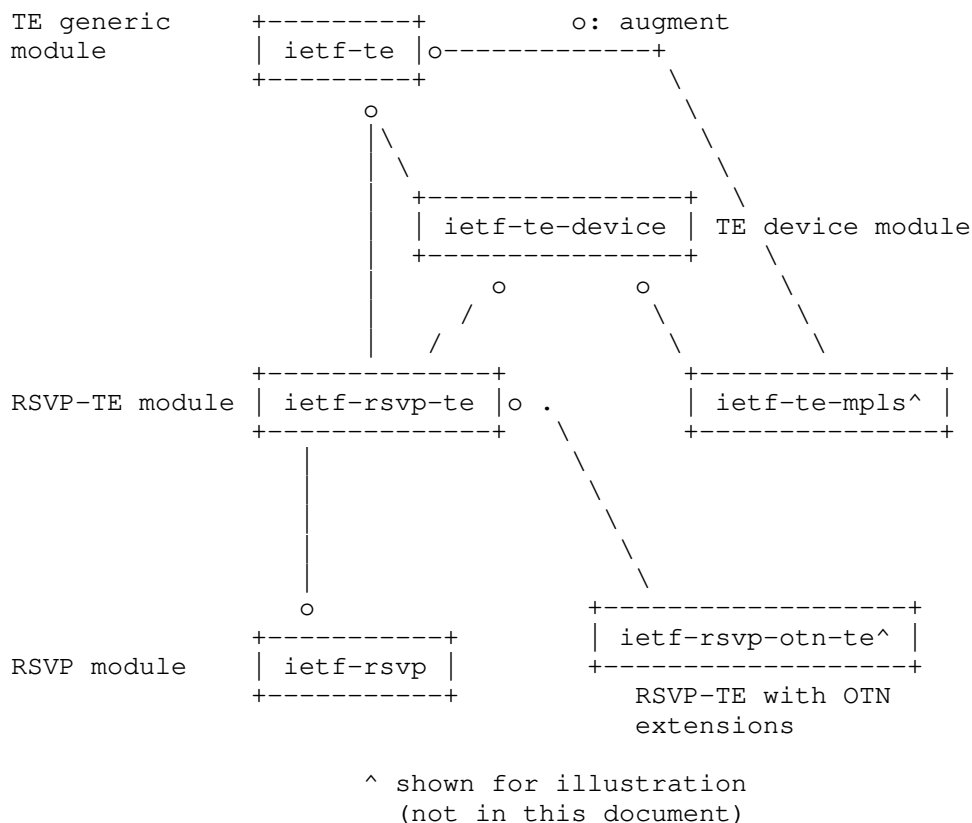


Figure 1: Relationship of TE module(s) with signaling protocol modules

## 5. TE YANG Model

The generic TE YANG module ('ietf-te') is meant to manage and operate a TE network. This includes creating, modifying and retrieving TE Tunnels, LSPs, and interfaces and their associated attributes (e.g. Administrative-Groups, SRLGs, etc.).

The detailed tree structure is provided in Figure 2.

### 5.1. Module Structure

The 'ietf-te' uses three main containers grouped under the main 'te' container (see Figure 2). The 'te' container is the top level container in the data model. The presence of the 'te' container enables TE function system wide. Below provides further descriptions of containers that exist under the 'te' top level container.

**globals:**

The 'globals' container maintains the set of global TE attributes that can be applicable to TE Tunnel(s) and interface(s).

**tunnels:**

The 'tunnels' container includes the list of TE Tunnels that are instantiated. Refer to Section 5.1.2 for further details on the properties of a TE Tunnel.

**lsps:**

The 'lsps' container includes the list of TE LSP(s) that are instantiated for TE Tunnels. Refer to Section 5.1.3 for further details on the properties of a TE LSP.

**tunnels-path-compute:**

A Remote Procedure Call (RPC) to request path computation for a specific TE Tunnel. The RPC allows requesting path computation using atomic and stateless operation. A tunnel may also be configured in 'compute-only' mode to provide stateful path updates - see Section 5.1.2 for further details.

**tunnels-action:**

An RPC to request a specific action (e.g. reoptimize, or tear-and-setup) to be taken on a specific tunnel or all tunnels.

```
module: ietf-te
  +--rw te!
    +--rw globals
      .
      .
    +--rw tunnels
      .
      .
    +-- lsps
```

```
rpcs:
  +---x tunnels-path-compute
  +---x tunnels-action
```

Figure 2: TE Tunnel model high-level YANG tree view



### 5.1.1. TE Globals

The 'globals' container covers properties that control TE features behavior system-wide, and its respective state (see Figure 3). The TE globals configuration include:

```

+--rw globals
|   +--rw named-admin-groups
|   |   +--rw named-admin-group* [name]
|   ..
|   +--rw named-srlgs
|   |   +--rw named-srlg* [name] {te-types:named-srlg-groups}?
|   ..
|   +--rw named-path-constraints
|   |   +--rw named-path-constraint* [name]
|   ..

```

Figure 3: TE globals YANG subtree high-level structure

#### named-admin-groups:

A YANG container for the list of named (extended) administrative groups that may be applied to TE links.

#### named-srlgs:

A YANG container for the list named Shared Risk Link Groups (SRLGs) that may be applied to TE links.

#### named-path-constraints:

A YANG container for a list of named path constraints. Each named path constraint is composed of a set of constraints that can be applied during path computation. A named path constraint can be applied to multiple TE Tunnels. Path constraints may also be specified directly under the TE Tunnel. The path constraint specified under the TE Tunnel take precedence over the path constraints derived from the referenced named path constraint. A named path constraint entry can be formed up of the following path constraints:

```

|   +--rw named-path-constraints
|       +--rw named-path-constraint* [name]
|           +--rw name                               string
|           +--rw te-bandwidth
| // ...
|       +--rw link-protection?                       identityref
|       +--rw setup-priority?                         uint8
|       +--rw hold-priority?                         uint8
|       +--rw signaling-type?                       identityref
|       +--rw path-metric-bounds
| // ...
|       +--rw path-affinities-values
| // ...
|       +--rw path-affinity-names
| // ...
|       +--rw path-srlgs-lists
| // ...
|       +--rw path-srlgs-names
| // ...
|       +--rw disjointness?
|           |   te-path-disjointness
| // ...
|       +--rw explicit-route-objects-always
| // ...
|       |   +--rw route-object-exclude-always* [index]
|       |   +--rw route-object-include-exclude* [index]

```

Figure 4: Named path constraints YANG subtree

- o te-bandwidth: A YANG container that holds the technology agnostic TE bandwidth constraint.
- o link-protection: A YANG leaf that holds the link protection type constraint required for the links to be included in the computed path.
- o setup/hold priority: A YANG leaf that holds the LSP setup and hold admission priority as defined in [RFC3209].
- o signaling-type: A YANG leaf that holds the LSP setup type, such as RSVP-TE or SR.
- o path-metric-bounds: A YANG container that holds the set of metric bounds applicable on the computed TE tunnel path.

- o `path-affinities-values`: A YANG container that holds the set of affinity values and mask to be used during path computation.
- o `path-affinity-names`: A YANG container that holds the set of named affinity constraints and corresponding inclusion or exclusions instruction for each to be used during path computation.
- o `path-srlgs-lists`: A YANG container that holds the set of SRLG values and corresponding inclusion or exclusions instruction to be used during path computation.
- o `path-srlgs-names`: A YANG container that holds the set of named SRLG constraints and corresponding inclusion or exclusions instruction for each to be used during path computation.
- o `disjointness`: The level of resource disjointness constraint that the secondary path of a TE tunnel has to adhere to.
- o `explicit-route-objects-always`: A YANG container that contains two route objects lists:
  - + `'route-object-exclude-always'`: a list of route entries to always exclude from the path computation.
  - + `'route-object-include-exclude'`: a list of route entries to include or exclude in the path computation.

The `'route-object-include-exclude'` is used to configure constraints on which route objects (e.g., nodes, links) are included or excluded in the path computation.

The interpretation of an empty `'route-object-include-exclude'` list depends on the TE Tunnel (end-to-end or Tunnel Segment) and on the specific path, according to the following rules:

1. An empty `'route-object-include-exclude'` list for the primary path of an end-to-end TE Tunnel indicates that there are no route objects to be included or excluded in the path computation.
2. An empty `'route-object-include-exclude'` list for the primary path of a TE Tunnel Segment indicates that no primary LSP is required for that TE Tunnel.

3. An empty 'route-object-include-exclude' list for a reverse path means it always follows the forward path (i.e., the TE Tunnel is co-routed). When the 'route-object-include-exclude' list is not empty, the reverse path is routed independently of the forward path.
4. An empty 'route-object-include-exclude' list for the secondary (forward) path indicates that the secondary path has the same endpoints as the primary path.

#### 5.1.2. TE Tunnels

The 'tunnels' container holds the list of TE Tunnels that are provisioned on devices in the network (see Figure 5).

A TE Tunnel in the list is uniquely identified by a name. When the model is used to manage a specific device, the 'tunnels' list contains the TE Tunnels originating from the specific device. When the model is used to manage a TE controller, the 'tunnels' list contains all TE Tunnels and TE tunnel segments originating from device(s) that the TE controller manages.

The TE Tunnel model allows the configuration and management of the following TE tunnel related objects:

##### TE Tunnel:

A YANG container of one or more LSPs established between the source and destination TE Tunnel termination points. A TE Tunnel LSP is a connection-oriented service provided by the network layer for the delivery of client data between a source and the destination of the TE Tunnel termination points.

##### TE Tunnel Segment:

A part of a multi-domain TE Tunnel that is within a specific network domain.

```

+--rw tunnels
|   +--rw tunnel* [name]
|   |   +--rw name                               string
|   |   +--rw alias?                             string
|   |   +--rw identifier?                         uint32
|   |   +--rw color?                             uint32
|   |   +--rw description?                       string
|   |   +--ro operational-state?                 identityref
|   |   +--rw encoding?                         identityref
|   |   +--rw switching-type?                   identityref
|   |   +--rw admin-state?                     identityref
|   |   +--rw reoptimize-timer?                 uint16
|   |   +--rw source?                           te-types:te-node-id
|   |   +--rw destination?                     te-types:te-node-id
|   |   +--rw src-tunnel-tp-id?                 binary
|   |   +--rw dst-tunnel-tp-id?                 binary
|   |   +--rw controller
|   |   |   +--rw protocol-origin?               identityref
|   |   |   +--rw controller-entity-id?         string
|   |   +--rw bidirectional?                   boolean
|   |   +--rw association-objects
|   |   |   +--rw association-object* [association-key]
|   |
|   |   // ..
|   |   |
|   |   +--rw protection
|   |
|   |   // ..
|   |   |
|   |   +--rw restoration
|   |
|   |   // ..
|   |   |
|   |   +--rw te-topology-identifier
|   |
|   |   // ..
|   |   |
|   |   +--rw hierarchy
|   |
|   |   // ..

```

Figure 5: TE Tunnel list YANG subtree structure

The TE Tunnel has a number of attributes that are set directly under the tunnel (see Figure 5). The main attributes of a TE Tunnel are described below:

**operational-state:**

A YANG leaf that holds the operational state of the tunnel.

**name:**

A YANG leaf that holds the name of a TE Tunnel. The name of the TE Tunnel uniquely identifies the tunnel within the TE tunnel list. The name of the TE Tunnel can be formatted as a Uniform

Resource Indicator (URI) by including the namespace to ensure uniqueness of the name amongst all the TE Tunnels present on devices and controllers.

alias:

A YANG leaf that holds an alternate name to the TE tunnel. Unlike the TE tunnel name, the alias can be modified at any time during the lifetime of the TE tunnel.

identifier:

A YANG leaf that holds an identifier of the tunnel. This identifier is unique amongst tunnels originated from the same ingress device.

color:

A YANG leaf that holds the color associated with the TE tunnel. The color is used to map or steer services that carry matching color on to the TE tunnel as described in [RFC9012].

encoding/switching:

The 'encoding' and 'switching-type' are YANG leafs that define the specific technology in which the tunnel operates in as described in [RFC3945].

reoptimize-timer:

A YANG leaf to set the interval period for tunnel reoptimization.

source/destination:

YANG leafs that define the tunnel source and destination node endpoints.

src-tunnel-tp-id/dst-tunnel-tp-id:

YANG leafs that hold the identifiers of source and destination TE Tunnel Termination Points (TTPs) [RFC8795] residing on the source and destination nodes. The TTP identifiers are optional on nodes that have a single TTP per node. For example, TTP identifiers are optional for packet (IP/MPLS) routers.

controller:

A YANG container that holds tunnel data relevant to an optional external TE controller that may initiate or control a tunnel. This target node may be augmented by external module(s), for example, to add data for PCEP initiated and/or delegated tunnels.

bidirectional:

A YANG leaf that when present indicates the LSPs of a TE Tunnel are bidirectional and co-routed.

association-objects:

A YANG container that holds the set of associations of the TE Tunnel to other TE Tunnels. Associations at the TE Tunnel level apply to all paths of the TE Tunnel. The TE tunnel associations can be overridden by associations configured directly under the TE Tunnel path.

protection:

A YANG container that holds the TE Tunnel protection properties.

restoration:

A YANG container that holds the TE Tunnel restoration properties.

te-topology-identifier:

A YANG container that holds the topology identifier associated with the topology where paths for the TE tunnel are computed.

```

+--rw hierarchy
|   +--rw dependency-tunnels
|   |   +--rw dependency-tunnel* [name]
|   |   |   +--rw name
|   |   |   |   -> ../../../../tunnels/tunnel/name
|   |   |   +--rw encoding?          identityref
|   |   |   +--rw switching-type?    identityref
|   |   +--rw hierarchical-link
|   |   |   +--rw local-te-node-id?    te-types:te-node-id
|   |   |   +--rw local-te-link-tp-id? te-types:te-tp-id
|   |   |   +--rw remote-te-node-id?   te-types:te-node-id
|   |   +--rw te-topology-identifier
|   |   |   +--rw provider-id?    te-global-id
|   |   |   +--rw client-id?      te-global-id
|   |   |   +--rw topology-id?    te-topology-id

```

Figure 6: TE Tunnel hierarchy YANG subtree

#### hierarchy:

A YANG container that holds hierarchy related properties of the TE Tunnel (see Figure 6. A TE LSP can be set up in MPLS or Generalized MPLS (GMPLS) networks to be used as a TE links to carry traffic in other (client) networks [RFC6107]. In this case, the model introduces the TE Tunnel hierarchical link endpoint parameters to identify the specific link in the client layer that the underlying TE Tunnel is associated with. The hierarchy container includes the following:

- o dependency-tunnels: A set of hierarchical TE Tunnels provisioned or to be provisioned in the immediate lower layer that this TE tunnel depends on for multi-layer path computation. A dependency TE Tunnel is provisioned if and only if it is used (selected by path computation) at least by one client layer TE Tunnel. The TE link in the client layer network topology supported by a dependent TE Tunnel is dynamically created only when the dependency TE Tunnel is actually provisioned.
- o hierarchical-link: A YANG container that holds the identity of the hierarchical link (in the client layer) that is supported by this TE Tunnel. The endpoints of the hierarchical link are defined by TE tunnel source and destination node endpoints. The hierarchical link can be identified by its source and destination link termination point identifiers.

#### 5.1.2.1. TE Tunnel Paths

The TE Tunnel can be configured with a set of paths that define the tunnel forward and reverse paths as described in Figure 7. Moreover, a primary path can be specified a set of candidate secondary paths that can be visited to support path protection. The following describe further the list of paths associated with a TE Tunnel.



```

|      +--rw primary-paths
|      |   +--rw primary-path* [name]
|      |   |   +--rw name
|      |   |   |   string
|  // ..
|      |   +
|      |   +--rw primary-reverse-path
|      |   |   +--rw name?
|      |   |   |   string
|  // ..
|      |   |
|      |   |   +--rw candidate-secondary-reverse-paths
|      |   |   |   +--rw candidate-secondary-reverse-path*
|      |   |   |   |   [secondary-path]
|      |   |   |   |   +--rw secondary-path
|      |   |   |   |   |   leafref
|      |   |   +--rw candidate-secondary-paths
|      |   |   |   +--rw candidate-secondary-path* [secondary-path]
|      |   |   |   |   +--rw secondary-path
|      |   |   |   |   |   leafref
|      |   |   |   |   +--ro active?
|      |   |   |   |   |   boolean
|
|      +--rw secondary-paths
|      |   +--rw secondary-path* [name]
|      |   |   +--rw name
|      |   |   |   string
|  // ..
|      +--rw secondary-reverse-paths
|      |   +--rw secondary-reverse-path* [name]
|      |   |   +--rw name
|      |   |   |   string

```

Figure 7: TE Tunnel paths YANG tree structure

**primary-paths:**

A YANG container that holds the list of primary paths. A primary path is identified by 'name'. A primary path is selected from the list to instantiate a primary forwarding LSP for the tunnel. The list of primary paths is visited by order of preference. A primary path has the following attributes:

- primary-reverse-path: A YANG container that holds properties of the primary reverse path. The reverse path is applicable to bidirectional TE Tunnels.
- candidate-secondary-paths: A YANG container that holds a list of candidate secondary paths which may be used for the primary path to support path protection. The candidate secondary path(s) reference path(s) from the tunnel secondary paths list. The preference of the secondary paths is specified within the list and dictates the order of visiting the secondary path from the list. The attributes of a secondary path can be defined

separately from the primary path. The attributes of a secondary path will be inherited from the associated 'active' primary when not explicitly defined for the secondary path.

#### secondary-paths:

A YANG container that holds the set of secondary paths. A secondary path is identified by 'name'. A secondary path can be referenced from the TE Tunnel's 'candidate-secondary-path' list. A secondary path contains attributes similar to a primary path.

#### secondary-reverse-paths:

A YANG container that holds the set of secondary reverse paths. A secondary reverse path is identified by 'name'. A secondary reverse path can be referenced from the TE Tunnel's 'candidate-secondary-reverse-paths' list. A secondary reverse path contains attributes similar to a primary path.

The following set common path attributes are shared for primary forward and reverse primary and secondary paths:

#### compute-only:

A path of TE Tunnel is, by default, provisioned so that it can be instantiated in forwarding to carry traffic as soon as a valid path is computed. In some cases, a TE path may be provisioned for the only purpose of computing a path and reporting it without the need to instantiate the LSP or commit any resources. In such a case, the path is configured in 'compute-only' mode to distinguish it from the default behavior. A 'compute-only' path is configured as a usual with the associated per path constraint(s) and properties on a device or TE controller. The device or TE controller computes the feasible path(s) subject to configured constraints. A client may query the 'compute-only' computed path properties 'on-demand', or alternatively, can subscribe to be notified of computed path(s) and whenever the path properties change.

#### use-path-computation:

A YANG leaf that indicates whether or not path computation is to be used for a specified path.

#### lockdown:

A YANG leaf that when set indicates the existing path should not be reoptimized after a failure on any of its traversed links.

te-topology-identifier:

A YANG container that holds the topology identifier associated with the tunnel.

optimizations:

a YANG container that holds the optimization objectives that path computation will use to select a path.

computed-paths-properties: > A YANG container that holds properties for the list of computed paths.

computed-path-error-infos:

A YANG container that holds a list of errors related to the path.

lsps:

a YANG container that holds a list of LSPs that are instantiated for this specific path.

#### 5.1.3. TE LSPs

The 'lsps' container includes the set of TE LSP(s) that are instantiated. A TE LSP is identified by a 3-tuple ('tunnel-name', 'node', 'lsp-id').

When the model is used to manage a specific device, the 'lsps' list contains all TE LSP(s) that traverse the device (including ingressing, transiting and egressing the device).

When the model is used to manage a TE controller, the 'lsps' list contains all TE LSP(s) that traverse all network devices (including ingressing, transiting and egressing the device) that the TE controller manages.

#### 5.2. Tree Diagram

Figure 8 shows the tree diagram of the generic TE YANG model defined in modules 'ietf-te.yang'.

```

module: ietf-te
  +--rw te!
    +--rw globals
      +--rw named-admin-groups
        +--rw named-admin-group* [name]
          {te-types:extended-admin-groups,te-types:named-extend
ed-admin-groups}?
          +--rw name string
          +--rw bit-position? uint32
      +--rw named-srlgs
        +--rw named-srlg* [name] {te-types:named-srlg-groups}?
          +--rw name string
          +--rw value? te-types:srlg
          +--rw cost? uint32
      +--rw named-path-constraints
        +--rw named-path-constraint* [name]
          {te-types:named-path-constraints}?
          +--rw name string
          +--rw te-bandwidth
            +--rw (technology)?
              +--:(generic)
                +--rw generic? te-bandwidth
          +--rw link-protection? identityref
          +--rw setup-priority? uint8
          +--rw hold-priority? uint8
          +--rw signaling-type? identityref
          +--rw path-metric-bounds
            +--rw path-metric-bound* [metric-type]
              +--rw metric-type identityref
              +--rw upper-bound? uint64
          +--rw path-affinities-values
            +--rw path-affinities-value* [usage]
              +--rw usage identityref
              +--rw value? admin-groups
          +--rw path-affinity-names
            +--rw path-affinity-name* [usage]
              +--rw usage identityref
              +--rw affinity-name* [name]
                +--rw name string
          +--rw path-srlgs-lists
            +--rw path-srlgs-list* [usage]
              +--rw usage identityref
              +--rw values* srlg
          +--rw path-srlgs-names
            +--rw path-srlgs-name* [usage]
              +--rw usage identityref
              +--rw names* string
          +--rw disjointness?

```

```

    te-path-disjointness
+--rw explicit-route-objects-always
+--rw route-object-exclude-always* [index]
+--rw index                               uint32
+--rw (type)?
+--:(numbered-node-hop)
+--rw numbered-node-hop
+--rw node-id         te-node-id
+--rw hop-type?       te-hop-type
+--:(numbered-link-hop)
+--rw numbered-link-hop
+--rw link-tp-id       te-tp-id
+--rw hop-type?       te-hop-type
+--rw direction?      te-link-direction
+--:(unnumbered-link-hop)
+--rw unnumbered-link-hop
+--rw link-tp-id       te-tp-id
+--rw node-id         te-node-id
+--rw hop-type?       te-hop-type
+--rw direction?      te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number        inet:as-number
+--rw hop-type?       te-hop-type
+--:(label)
+--rw label-hop
+--rw te-label
+--rw (technology)?
+--:(generic)
+--rw generic?
+--rw rt-types:generalized-label
+--rw direction?
+--rw te-label-direction
+--rw route-object-include-exclude* [index]
+--rw explicit-route-usage?          identityref
+--rw index                          uint32
+--rw (type)?
+--:(numbered-node-hop)
+--rw numbered-node-hop
+--rw node-id         te-node-id
+--rw hop-type?       te-hop-type
+--:(numbered-link-hop)
+--rw numbered-link-hop
+--rw link-tp-id       te-tp-id
+--rw hop-type?       te-hop-type
+--rw direction?      te-link-direction
+--:(unnumbered-link-hop)
+--rw unnumbered-link-hop

```

```

    +---rw link-tp-id      te-tp-id
    +---rw node-id         te-node-id
    +---rw hop-type?       te-hop-type
    +---rw direction?      te-link-direction
+---:(as-number)
    +---rw as-number-hop
    +---rw as-number       inet:as-number
    +---rw hop-type?       te-hop-type
+---:(label)
    +---rw label-hop
    +---rw te-label
    +---rw (technology)?
    |   +---:(generic)
    |       +---rw generic?
    |           rt-types:generalized-label
    +---rw direction?
    |       te-label-direction
+---:(srlg)
    +---rw srlg
    +---rw srlg?          uint32
+---rw path-in-segment!
    +---rw label-restrictions
    +---rw label-restriction* [index]
    +---rw restriction?     enumeration
    +---rw index            uint32
    +---rw label-start
    |   +---rw te-label
    |       +---rw (technology)?
    |           +---:(generic)
    |               +---rw generic?
    |                   rt-types:generalized-label
    +---rw direction?
    |       te-label-direction
+---rw label-end
    +---rw te-label
    +---rw (technology)?
    |   +---:(generic)
    |       +---rw generic?
    |           rt-types:generalized-label
    +---rw direction?
    |       te-label-direction
+---rw label-step
    +---rw (technology)?
    |   +---:(generic)
    |       +---rw generic?      int32
    +---rw range-bitmap?      yang:hex-string
+---rw path-out-segment!
    +---rw label-restrictions

```

```

        +---rw label-restriction* [index]
            +---rw restriction?    enumeration
            +---rw index           uint32
            +---rw label-start
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?
                        te-label-direction
            +---rw label-end
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?
                        te-label-direction
            +---rw label-step
                +---rw (technology)?
                    +---:(generic)
                        +---rw generic?    int32
            +---rw range-bitmap?    yang:hex-string
+---rw tunnels
    +---rw tunnel* [name]
        +---rw name                string
        +---rw alias?              string
        +---rw identifier?         uint32
        +---rw color?              uint32
        +---rw description?        string
        +---rw admin-state?        identityref
        +---ro operational-state?   identityref
        +---rw encoding?           identityref
        +---rw switching-type?     identityref
        +---rw source?             te-types:te-node-id
        +---rw destination?        te-types:te-node-id
        +---rw src-tunnel-tp-id?   binary
        +---rw dst-tunnel-tp-id?   binary
        +---rw bidirectional?      boolean
        +---rw controller
            +---rw protocol-origin? identityref
            +---rw controller-entity-id? string
        +---rw reoptimize-timer?   uint16
        +---rw association-objects
            +---rw association-object* [association-key]
                +---rw association-key    string
                +---rw type?              identityref

```

```

+--rw id?                               uint16
+--rw source
  +--rw id?      te-gen-node-id
  +--rw type?    enumeration
+--rw association-object-extended* [association-key]
  +--rw association-key    string
  +--rw type?              identityref
  +--rw id?                uint16
  +--rw source
    +--rw id?      te-gen-node-id
    +--rw type?    enumeration
  +--rw global-source?    uint32
  +--rw extended-id?      yang:hex-string
+--rw protection
  +--rw enable?                                boolean
  +--rw protection-type?                       identityref
  +--rw protection-reversion-disable?          boolean
  +--rw hold-off-time?                         uint32
  +--rw wait-to-revert?                       uint16
  +--rw aps-signal-id?                        uint8
+--rw restoration
  +--rw enable?                                boolean
  +--rw restoration-type?                     identityref
  +--rw restoration-scheme?                   identityref
  +--rw restoration-reversion-disable?          boolean
  +--rw hold-off-time?                         uint32
  +--rw wait-to-restore?                      uint16
  +--rw wait-to-revert?                      uint16
+--rw te-topology-identifier
  +--rw provider-id?    te-global-id
  +--rw client-id?      te-global-id
  +--rw topology-id?    te-topology-id
+--rw te-bandwidth
  +--rw (technology)?
    +--:(generic)
      +--rw generic?    te-bandwidth
+--rw link-protection?                identityref
+--rw setup-priority?                  uint8
+--rw hold-priority?                   uint8
+--rw signaling-type?                  identityref
+--rw hierarchy
  +--rw dependency-tunnels
    +--rw dependency-tunnel* [name]
      +--rw name
        -> /te/tunnels/tunnel/name
      +--rw encoding?            identityref
      +--rw switching-type?      identityref
+--rw hierarchical-link

```



```

+--rw local-te-node-id?          te-types:te-node-id
+--rw local-te-link-tp-id?       te-types:te-tp-id
+--rw remote-te-node-id?        te-types:te-node-id
+--rw te-topology-identifier
  +--rw provider-id?            te-global-id
  +--rw client-id?              te-global-id
  +--rw topology-id?            te-topology-id
+--rw primary-paths
  +--rw primary-path* [name]
    +--rw name                    string
    +--rw path-computation-method? identityref
    +--rw path-computation-server
      +--rw id?                  te-gen-node-id
      +--rw type?                enumeration
    +--rw compute-only?          empty
    +--rw use-path-computation?  boolean
    +--rw lockdown?              empty
    +--rw path-scope?            identityref
    +--rw preference?            uint8
    +--rw k-requested-paths?     uint8
    +--rw association-objects
      +--rw association-object* [association-key]
        +--rw association-key    string
        +--rw type?              identityref
        +--rw id?                uint16
        +--rw source
          +--rw id?              te-gen-node-id
          +--rw type?            enumeration
      +--rw association-object-extended*
        [association-key]
        +--rw association-key    string
        +--rw type?              identityref
        +--rw id?                uint16
        +--rw source
          +--rw id?              te-gen-node-id
          +--rw type?            enumeration
        +--rw global-source?     uint32
        +--rw extended-id?      yang:hex-string
+--rw optimizations
  +--rw (algorithm)?
    +--:(metric) {path-optimization-metric}?
      +--rw optimization-metric* [metric-type]
        +--rw metric-type
          identityref
        +--rw weight?
          uint8
        +--rw explicit-route-exclude-objects
        +--rw route-object-exclude-object*

```

```

[index]
+--rw index
|   uint32
+--rw (type)?
+--:(numbered-node-hop)
|   +--rw numbered-node-hop
|       +--rw node-id
|           |   te-node-id
|       +--rw hop-type?
|           |   te-hop-type
+--:(numbered-link-hop)
|   +--rw numbered-link-hop
|       +--rw link-tp-id
|           |   te-tp-id
|       +--rw hop-type?
|           |   te-hop-type
|       +--rw direction?
|           |   te-link-direction
+--:(unnumbered-link-hop)
|   +--rw unnumbered-link-hop
|       +--rw link-tp-id
|           |   te-tp-id
|       +--rw node-id
|           |   te-node-id
|       +--rw hop-type?
|           |   te-hop-type
|       +--rw direction?
|           |   te-link-direction
+--:(as-number)
|   +--rw as-number-hop
|       +--rw as-number
|           |   inet:as-number
|       +--rw hop-type?
|           |   te-hop-type
+--:(label)
|   +--rw label-hop
|       +--rw te-label
|           +--rw (technology)?
|               |   +--:(generic)
|                   |   +--rw generic?
|                       |   rt-types:ge
neralized-label
|
|   +--rw direction?
|       |   te-label-directio
n
+--:(srlg)
|   +--rw srlg
|       +--rw srlg?   uint32

```

```

+---rw explicit-route-include-objects
+---rw route-object-include-object*
    [index]
+---rw index
    |   uint32
+---rw (type)?
+---:(numbered-node-hop)
    +---rw numbered-node-hop
        +---rw node-id
            |   te-node-id
        +---rw hop-type?
            |   te-hop-type
+---:(numbered-link-hop)
    +---rw numbered-link-hop
        +---rw link-tp-id
            |   te-tp-id
        +---rw hop-type?
            |   te-hop-type
        +---rw direction?
            |   te-link-direction
+---:(unnumbered-link-hop)
    +---rw unnumbered-link-hop
        +---rw link-tp-id
            |   te-tp-id
        +---rw node-id
            |   te-node-id
        +---rw hop-type?
            |   te-hop-type
        +---rw direction?
            |   te-link-direction
+---:(as-number)
    +---rw as-number-hop
        +---rw as-number
            |   inet:as-number
        +---rw hop-type?
            |   te-hop-type
+---:(label)
    +---rw label-hop
        +---rw te-label
            +---rw (technology)?
                +---:(generic)
                    +---rw generic?
                        |   rt-types:ge
neralized-label
n
+---rw tiebreakers

```

```

        +---rw tiebreaker* [tiebreaker-type]
        +---rw tiebreaker-type identityref
    +---:(objective-function)
        {path-optimization-objective-function}?
        +---rw objective-function
        +---rw objective-function-type?
            identityref
+---rw named-path-constraint? leafref
    {te-types:named-path-constraints}?
+---rw te-bandwidth
    +---rw (technology)?
    +---:(generic)
        +---rw generic? te-bandwidth
+---rw link-protection? identityref
+---rw setup-priority? uint8
+---rw hold-priority? uint8
+---rw signaling-type? identityref
+---rw path-metric-bounds
    +---rw path-metric-bound* [metric-type]
    +---rw metric-type identityref
    +---rw upper-bound? uint64
+---rw path-affinities-values
    +---rw path-affinities-value* [usage]
    +---rw usage identityref
    +---rw value? admin-groups
+---rw path-affinity-names
    +---rw path-affinity-name* [usage]
    +---rw usage identityref
    +---rw affinity-name* [name]
    +---rw name string
+---rw path-srlgs-lists
    +---rw path-srlgs-list* [usage]
    +---rw usage identityref
    +---rw values* srlg
+---rw path-srlgs-names
    +---rw path-srlgs-name* [usage]
    +---rw usage identityref
    +---rw names* string
+---rw disjointness?
    te-path-disjointness
+---rw explicit-route-objects-always
    +---rw route-object-exclude-always* [index]
    +---rw index uint32
    +---rw (type)?
    +---:(numbered-node-hop)
        +---rw numbered-node-hop
        +---rw node-id te-node-id
        +---rw hop-type? te-hop-type

```

bel

```

+---:(numbered-link-hop)
|   +---rw numbered-link-hop
|       +---rw link-tp-id      te-tp-id
|       +---rw hop-type?      te-hop-type
|       +---rw direction?     te-link-direction
+---:(unnumbered-link-hop)
|   +---rw unnumbered-link-hop
|       +---rw link-tp-id      te-tp-id
|       +---rw node-id        te-node-id
|       +---rw hop-type?      te-hop-type
|       +---rw direction?     te-link-direction
+---:(as-number)
|   +---rw as-number-hop
|       +---rw as-number      inet:as-number
|       +---rw hop-type?      te-hop-type
+---:(label)
|   +---rw label-hop
|       +---rw te-label
|           +---rw (technology)?
|               +---:(generic)
|                   +---rw generic?
|                       rt-types:generalized-la
|
|       +---rw direction?
|           te-label-direction
+---rw route-object-include-exclude* [index]
+---rw explicit-route-usage?      identityref
+---rw index                      uint32
+---rw (type)?
+---:(numbered-node-hop)
|   +---rw numbered-node-hop
|       +---rw node-id      te-node-id
|       +---rw hop-type?    te-hop-type
+---:(numbered-link-hop)
|   +---rw numbered-link-hop
|       +---rw link-tp-id    te-tp-id
|       +---rw hop-type?    te-hop-type
|       +---rw direction?   te-link-direction
+---:(unnumbered-link-hop)
|   +---rw unnumbered-link-hop
|       +---rw link-tp-id    te-tp-id
|       +---rw node-id      te-node-id
|       +---rw hop-type?    te-hop-type
|       +---rw direction?   te-link-direction
+---:(as-number)
|   +---rw as-number-hop
|       +---rw as-number    inet:as-number
|       +---rw hop-type?    te-hop-type

```

bel

```

+---:(label)
|   +---rw label-hop
|       +---rw te-label
|           +---rw (technology)?
|               +---:(generic)
|                   +---rw generic?
|                       rt-types:generalized-la
|
|       +---rw direction?
|           te-label-direction
+---:(srlg)
|   +---rw srlg
|       +---rw srlg?    uint32
+---rw path-in-segment!
+---rw label-restrictions
+---rw label-restriction* [index]
+---rw restriction?    enumeration
+---rw index            uint32
+---rw label-start
|   +---rw te-label
|       +---rw (technology)?
|           +---:(generic)
|               +---rw generic?
|                   rt-types:generalized-label
|       +---rw direction?
|           te-label-direction
+---rw label-end
|   +---rw te-label
|       +---rw (technology)?
|           +---:(generic)
|               +---rw generic?
|                   rt-types:generalized-label
|       +---rw direction?
|           te-label-direction
+---rw label-step
|   +---rw (technology)?
|       +---:(generic)
|           +---rw generic?    int32
+---rw range-bitmap?    yang:hex-string
+---rw path-out-segment!
+---rw label-restrictions
+---rw label-restriction* [index]
+---rw restriction?    enumeration
+---rw index            uint32
+---rw label-start
|   +---rw te-label
|       +---rw (technology)?
|           +---:(generic)

```

```

|         |        +---rw generic?
|           rt-types:generalized-label
+---rw direction?
|       te-label-direction
+---rw label-end
|   +---rw te-label
|       +---rw (technology)?
|           +---:(generic)
|               +---rw generic?
|                   rt-types:generalized-label
+---rw direction?
|       te-label-direction
+---rw label-step
|   +---rw (technology)?
|       +---:(generic)
|           +---rw generic?      int32
+---rw range-bitmap?    yang:hex-string
+---ro computed-paths-properties
+---ro computed-path-properties* [k-index]
+---ro k-index          uint8
+---ro path-properties
+---ro path-metric* [metric-type]
|     +---ro metric-type      identityref
|     +---ro accumulative-value?  uint64
+---ro path-affinities-values
|     +---ro path-affinities-value* [usage]
|         +---ro usage      identityref
|         +---ro value?    admin-groups
+---ro path-affinity-names
|     +---ro path-affinity-name* [usage]
|         +---ro usage      identityref
|         +---ro affinity-name* [name]
|             +---ro name    string
+---ro path-srlgs-lists
|     +---ro path-srlgs-list* [usage]
|         +---ro usage      identityref
|         +---ro values*    srlg
+---ro path-srlgs-names
|     +---ro path-srlgs-name* [usage]
|         +---ro usage      identityref
|         +---ro names*    string
+---ro path-route-objects
|     +---ro path-route-object* [index]
|         +---ro index
|             |      uint32
+---ro (type)?
|     +---:(numbered-node-hop)
|         +---ro numbered-node-hop

```

```

+--ro node-id      te-node-id
+--ro hop-type?
    te-hop-type
+--:(numbered-link-hop)
+--ro numbered-link-hop
+--ro link-tp-id    te-tp-id
+--ro hop-type?
    |
    te-hop-type
+--ro direction?
    te-link-direction
+--:(unnumbered-link-hop)
+--ro unnumbered-link-hop
+--ro link-tp-id    te-tp-id
+--ro node-id
    |
    te-node-id
+--ro hop-type?
    |
    te-hop-type
+--ro direction?
    te-link-direction
+--:(as-number)
+--ro as-number-hop
+--ro as-number
    |
    inet:as-number
+--ro hop-type?
    te-hop-type
+--:(label)
+--ro label-hop
+--ro te-label
    +--ro (technology)?
        |
        +--:(generic)
        |
        +--ro generic?
            rt-types:gener
+--ro direction?
    te-label-direction
+--ro te-bandwidth
+--ro (technology)?
    +--:(generic)
    +--ro generic?
        te-bandwidth
+--ro disjointness-type?
    te-types:te-path-disjointness
+--ro computed-path-error-infos
+--ro computed-path-error-info* []
+--ro error-description?
    string
+--ro error-timestamp?
    yang:date-and-time
+--ro error-reason?
    identityref
+--ro lsp-provisioning-error-infos
+--ro lsp-provisioning-error-info* []

```



```

+--ro error-description?  string
+--ro error-timestamp?    yang:date-and-time
+--ro error-node-id?      te-types:te-node-id
+--ro error-link-id?      te-types:te-tp-id
+--ro lsp-id?             uint16
+--ro lsps
  +--ro lsp* [node lsp-id]
    +--ro tunnel-name?
      |      -> /te/lsps/lsp/tunnel-name
    +--ro node      -> /te/lsps/lsp/node
    +--ro lsp-id    -> /te/lsps/lsp/lsp-id
+--rw primary-reverse-path
  +--rw name?                                string
  +--rw path-computation-method?
    |      identityref
  +--rw path-computation-server
    +--rw id?      te-gen-node-id
    +--rw type?    enumeration
  +--rw compute-only?                          empty
  +--rw use-path-computation?
    |      boolean
  +--rw lockdown?                              empty
  +--ro path-scope?
    |      identityref
  +--rw association-objects
    +--rw association-object* [association-key]
      +--rw association-key  string
      +--rw type?            identityref
      +--rw id?              uint16
      +--rw source
        +--rw id?      te-gen-node-id
        +--rw type?    enumeration
    +--rw association-object-extended*
      [association-key]
      +--rw association-key  string
      +--rw type?            identityref
      +--rw id?              uint16
      +--rw source
        +--rw id?      te-gen-node-id
        +--rw type?    enumeration
      +--rw global-source?   uint32
      +--rw extended-id?     yang:hex-string
  +--rw optimizations
    +--rw (algorithm)?
      +--:(metric) {path-optimization-metric}?
        +--rw optimization-metric* [metric-type]
          +--rw metric-type
            |      identityref

```

```

+---rw weight?
|   uint8
+---rw explicit-route-exclude-objects
|   +---rw route-object-exclude-object*
|       [index]
|       +---rw index
|           |   uint32
|       +---rw (type)?
|           +---:(numbered-node-hop)
|               +---rw numbered-node-hop
|                   +---rw node-id
|                       |   te-node-id
|                   +---rw hop-type?
|                       |   te-hop-type
|           +---:(numbered-link-hop)
|               +---rw numbered-link-hop
|                   +---rw link-tp-id
|                       |   te-tp-id
|                   +---rw hop-type?
|                       |   te-hop-type
|                   +---rw direction?
|                       |   te-link-direction
|           +---:(unnumbered-link-hop)
|               +---rw unnumbered-link-hop
|                   +---rw link-tp-id
|                       |   te-tp-id
|                   +---rw node-id
|                       |   te-node-id
|                   +---rw hop-type?
|                       |   te-hop-type
|                   +---rw direction?
|                       |   te-link-direction
|           +---:(as-number)
|               +---rw as-number-hop
|                   +---rw as-number
|                       |   inet:as-number
|                   +---rw hop-type?
|                       |   te-hop-type
|           +---:(label)
|               +---rw label-hop
|                   +---rw te-label
|                       +---rw (technology)?
|                           +---:(generic)
|                               +---rw generic?
|                                   rt-types
:generalized-label
+---rw direction?
    te-label-direc

```

tion

```

+---:(srlg)
+---rw srlg
+---rw srlg?   uint32
+---rw explicit-route-include-objects
+---rw route-object-include-object*
+---[index]
+---rw index
+---|   uint32
+---rw (type)?
+---:(numbered-node-hop)
+---|   +---rw numbered-node-hop
+---|   |   +---rw node-id
+---|   |   |   te-node-id
+---|   |   +---rw hop-type?
+---|   |   |   te-hop-type
+---:(numbered-link-hop)
+---|   +---rw numbered-link-hop
+---|   |   +---rw link-tp-id
+---|   |   |   te-tp-id
+---|   |   +---rw hop-type?
+---|   |   |   te-hop-type
+---|   |   +---rw direction?
+---|   |   |   te-link-direction
+---:(unnumbered-link-hop)
+---|   +---rw unnumbered-link-hop
+---|   |   +---rw link-tp-id
+---|   |   |   te-tp-id
+---|   |   +---rw node-id
+---|   |   |   te-node-id
+---|   |   +---rw hop-type?
+---|   |   |   te-hop-type
+---|   |   +---rw direction?
+---|   |   |   te-link-direction
+---:(as-number)
+---|   +---rw as-number-hop
+---|   |   +---rw as-number
+---|   |   |   inet:as-number
+---|   |   +---rw hop-type?
+---|   |   |   te-hop-type
+---:(label)
+---|   +---rw label-hop
+---|   |   +---rw te-label
+---|   |   |   +---rw (technology)?
+---|   |   |   |   +---:(generic)
+---|   |   |   |   |   +---rw generic?
+---|   |   |   |   |   |   rt-types
:generalized-label

```

```

        +---rw direction?
            te-label-direct
tion
        +---rw tiebreakers
            +---rw tiebreaker* [tiebreaker-type]
                +---rw tiebreaker-type
                    identityref
        +---:(objective-function)
            {path-optimization-objective-function
}?
        +---rw objective-function
            +---rw objective-function-type?
                identityref
        +---rw named-path-constraint? leafref
            {te-types:named-path-constraints}?
        +---rw te-bandwidth
            +---rw (technology)?
                +---:(generic)
                    +---rw generic? te-bandwidth
        +---rw link-protection?
            identityref
        +---rw setup-priority? uint8
        +---rw hold-priority? uint8
        +---rw signaling-type?
            identityref
        +---rw path-metric-bounds
            +---rw path-metric-bound* [metric-type]
                +---rw metric-type identityref
                +---rw upper-bound? uint64
        +---rw path-affinities-values
            +---rw path-affinities-value* [usage]
                +---rw usage identityref
                +---rw value? admin-groups
        +---rw path-affinity-names
            +---rw path-affinity-name* [usage]
                +---rw usage identityref
                +---rw affinity-name* [name]
                    +---rw name string
        +---rw path-srlgs-lists
            +---rw path-srlgs-list* [usage]
                +---rw usage identityref
                +---rw values* srlg
        +---rw path-srlgs-names
            +---rw path-srlgs-name* [usage]
                +---rw usage identityref
                +---rw names* string
        +---rw disjointness?
            te-path-disjointness

```

```

+---rw explicit-route-objects-always
+---rw route-object-exclude-always* [index]
+---rw index                               uint32
+---rw (type)?
+---:(numbered-node-hop)
+---rw numbered-node-hop
+---rw node-id         te-node-id
+---rw hop-type?       te-hop-type
+---:(numbered-link-hop)
+---rw numbered-link-hop
+---rw link-tp-id       te-tp-id
+---rw hop-type?       te-hop-type
+---rw direction?
+---rw                    te-link-direction
+---:(unnumbered-link-hop)
+---rw unnumbered-link-hop
+---rw link-tp-id       te-tp-id
+---rw node-id         te-node-id
+---rw hop-type?       te-hop-type
+---rw direction?
+---rw                    te-link-direction
+---:(as-number)
+---rw as-number-hop
+---rw as-number       inet:as-number
+---rw hop-type?       te-hop-type
+---:(label)
+---rw label-hop
+---rw te-label
+---rw (technology)?
+---:(generic)
+---rw generic?
+---rw                    rt-types:generalized
- label
+---rw direction?
+---rw                    te-label-direction
+---rw route-object-include-exclude* [index]
+---rw explicit-route-usage?
+---rw                    identityref
+---rw index                               uint32
+---rw (type)?
+---:(numbered-node-hop)
+---rw numbered-node-hop
+---rw node-id         te-node-id
+---rw hop-type?       te-hop-type
+---:(numbered-link-hop)
+---rw numbered-link-hop
+---rw link-tp-id       te-tp-id
+---rw hop-type?       te-hop-type

```

					<pre> +---rw direction?     te-link-direction +---:(unnumbered-link-hop) +---rw unnumbered-link-hop +---rw link-tp-id      te-tp-id +---rw node-id        te-node-id +---rw hop-type?      te-hop-type +---rw direction?     te-link-direction +---:(as-number) +---rw as-number-hop +---rw as-number      inet:as-number +---rw hop-type?      te-hop-type +---:(label) +---rw label-hop +---rw te-label +---rw (technology)?     +---:(generic)     +---rw generic?         rt-types:generalized </pre>
-label					
					<pre> +---rw direction?     te-label-direction +---:(srlg) +---rw srlg +---rw srlg?      uint32 +---rw path-in-segment! +---rw label-restrictions +---rw label-restriction* [index] +---rw restriction?      enumeration +---rw index              uint32 +---rw label-start +---rw te-label +---rw (technology)?     +---:(generic)     +---rw generic?         rt-types:generalized-la </pre>
bel					
					<pre> +---rw direction?     te-label-direction +---rw label-end +---rw te-label +---rw (technology)?     +---:(generic)     +---rw generic?         rt-types:generalized-la </pre>
bel					
					<pre> +---rw direction? </pre>

					<pre>                                 te-label-direction +---rw label-step     +---rw (technology)?         +---:(generic)             +---rw generic?    int32 +---rw range-bitmap?    yang:hex-string +---rw path-out-segment!     +---rw label-restrictions         +---rw label-restriction* [index]             +---rw restriction?    enumeration             +---rw index            uint32 +---rw label-start     +---rw te-label         +---rw (technology)?             +---:(generic)                 +---rw generic?                     rt-types:generalized-la </pre>
bel					<pre>                                 +---rw direction?                                 te-label-direction +---rw label-end     +---rw te-label         +---rw (technology)?             +---:(generic)                 +---rw generic?                     rt-types:generalized-la </pre>
bel					<pre>                                 +---rw direction?                                 te-label-direction +---rw label-step     +---rw (technology)?         +---:(generic)             +---rw generic?    int32 +---rw range-bitmap?    yang:hex-string +---ro computed-paths-properties     +---ro computed-path-properties* [k-index]         +---ro k-index            uint8 +---ro path-properties     +---ro path-metric* [metric-type]         +---ro metric-type             identityref         +---ro accumulative-value?    uint64 +---ro path-affinities-values     +---ro path-affinities-value* [usage]         +---ro usage            identityref         +---ro value?    admin-groups +---ro path-affinity-names     +---ro path-affinity-name* [usage] </pre>

```

+---ro usage          identityref
+---ro affinity-name* [name]
    +---ro name        string
+---ro path-srlgs-lists
    +---ro path-srlgs-list* [usage]
        +---ro usage    identityref
        +---ro values*   srlg
+---ro path-srlgs-names
    +---ro path-srlgs-name* [usage]
        +---ro usage    identityref
        +---ro names*   string
+---ro path-route-objects
    +---ro path-route-object* [index]
        +---ro index
            |
            uint32
        +---ro (type)?
            +---:(numbered-node-hop)
                +---ro numbered-node-hop
                    +---ro node-id
                        |
                        te-node-id
                    +---ro hop-type?
                        |
                        te-hop-type
            +---:(numbered-link-hop)
                +---ro numbered-link-hop
                    +---ro link-tp-id
                        |
                        te-tp-id
                    +---ro hop-type?
                        |
                        te-hop-type
                    +---ro direction?
                        |
                        te-link-direction
            +---:(unnumbered-link-hop)
                +---ro unnumbered-link-hop
                    +---ro link-tp-id
                        |
                        te-tp-id
                    +---ro node-id
                        |
                        te-node-id
                    +---ro hop-type?
                        |
                        te-hop-type
                    +---ro direction?
                        |
                        te-link-direction
            +---:(as-number)
                +---ro as-number-hop
                    +---ro as-number
                        |
                        inet:as-number
                    +---ro hop-type?
                        |
                        te-hop-type
            +---:(label)
                +---ro label-hop

```



```

+---ro te-label
+---ro (technology)?
|   +---:(generic)
|       +---ro generic?
|           rt-types:ge
neralized-label
+---ro direction?
te-label-directio
n
+---ro te-bandwidth
|   +---ro (technology)?
|       +---:(generic)
|           +---ro generic?   te-bandwidth
+---ro disjointness-type?
te-types:te-path-disjointness
+---ro computed-path-error-infos
+---ro computed-path-error-info* []
+---ro error-description?   string
+---ro error-timestamp?
|   yang:date-and-time
+---ro error-reason?       identityref
+---ro lsp-provisioning-error-infos
+---ro lsp-provisioning-error-info* []
+---ro error-description?   string
+---ro error-timestamp?
|   yang:date-and-time
+---ro error-node-id?
|   te-types:te-node-id
+---ro error-link-id?
|   te-types:te-tp-id
+---ro lsp-id?              uint16
+---ro lsps
|   +---ro lsp* [node lsp-id]
|       +---ro tunnel-name?
|           -> /te/lsps/lsp/tunnel-name
|       +---ro node          -> /te/lsps/lsp/node
|       +---ro lsp-id        -> /te/lsps/lsp/lsp-id
+---rw candidate-secondary-reverse-paths
+---rw candidate-secondary-reverse-path*
|   [secondary-path]
+---rw secondary-path       leafref
+---rw candidate-secondary-paths
+---rw candidate-secondary-path* [secondary-path]
+---rw secondary-path       leafref
+---ro active?              boolean
+---rw secondary-paths
+---rw secondary-path* [name]
+---rw name                  string

```

```

+--rw path-computation-method?          identityref
+--rw path-computation-server
|   +--rw id?      te-gen-node-id
|   +--rw type?    enumeration
+--rw compute-only?                      empty
+--rw use-path-computation?              boolean
+--rw lockdown?                          empty
+--ro path-scope?                        identityref
+--rw preference?                        uint8
+--rw association-objects
|   +--rw association-object* [association-key]
|   |   +--rw association-key    string
|   |   +--rw type?             identityref
|   |   +--rw id?               uint16
|   |   +--rw source
|   |   |   +--rw id?      te-gen-node-id
|   |   |   +--rw type?    enumeration
|   +--rw association-object-extended*
|   |   [association-key]
|   |   +--rw association-key    string
|   |   +--rw type?             identityref
|   |   +--rw id?               uint16
|   |   +--rw source
|   |   |   +--rw id?      te-gen-node-id
|   |   |   +--rw type?    enumeration
|   +--rw global-source?          uint32
|   +--rw extended-id?            yang:hex-string
+--rw optimizations
|   +--rw (algorithm)?
|   |   +--:(metric) {path-optimization-metric}?
|   |   |   +--rw optimization-metric* [metric-type]
|   |   |   |   +--rw metric-type
|   |   |   |   |   identityref
|   |   |   |   +--rw weight?
|   |   |   |   |   uint8
|   |   |   +--rw explicit-route-exclude-objects
|   |   |   |   +--rw route-object-exclude-object*
|   |   |   |   |   [index]
|   |   |   |   |   +--rw index
|   |   |   |   |   |   uint32
|   |   |   |   +--rw (type)?
|   |   |   |   |   +--:(numbered-node-hop)
|   |   |   |   |   |   +--rw numbered-node-hop
|   |   |   |   |   |   |   +--rw node-id
|   |   |   |   |   |   |   |   te-node-id
|   |   |   |   |   |   |   +--rw hop-type?
|   |   |   |   |   |   |   |   te-hop-type
|   |   |   |   +--:(numbered-link-hop)

```

```

+--rw numbered-link-hop
+--rw link-tp-id
|   te-tp-id
+--rw hop-type?
|   te-hop-type
+--rw direction?
|   te-link-direction
+--:(unnumbered-link-hop)
+--rw unnumbered-link-hop
+--rw link-tp-id
|   te-tp-id
+--rw node-id
|   te-node-id
+--rw hop-type?
|   te-hop-type
+--rw direction?
|   te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number
|   inet:as-number
+--rw hop-type?
|   te-hop-type
+--:(label)
+--rw label-hop
+--rw te-label
|   +--rw (technology)?
|   |   +--:(generic)
|   |   +--rw generic?
|   |   rt-types:ge
+--rw direction?
|   te-label-direction
+--:(srlg)
+--rw srlg
|   +--rw srlg?   uint32
+--rw explicit-route-include-objects
+--rw route-object-include-object*
|   [index]
+--rw index
|   uint32
+--rw (type)?
+--:(numbered-node-hop)
+--rw numbered-node-hop
|   +--rw node-id
|   |   te-node-id
+--rw hop-type?

```

						te-hop-type
					+	---:(numbered-link-hop)
						+---rw numbered-link-hop
						+---rw link-tp-id
						te-tp-id
						+---rw hop-type?
						te-hop-type
						+---rw direction?
						te-link-direction
					+	---:(unnumbered-link-hop)
						+---rw unnumbered-link-hop
						+---rw link-tp-id
						te-tp-id
						+---rw node-id
						te-node-id
						+---rw hop-type?
						te-hop-type
						+---rw direction?
						te-link-direction
					+	---:(as-number)
						+---rw as-number-hop
						+---rw as-number
						inet:as-number
						+---rw hop-type?
						te-hop-type
					+	---:(label)
						+---rw label-hop
						+---rw te-label
						+---rw (technology)?
						+---:(generic)
						+---rw generic?
						rt-types:ge
neralized-label						
						+---rw direction?
						te-label-directio
n						
						+---rw tiebreakers
						+---rw tiebreaker* [tiebreaker-type]
						+---rw tiebreaker-type identityref
					+	---:(objective-function)
						{path-optimization-objective-function}?
						+---rw objective-function
						+---rw objective-function-type?
						identityref
					+	---rw named-path-constraint? leafref
						{te-types:named-path-constraints}?
					+	---rw te-bandwidth
						+---rw (technology)?

```

+---:(generic)
+---rw generic?      te-bandwidth
+---rw link-protection?      identityref
+---rw setup-priority?      uint8
+---rw hold-priority?      uint8
+---rw signaling-type?      identityref
+---rw path-metric-bounds
+---rw path-metric-bound* [metric-type]
+---rw metric-type      identityref
+---rw upper-bound?      uint64
+---rw path-affinities-values
+---rw path-affinities-value* [usage]
+---rw usage      identityref
+---rw value?      admin-groups
+---rw path-affinity-names
+---rw path-affinity-name* [usage]
+---rw usage      identityref
+---rw affinity-name* [name]
+---rw name      string
+---rw path-srlgs-lists
+---rw path-srlgs-list* [usage]
+---rw usage      identityref
+---rw values*      srlg
+---rw path-srlgs-names
+---rw path-srlgs-name* [usage]
+---rw usage      identityref
+---rw names*      string
+---rw disjointness?
+---rw te-path-disjointness
+---rw explicit-route-objects-always
+---rw route-object-exclude-always* [index]
+---rw index      uint32
+---rw (type)?
+---:(numbered-node-hop)
+---rw numbered-node-hop
+---rw node-id      te-node-id
+---rw hop-type?      te-hop-type
+---:(numbered-link-hop)
+---rw numbered-link-hop
+---rw link-tp-id      te-tp-id
+---rw hop-type?      te-hop-type
+---rw direction?      te-link-direction
+---:(unnumbered-link-hop)
+---rw unnumbered-link-hop
+---rw link-tp-id      te-tp-id
+---rw node-id      te-node-id
+---rw hop-type?      te-hop-type
+---rw direction?      te-link-direction

```

				<pre> +--:(as-number)     +--rw as-number-hop         +--rw as-number      inet:as-number         +--rw hop-type?      te-hop-type +--:(label)     +--rw label-hop         +--rw te-label             +--rw (technology)?                 +--:(generic)                     +--rw generic?                         rt-types:generalized-label </pre>
bel				<pre> +--rw direction?     te-label-direction +--rw route-object-include-exclude* [index] +--rw explicit-route-usage?      identityref +--rw index                      uint32 +--rw (type)? +--:(numbered-node-hop)     +--rw numbered-node-hop         +--rw node-id      te-node-id         +--rw hop-type?    te-hop-type +--:(numbered-link-hop)     +--rw numbered-link-hop         +--rw link-tp-id    te-tp-id         +--rw hop-type?    te-hop-type         +--rw direction?   te-link-direction +--:(unnumbered-link-hop)     +--rw unnumbered-link-hop         +--rw link-tp-id    te-tp-id         +--rw node-id      te-node-id         +--rw hop-type?    te-hop-type         +--rw direction?   te-link-direction +--:(as-number)     +--rw as-number-hop         +--rw as-number      inet:as-number         +--rw hop-type?      te-hop-type +--:(label)     +--rw label-hop         +--rw te-label             +--rw (technology)?                 +--:(generic)                     +--rw generic?                         rt-types:generalized-label </pre>
bel				<pre> +--rw direction?     te-label-direction +--:(srlg) </pre>

```

        +---rw srlg
            +---rw srlg?   uint32
+---rw path-in-segment!
    +---rw label-restrictions
        +---rw label-restriction* [index]
            +---rw restriction?   enumeration
            +---rw index          uint32
            +---rw label-start
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?
                        te-label-direction
            +---rw label-end
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?
                        te-label-direction
            +---rw label-step
                +---rw (technology)?
                    +---:(generic)
                        +---rw generic?   int32
            +---rw range-bitmap?   yang:hex-string
+---rw path-out-segment!
    +---rw label-restrictions
        +---rw label-restriction* [index]
            +---rw restriction?   enumeration
            +---rw index          uint32
            +---rw label-start
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?
                        te-label-direction
            +---rw label-end
                +---rw te-label
                    +---rw (technology)?
                        +---:(generic)
                            +---rw generic?
                                rt-types:generalized-label
                    +---rw direction?

```

```

|                                     te-label-direction
|                                     +---rw label-step
|                                     |   +---rw (technology)?
|                                     |   +---:(generic)
|                                     |   +---rw generic?    int32
|                                     +---rw range-bitmap?   yang:hex-string
+---rw protection
|   +---rw enable?                                boolean
|   +---rw protection-type?                        identityref
|   +---rw protection-reversion-disable?          boolean
|   +---rw hold-off-time?                          uint32
|   +---rw wait-to-revert?                         uint16
|   +---rw aps-signal-id?                          uint8
+---rw restoration
|   +---rw enable?                                boolean
|   +---rw restoration-type?
|   |   identityref
|   +---rw restoration-scheme?
|   |   identityref
|   +---rw restoration-reversion-disable?          boolean
|   +---rw hold-off-time?                          uint32
|   +---rw wait-to-restore?                        uint16
|   +---rw wait-to-revert?                         uint16
+---ro computed-paths-properties
|   +---ro computed-path-properties* [k-index]
|   |   +---ro k-index                          uint8
|   |   +---ro path-properties
|   |   |   +---ro path-metric* [metric-type]
|   |   |   |   +---ro metric-type              identityref
|   |   |   |   +---ro accumulative-value?      uint64
|   |   |   +---ro path-affinities-values
|   |   |   |   +---ro path-affinities-value* [usage]
|   |   |   |   |   +---ro usage                identityref
|   |   |   |   |   +---ro value?              admin-groups
|   |   |   +---ro path-affinity-names
|   |   |   |   +---ro path-affinity-name* [usage]
|   |   |   |   |   +---ro usage                identityref
|   |   |   |   |   +---ro affinity-name* [name]
|   |   |   |   |   +---ro name                string
|   |   +---ro path-srlgs-lists
|   |   |   +---ro path-srlgs-list* [usage]
|   |   |   |   +---ro usage                identityref
|   |   |   |   +---ro values*              srlg
|   |   +---ro path-srlgs-names
|   |   |   +---ro path-srlgs-name* [usage]
|   |   |   |   +---ro usage                identityref
|   |   |   |   +---ro names*              string
|   +---ro path-route-objects

```



				+--ro path-route-object* [index]
				+--ro index
				uint32
				+--ro (type)?
				+--:(numbered-node-hop)
				+--ro numbered-node-hop
				+--ro node-id      te-node-id
				+--ro hop-type?
				te-hop-type
				+--:(numbered-link-hop)
				+--ro numbered-link-hop
				+--ro link-tp-id   te-tp-id
				+--ro hop-type?
				te-hop-type
				+--ro direction?
				te-link-direction
				+--:(unnumbered-link-hop)
				+--ro unnumbered-link-hop
				+--ro link-tp-id   te-tp-id
				+--ro node-id
				te-node-id
				+--ro hop-type?
				te-hop-type
				+--ro direction?
				te-link-direction
				+--:(as-number)
				+--ro as-number-hop
				+--ro as-number
				inet:as-number
				+--ro hop-type?
				te-hop-type
				+--:(label)
				+--ro label-hop
				+--ro te-label
				+--ro (technology)?
				+--:(generic)
				+--ro generic?
				rt-types:gener
				+--ro direction?
				te-label-direction
				+--ro te-bandwidth
				+--ro (technology)?
				+--:(generic)
				+--ro generic?   te-bandwidth
				+--ro disjointness-type?
				te-types:te-path-disjointness
				+--ro computed-path-error-infos

alized-label

```

    +---ro computed-path-error-info* []
      +---ro error-description?   string
      +---ro error-timestamp?     yang:date-and-time
      +---ro error-reason?        identityref
+---ro lsp-provisioning-error-infos
  +---ro lsp-provisioning-error-info* []
    +---ro error-description?   string
    +---ro error-timestamp?     yang:date-and-time
    +---ro error-node-id?       te-types:te-node-id
    +---ro error-link-id?       te-types:te-tp-id
    +---ro lsp-id?              uint16
+---ro lsps
  +---ro lsp* [node lsp-id]
    +---ro tunnel-name?
      |         -> /te/lsps/lsp/tunnel-name
    +---ro node         -> /te/lsps/lsp/node
    +---ro lsp-id        -> /te/lsps/lsp/lsp-id
+---rw secondary-reverse-paths
  +---rw secondary-reverse-path* [name]
    +---rw name                      string
    +---rw path-computation-method?  identityref
    +---rw path-computation-server
      | +---rw id?      te-gen-node-id
      | +---rw type?    enumeration
    +---rw compute-only?              empty
    +---rw use-path-computation?      boolean
    +---rw lockdown?                  empty
    +---ro path-scope?                identityref
    +---rw preference?                uint8
    +---rw association-objects
      +---rw association-object* [association-key]
        +---rw association-key      string
        +---rw type?                identityref
        +---rw id?                  uint16
        +---rw source
          | +---rw id?      te-gen-node-id
          | +---rw type?    enumeration
      +---rw association-object-extended*
        [association-key]
        +---rw association-key      string
        +---rw type?                identityref
        +---rw id?                  uint16
        +---rw source
          | +---rw id?      te-gen-node-id
          | +---rw type?    enumeration
        +---rw global-source?      uint32
        +---rw extended-id?        yang:hex-string
+---rw optimizations

```

```

+--rw (algorithm)?
+--:(metric) {path-optimization-metric}?
+--rw optimization-metric* [metric-type]
+--rw metric-type
|   identityref
+--rw weight?
|   uint8
+--rw explicit-route-exclude-objects
+--rw route-object-exclude-object*
|   [index]
+--rw index
|   uint32
+--rw (type)?
+--:(numbered-node-hop)
+--rw numbered-node-hop
+--rw node-id
|   te-node-id
+--rw hop-type?
|   te-hop-type
+--:(numbered-link-hop)
+--rw numbered-link-hop
+--rw link-tp-id
|   te-tp-id
+--rw hop-type?
|   te-hop-type
+--rw direction?
|   te-link-direction
+--:(unnumbered-link-hop)
+--rw unnumbered-link-hop
+--rw link-tp-id
|   te-tp-id
+--rw node-id
|   te-node-id
+--rw hop-type?
|   te-hop-type
+--rw direction?
|   te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number
|   inet:as-number
+--rw hop-type?
|   te-hop-type
+--:(label)
+--rw label-hop
+--rw te-label
+--rw (technology)?
|   +--:(generic)

```

```

+---rw generic?
      rt-types:generic
+---rw direction?
      te-label-direction
+---rw explicit-route-include-objects
+---rw route-object-include-object*
      [index]
+---rw index
      |
      uint32
+---rw (type)?
+---:(numbered-node-hop)
      +---rw numbered-node-hop
      |
      +---rw node-id
      |
      te-node-id
      +---rw hop-type?
      |
      te-hop-type
+---:(numbered-link-hop)
      +---rw numbered-link-hop
      |
      +---rw link-tp-id
      |
      te-tp-id
      +---rw hop-type?
      |
      te-hop-type
      +---rw direction?
      |
      te-link-direction
+---:(unnumbered-link-hop)
      +---rw unnumbered-link-hop
      |
      +---rw link-tp-id
      |
      te-tp-id
      +---rw node-id
      |
      te-node-id
      +---rw hop-type?
      |
      te-hop-type
      +---rw direction?
      |
      te-link-direction
+---:(as-number)
      +---rw as-number-hop
      |
      +---rw as-number
      |
      inet:as-number
      +---rw hop-type?
      |
      te-hop-type
+---:(label)
      +---rw label-hop
      |
      +---rw te-label

```

```

+---rw (technology)?
+---:(generic)
+---rw generic?
rt-types:generic
+---rw direction?
te-label-direction:generic
+---rw tiebreakers
+---rw tiebreaker* [tiebreaker-type]
+---rw tiebreaker-type identityref
+---:(objective-function)
{path-optimization-objective-function}?
+---rw objective-function
+---rw objective-function-type?
identityref
+---rw named-path-constraint? leafref
{te-types:named-path-constraints}?
+---rw te-bandwidth
+---rw (technology)?
+---:(generic)
+---rw generic? te-bandwidth
+---rw link-protection? identityref
+---rw setup-priority? uint8
+---rw hold-priority? uint8
+---rw signaling-type? identityref
+---rw path-metric-bounds
+---rw path-metric-bound* [metric-type]
+---rw metric-type identityref
+---rw upper-bound? uint64
+---rw path-affinities-values
+---rw path-affinities-value* [usage]
+---rw usage identityref
+---rw value? admin-groups
+---rw path-affinity-names
+---rw path-affinity-name* [usage]
+---rw usage identityref
+---rw affinity-name* [name]
+---rw name string
+---rw path-srlgs-lists
+---rw path-srlgs-list* [usage]
+---rw usage identityref
+---rw values* srlg
+---rw path-srlgs-names
+---rw path-srlgs-name* [usage]
+---rw usage identityref
+---rw names* string
+---rw disjointness?

```

				te-path-disjointness
			+--rw	explicit-route-objects-always
			+--rw	route-object-exclude-always* [index]
			+--rw	index uint32
			+--rw	(type)?
			+--:	(numbered-node-hop)
			+--rw	numbered-node-hop
			+--rw	node-id te-node-id
			+--rw	hop-type? te-hop-type
			+--:	(numbered-link-hop)
			+--rw	numbered-link-hop
			+--rw	link-tp-id te-tp-id
			+--rw	hop-type? te-hop-type
			+--rw	direction? te-link-direction
			+--:	(unnumbered-link-hop)
			+--rw	unnumbered-link-hop
			+--rw	link-tp-id te-tp-id
			+--rw	node-id te-node-id
			+--rw	hop-type? te-hop-type
			+--rw	direction? te-link-direction
			+--:	(as-number)
			+--rw	as-number-hop
			+--rw	as-number inet:as-number
			+--rw	hop-type? te-hop-type
			+--:	(label)
			+--rw	label-hop
			+--rw	te-label
			+--rw	(technology)?
			+--:	(generic)
			+--rw	generic?
				rt-types:generalized-la
			+--rw	direction?
				te-label-direction
			+--rw	route-object-include-exclude* [index]
			+--rw	explicit-route-usage? identityref
			+--rw	index uint32
			+--rw	(type)?
			+--:	(numbered-node-hop)
			+--rw	numbered-node-hop
			+--rw	node-id te-node-id
			+--rw	hop-type? te-hop-type
			+--:	(numbered-link-hop)
			+--rw	numbered-link-hop
			+--rw	link-tp-id te-tp-id
			+--rw	hop-type? te-hop-type
			+--rw	direction? te-link-direction
			+--:	(unnumbered-link-hop)

```

+--rw unnumbered-link-hop
+--rw link-tp-id      te-tp-id
+--rw node-id         te-node-id
+--rw hop-type?       te-hop-type
+--rw direction?      te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number        inet:as-number
+--rw hop-type?        te-hop-type
+--:(label)
+--rw label-hop
+--rw te-label
+--rw (technology)?
+--:(generic)
+--rw generic?
rt-types:generalized-label
+--rw direction?
te-label-direction
+--:(srlg)
+--rw srlg
+--rw srlg?    uint32
+--rw path-in-segment!
+--rw label-restrictions
+--rw label-restriction* [index]
+--rw restriction?       enumeration
+--rw index              uint32
+--rw label-start
+--rw te-label
+--rw (technology)?
+--:(generic)
+--rw generic?
rt-types:generalized-label
+--rw direction?
te-label-direction
+--rw label-end
+--rw te-label
+--rw (technology)?
+--:(generic)
+--rw generic?
rt-types:generalized-label
+--rw direction?
te-label-direction
+--rw label-step
+--rw (technology)?
+--:(generic)
+--rw generic?    int32
+--rw range-bitmap? yang:hex-string

```

```

+--rw path-out-segment!
  +--rw label-restrictions
    +--rw label-restriction* [index]
      +--rw restriction?    enumeration
      +--rw index          uint32
      +--rw label-start
        +--rw te-label
          +--rw (technology)?
            +--:(generic)
              +--rw generic?
                rt-types:generalized-label
          +--rw direction?
            te-label-direction
      +--rw label-end
        +--rw te-label
          +--rw (technology)?
            +--:(generic)
              +--rw generic?
                rt-types:generalized-label
          +--rw direction?
            te-label-direction
      +--rw label-step
        +--rw (technology)?
          +--:(generic)
            +--rw generic?    int32
        +--rw range-bitmap?  yang:hex-string
+--rw protection
  +--rw enable?                boolean
  +--rw protection-type?       identityref
  +--rw protection-reversion-disable? boolean
  +--rw hold-off-time?         uint32
  +--rw wait-to-revert?        uint16
  +--rw aps-signal-id?         uint8
+--rw restoration
  +--rw enable?                boolean
  +--rw restoration-type?
    | identityref
  +--rw restoration-scheme?
    | identityref
  +--rw restoration-reversion-disable? boolean
  +--rw hold-off-time?         uint32
  +--rw wait-to-restore?       uint16
  +--rw wait-to-revert?        uint16
+--ro computed-paths-properties
  +--ro computed-path-properties* [k-index]
    +--ro k-index              uint8
    +--ro path-properties
      +--ro path-metric* [metric-type]

```



```

|   +---ro metric-type          identityref
|   +---ro accumulative-value?  uint64
+---ro path-affinities-values
|   +---ro path-affinities-value* [usage]
|   +---ro usage                identityref
|   +---ro value?               admin-groups
+---ro path-affinity-names
|   +---ro path-affinity-name* [usage]
|   +---ro usage                identityref
|   +---ro affinity-name* [name]
|   +---ro name                 string
+---ro path-srlgs-lists
|   +---ro path-srlgs-list* [usage]
|   +---ro usage                identityref
|   +---ro values*             srlg
+---ro path-srlgs-names
|   +---ro path-srlgs-name* [usage]
|   +---ro usage                identityref
|   +---ro names*              string
+---ro path-route-objects
|   +---ro path-route-object* [index]
|   +---ro index
|   |   uint32
|   +---ro (type)?
|   |   +---:(numbered-node-hop)
|   |   |   +---ro numbered-node-hop
|   |   |   |   +---ro node-id      te-node-id
|   |   |   |   +---ro hop-type?
|   |   |   |   |   te-hop-type
|   |   |   +---:(numbered-link-hop)
|   |   |   |   +---ro numbered-link-hop
|   |   |   |   |   +---ro link-tp-id  te-tp-id
|   |   |   |   |   +---ro hop-type?
|   |   |   |   |   |   te-hop-type
|   |   |   |   |   +---ro direction?
|   |   |   |   |   |   te-link-direction
|   |   |   +---:(unnumbered-link-hop)
|   |   |   |   +---ro unnumbered-link-hop
|   |   |   |   |   +---ro link-tp-id  te-tp-id
|   |   |   |   |   +---ro node-id
|   |   |   |   |   |   te-node-id
|   |   |   |   |   +---ro hop-type?
|   |   |   |   |   |   te-hop-type
|   |   |   |   |   +---ro direction?
|   |   |   |   |   |   te-link-direction
|   |   +---:(as-number)
|   |   |   +---ro as-number-hop
|   |   |   +---ro as-number

```

```

|                                     |      inet:as-number
|                                     |      +---ro hop-type?
|                                     |          te-hop-type
|                                     |      +---:(label)
|                                     |          +---ro label-hop
|                                     |              +---ro te-label
|                                     |                  +---ro (technology)?
|                                     |                      +---:(generic)
|                                     |                          +---ro generic?
|                                     |                              rt-types:gener
alized-label
|                                     |      +---ro direction?
|                                     |          te-label-direction
|                                     |      +---ro te-bandwidth
|                                     |          +---ro (technology)?
|                                     |              +---:(generic)
|                                     |                  +---ro generic?    te-bandwidth
|                                     |      +---ro disjointness-type?
|                                     |          te-types:te-path-disjointness
+---ro computed-path-error-infos
|   +---ro computed-path-error-info* []
|       +---ro error-description?    string
|       +---ro error-timestamp?     yang:date-and-time
|       +---ro error-reason?        identityref
+---ro lsp-provisioning-error-infos
|   +---ro lsp-provisioning-error-info* []
|       +---ro error-description?    string
|       +---ro error-timestamp?     yang:date-and-time
|       +---ro error-node-id?       te-types:te-node-id
|       +---ro error-link-id?       te-types:te-tp-id
|       +---ro lsp-id?               uint16
+---ro lsps
|   +---ro lsp* [node lsp-id]
|       +---ro tunnel-name?
|           |         -> /te/lsps/lsp/tunnel-name
|       +---ro node         -> /te/lsps/lsp/node
|       +---ro lsp-id       -> /te/lsps/lsp/lsp-id
+---x tunnel-action
|   +---w input
|       |   +---w action-type?    identityref
|   +---ro output
|       +---ro action-result?    identityref
+---x protection-external-commands
|   +---w input
|       +---w protection-external-command?
|           |         identityref
|   +---w protection-group-ingress-node-id?
|       |         te-types:te-node-id

```

```

|         +---w protection-group-egress-node-id?
|         |         te-types:te-node-id
|         +---w path-ref?                                path-ref
|         +---w traffic-type?
|         |         enumeration
|         +---w extra-traffic-tunnel-ref?                tunnel-ref
+---ro lsp
+---ro lsp* [tunnel-name lsp-id node]
+---ro tunnel-name                                string
+---ro lsp-id                                    uint16
+---ro node
|         te-types:te-node-id
+---ro source?
|         te-types:te-node-id
+---ro destination?
|         te-types:te-node-id
+---ro tunnel-id?                                uint16
+---ro extended-tunnel-id?                        yang:dotted-quad
+---ro operational-state?                         identityref
+---ro signaling-type?                           identityref
+---ro origin-type?                              enumeration
+---ro lsp-resource-status?                       enumeration
+---ro lockout-of-normal?                         boolean
+---ro freeze?                                   boolean
+---ro lsp-protection-role?                       enumeration
+---ro lsp-protection-state?                     identityref
+---ro protection-group-ingress-node-id?
|         te-types:te-node-id
+---ro protection-group-egress-node-id?
|         te-types:te-node-id
+---ro lsp-record-route-information
+---ro lsp-record-route-information* [index]
+---ro index                                    uint32
+---ro (type)?
+---:(numbered-node-hop)
|         +---ro numbered-node-hop
|         |         +---ro node-id      te-node-id
|         |         +---ro flags*      path-attribute-flags
+---:(numbered-link-hop)
|         +---ro numbered-link-hop
|         |         +---ro link-tp-id   te-tp-id
|         |         +---ro flags*      path-attribute-flags
+---:(unnumbered-link-hop)
|         +---ro unnumbered-link-hop
|         |         +---ro link-tp-id   te-tp-id
|         |         +---ro node-id?    te-node-id
|         |         +---ro flags*      path-attribute-flags
+---:(label)

```

```

      +---ro label-hop
      |   +---ro te-label
      |   |   +---ro (technology)?
      |   |   |   +---:(generic)
      |   |   |   +---ro generic?
      |   |   |       rt-types:generalized-label
      |   +---ro direction?
      |       te-label-direction
      +---ro flags*          path-attribute-flags

rpcs:
  +---x tunnels-path-compute
  |   +---w input
  |   |   +---w path-compute-info
  |   +---ro output
  |       +---ro path-compute-result
  +---x tunnels-actions
  |   +---w input
  |   |   +---w tunnel-info
  |   |   |   +---w (filter-type)
  |   |   |   |   +---:(all-tunnels)
  |   |   |   |   |   +---w all          empty
  |   |   |   |   +---:(one-tunnel)
  |   |   |   |   +---w tunnel?      tunnel-ref
  |   |   +---w action-info
  |   |   |   +---w action?          identityref
  |   |   |   +---w disruptive?     empty
  |   +---ro output
  |       +---ro action-result?     identityref

```

Figure 8: TE Tunnel generic model YANG tree diagram

### 5.3. YANG Module

The generic TE YANG module 'ietf-te' imports the following modules:

- \* ietf-yang-types and ietf-inet-types defined in [RFC6991]
- \* ietf-te-types defined in [RFC8776]

This module references the following documents: [RFC6991], [RFC4875], [RFC7551], [RFC4206], [RFC4427], [RFC4872], [RFC3945], [RFC3209], [RFC6780], [RFC8800], and [RFC7308].

```
<CODE BEGINS> file "ietf-te@2021-10-22.yang"
module ietf-te {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te";

  /* Replace with IANA when assigned */

  prefix te;

  /* Import TE generic types */

  import ietf-te-types {
    prefix te-types;
    reference
      "RFC8776: Common YANG Data Types for Traffic Engineering.";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC6991: Common YANG Data Types.";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC6991: Common YANG Data Types.";
  }
}

organization
  "IETF Traffic Engineering Architecture and Signaling (TEAS)
  Working Group.";
contact
  "WG Web:  <http://tools.ietf.org/wg/teas/>
  WG List:  <mailto:teas@ietf.org>

  Editor:   Tarek Saad
            <mailto:tsaad@juniper.net>

  Editor:   Rakesh Gandhi
            <mailto:rgandhi@cisco.com>

  Editor:   Vishnu Pavan Beeram
            <mailto:vbeeram@juniper.net>

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

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

```
Editor:  Igor Bryskin
        <mailto:i_bryskin@yahoo.com>;
description
  "YANG data module for TE configuration, state, and RPCs.
  The model fully conforms to the Network Management
  Datastore Architecture (NMDA).

  Copyright (c) 2019 IETF Trust and the persons
  identified as authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).
  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision 2021-10-22 {
  description
    "Latest update to TE generic YANG module.";
  reference
    "RFCXXXX: A YANG Data Model for Traffic Engineering Tunnels
    and Interfaces.";
}

identity path-computation-error-reason {
  description
    "Base identity for path computation error reasons.";
}

identity path-computation-error-no-topology {
  base path-computation-error-reason;
  description
    "Path computation has failed because there is no topology
    with the provided topology-identifier.";
}

identity path-computation-error-no-dependent-server {
  base path-computation-error-reason;
  description
    "Path computation has failed because one or more dependent
```

```
        path computation servers are unavailable.
        The dependent path computation server could be
        a Backward-Recursive Path Computation (BRPC) downstream
        PCE or a child PCE.";
    reference
        "RFC5441, RFC8685";
}

identity path-computation-error-pce-unavailable {
    base path-computation-error-reason;
    description
        "Path computation has failed because PCE is not available.";
    reference
        "RFC5440";
}

identity path-computation-error-no-inclusion-hop {
    base path-computation-error-reason;
    description
        "Path computation has failed because there is no
        node or link provided by one or more inclusion hops.";
    reference
        "RFC8685";
}

identity path-computation-error-destination-unknown-in-domain {
    base path-computation-error-reason;
    description
        "Path computation has failed because the destination node is
        unknown in indicated destination domain.";
    reference
        "RFC8685";
}

identity path-computation-error-no-resource {
    base path-computation-error-reason;
    description
        "Path computation has failed because there is no
        available resource in one or more domains.";
    reference
        "RFC8685";
}

identity path-computation-error-child-pce-unresponsive {
    base path-computation-error-reason;
    description
        "Path computation has failed because child PCE is not
        responsive.";
```

```
        reference
          "RFC8685";
      }

      identity path-computation-error-destination-domain-unknown {
        base path-computation-error-reason;
        description
          "Path computation has failed because the destination domain
           was unknown.";
        reference
          "RFC8685";
      }

      identity path-computation-error-p2mp {
        base path-computation-error-reason;
        description
          "Path computation has failed because of P2MP reachability
           problem.";
        reference
          "RFC8306";
      }

      identity path-computation-error-no-gco-migration {
        base path-computation-error-reason;
        description
          "Path computation has failed because of no Global Concurrent
           Optimization (GCO) migration path found.";
        reference
          "RFC5557";
      }

      identity path-computation-error-no-gco-solution {
        base path-computation-error-reason;
        description
          "Path computation has failed because of no GCO solution
           found.";
        reference
          "RFC5557";
      }

      identity path-computation-error-path-not-found {
        base path-computation-error-reason;
        description
          "Path computation no path found error reason.";
        reference
          "RFC5440";
      }
    }
```



```
identity path-computation-error-pks-expansion {
  base path-computation-error-reason;
  description
    "Path computation has failed because of Path-Key Subobject
    (PKS) expansion failure.";
  reference
    "RFC5520";
}

identity path-computation-error-brpc-chain-unavailable {
  base path-computation-error-reason;
  description
    "Path computation has failed because PCE BRPC chain
    unavailable.";
  reference
    "RFC5441";
}

identity path-computation-error-source-unknown {
  base path-computation-error-reason;
  description
    "Path computation has failed because source node is unknown.";
  reference
    "RFC5440";
}

identity path-computation-error-destination-unknown {
  base path-computation-error-reason;
  description
    "Path computation has failed because destination node is
    unknown.";
  reference
    "RFC5440";
}

identity path-computation-error-no-server {
  base path-computation-error-reason;
  description
    "Path computation has failed because path computation
    server is unavailable.";
  reference
    "RFC5440";
}

identity tunnel-actions-type {
  description
    "TE tunnel actions type.";
}
```

```
identity tunnel-action-reoptimize {
  base tunnel-actions-type;
  description
    "Reoptimize tunnel action type.";
}

identity tunnel-admin-auto {
  base te-types:tunnel-admin-state-type;
  description
    "Tunnel administrative auto state. The administrative status
    in state datastore transitions to 'tunnel-admin-up' when the
    tunnel used by the client layer, and to 'tunnel-admin-down'
    when it is not used by the client layer.";
}

identity association-type-diversity {
  base te-types:association-type;
  description
    "Association Type diversity used to associate LSPs whose paths
    are to be diverse from each other.";
  reference
    "RFC8800";
}

identity protocol-origin-type {
  description
    "Base identity for protocol origin type.";
}

identity protocol-origin-api {
  base protocol-origin-type;
  description
    "Protocol origin is via Application Programmable Interface
    (API).";
}

identity protocol-origin-pcep {
  base protocol-origin-type;
  description
    "Protocol origin is Path Computation Engine Protocol (PCEP).";
  reference "RFC5440";
}

identity protocol-origin-bgp {
  base protocol-origin-type;
  description
    "Protocol origin is Border Gateway Protocol (BGP).";
  reference "RFC5512";
}

typedef tunnel-ref {
```

```
    type leafref {
      path "/te:te/te:tunnels/te:tunnel/te:name";
    }
    description
      "This type is used by data models that need to reference
       configured TE tunnel.";
  }

  typedef path-ref {
    type union {
      type leafref {
        path "/te:te/te:tunnels/te:tunnel/"
          + "te:primary-paths/te:primary-path/te:name";
      }
      type leafref {
        path "/te:te/te:tunnels/te:tunnel/"
          + "te:secondary-paths/te:secondary-path/te:name";
      }
    }
    description
      "This type is used by data models that need to reference
       configured primary or secondary path of a TE tunnel.";
  }

  typedef te-gen-node-id {
    type union {
      type te-types:te-node-id;
      type inet:ip-address;
    }
    description
      "Generic type that identifies a node in a TE topology.";
  }

  /**
   * TE tunnel generic groupings
   */

  grouping te-generic-node-id {
    description
      "A reusable grouping for a TE generic node identifier.";
    leaf id {
      type te-gen-node-id;
      description
        "The identifier of the node. Can be represented as IP
         address or dotted quad address.";
    }
    leaf type {
      type enumeration {
```

```
    enum ip {
      description
        "IP address representation of the node identifier.";
    }
    enum dotted-quad {
      description
        "Dotted quad address representation of the node
        identifier.";
    }
  }
  description
    "Type of node identifier representation.";
}

grouping primary-path {
  description
    "The tunnel primary path properties.";
  uses path-common-properties;
  uses path-preference;
  uses k-requested-paths;
  uses path-compute-info;
  uses path-state;
}

grouping primary-reverse-path {
  description
    "The tunnel primary reverse path properties.";
  reference
    "RFC7551";
  uses path-common-properties;
  uses path-compute-info;
  uses path-state;
}

grouping secondary-path {
  description
    "The tunnel secondary path properties.";
  uses path-common-properties;
  uses path-preference;
  uses path-compute-info;
  uses protection-restoration-properties;
  uses path-state;
}

grouping secondary-reverse-path {
  description
    "The tunnel secondary reverse path properties.";
```

```
    uses path-common-properties;
    uses path-preference;
    uses path-compute-info;
    uses protection-restoration-properties;
    uses path-state;
}

grouping path-common-properties {
  description
    "Common path attributes.";
  leaf name {
    type string;
    description
      "TE path name.";
  }
  leaf path-computation-method {
    type identityref {
      base te-types:path-computation-method;
    }
    default "te-types:path-locally-computed";
    description
      "The method used for computing the path, either
       locally computed, queried from a server or not
       computed at all (explicitly configured).";
  }
  container path-computation-server {
    when "derived-from-or-self(..path-computation-method, "
      + "'te-types:path-externally-queried') " {
      description
        "The path-computation server when the path is
         externally queried.";
    }
    uses te-generic-node-id;
    description
      "Address of the external path computation
       server.";
  }
  leaf compute-only {
    type empty;
    description
      "When set, the path is computed and updated whenever
       the topology is updated. No resources are committed
       or reserved in the network.";
  }
  leaf use-path-computation {
    when "derived-from-or-self(..path-computation-method, "
      + "'te-types:path-locally-computed') " {
      type boolean;
    }
  }
}
```

```
    default "true";
    description
        "When 'true' indicates the path is dynamically computed
        and/or validated against the Traffic-Engineering Database
        (TED), and when 'false' indicates no validation against
        the TED is required.";
}
leaf lockdown {
    type empty;
    description
        "Indicates no reoptimization to be attempted for this path.";
}
leaf path-scope {
    type identityref {
        base te-types:path-scope-type;
    }
    default "te-types:path-scope-end-to-end";
    config false;
    description
        "Path scope if segment or an end-to-end path.";
}
}

/* This grouping will be re-used in path-computation rpc */

grouping path-compute-info {
    description
        "Attributes used for path computation request.";
    uses tunnel-associations-properties;
    uses te-types:generic-path-optimization;
    leaf named-path-constraint {
        if-feature "te-types:named-path-constraints";
        type leafref {
            path "/te:te/te:globals/te:named-path-constraints/"
                + "te:named-path-constraint/te:name";
        }
        description
            "Reference to a globally defined named path constraint set.";
    }
    uses path-constraints-common;
}

/* This grouping will be re-used in path-computation rpc */

grouping path-preference {
    description
        "The path preference.";
    leaf preference {
```

```
    type uint8 {
      range "1..255";
    }
    default "1";
    description
      "Specifies a preference for this path. The lower the number
       higher the preference.";
  }
}

/* This grouping will be re-used in path-computation rpc */

grouping k-requested-paths {
  description
    "The k-shortest paths requests.";
  leaf k-requested-paths {
    type uint8;
    default "1";
    description
      "The number of k-shortest-paths requested from the path
       computation server and returned sorted by its optimization
       objective. The value 0 all possible paths.";
  }
}

grouping path-properties {
  description
    "TE computed path properties grouping.";
  uses te-types:generic-path-properties {
    augment "path-properties" {
      description
        "additional path properties returned by path computation.";
      uses te-types:te-bandwidth;
      leaf disjointness-type {
        type te-types:te-path-disjointness;
        config false;
        description
          "The type of resource disjointness.
           When reported for a primary path, it represents the
           minimum level of disjointness of all the secondary
           paths.
           When reported for a secondary path, it represents the
           disjointness of the secondary path.";
      }
    }
  }
}
```

```
grouping path-state {
  description
    "TE per path state parameters.";
  uses path-computation-response;
  uses lsp-provisioning-error-info {
    augment "lsp-provisioning-error-infos/"
      + "lsp-provisioning-error-info" {
      description
        "Augmentation of LSP provisioning information under a
        specific path.";
      leaf lsp-id {
        type uint16;
        description
          "The LSP-ID for which path computation was performed.";
      }
    }
  }
}
container lsps {
  config false;
  description
    "The TE LSPs container.";
  list lsp {
    key "node lsp-id";
    description
      "List of LSPs associated with the tunnel.";
    leaf tunnel-name {
      type leafref {
        path "/te:te/te:lsps/te:lsp/te:tunnel-name";
      }
      description "TE tunnel name.";
    }
    leaf node {
      type leafref {
        path "/te:te/te:lsps/te:lsp/te:node";
      }
      description "The node where the LSP state resides on.";
    }
    leaf lsp-id {
      type leafref {
        path "/te:te/te:lsps/te:lsp/te:lsp-id";
      }
      description "The TE LSP identifier.";
    }
  }
}

/* This grouping will be re-used in path-computation rpc */
```



```
grouping path-computation-response {
  description
    "Attributes reported by path computation response.";
  container computed-paths-properties {
    config false;
    description
      "Computed path properties container.";
    list computed-path-properties {
      key "k-index";
      description
        "List of computed paths.";
      leaf k-index {
        type uint8;
        description
          "The k-th path returned from the computation server.
          A lower k value path is more optimal than higher k
          value path(s)";
      }
      uses path-properties {
        description
          "The TE path computed properties.";
      }
    }
  }
}
container computed-path-error-infos {
  config false;
  description
    "Path computation information container.";
  list computed-path-error-info {
    description
      "List of path computation info entries.";
    leaf error-description {
      type string;
      description
        "Textual representation of the error occurred during
        path computation.";
    }
    leaf error-timestamp {
      type yang:date-and-time;
      description
        "Timestamp of last path computation attempt.";
    }
    leaf error-reason {
      type identityref {
        base path-computation-error-reason;
      }
      description
        "Reason for the path computation error.";
    }
  }
}
```

```
    }
  }
}

grouping lsp-provisioning-error-info {
  description
    "Grouping for LSP provisioning error information.";
  container lsp-provisioning-error-infos {
    config false;
    description
      "LSP provisioning error information.";
    list lsp-provisioning-error-info {
      description
        "List of LSP provisioning error info entries.";
      leaf error-description {
        type string;
        description
          "Textual representation of the error occurred during
          path computation.";
      }
      leaf error-timestamp {
        type yang:date-and-time;
        description
          "Timestamp of when the reported error occurred.";
      }
      leaf error-node-id {
        type te-types:te-node-id;
        default "0.0.0.0";
        description
          "Node identifier of node where error occurred.";
      }
      leaf error-link-id {
        type te-types:te-tp-id;
        default "0";
        description
          "Link ID where the error occurred.";
      }
    }
  }
}

grouping protection-restoration-properties-state {
  description
    "Protection parameters grouping.";
  leaf lockout-of-normal {
    type boolean;
    default "false";
  }
}
```

```
description
  "When set to 'True', it represents a lockout of normal
  traffic external command. When set to 'False', it
  represents a clear lockout of normal traffic external
  command. The lockout of normal traffic command applies
  to this Tunnel.";
reference
  "RFC4427";
}
leaf freeze {
  type boolean;
  default "false";
  description
    "When set to 'True', it represents a freeze external command.
    When set to 'False', it represents a clear freeze external
    command. The freeze command applies to all the Tunnels which
    are sharing the protection resources with this Tunnel.";
  reference
    "RFC4427";
}
leaf lsp-protection-role {
  type enumeration {
    enum working {
      description
        "A working LSP must be a primary LSP whilst a protecting
        LSP can be either a primary or a secondary LSP. Also,
        known as protected LSPs when working LSPs are associated
        with protecting LSPs.";
    }
    enum protecting {
      description
        "A secondary LSP is an LSP that has been provisioned
        in the control plane only; e.g. resource allocation
        has not been committed at the data plane.";
    }
  }
  default "working";
  description
    "LSP role type.";
  reference
    "RFC4872, section 4.2.1";
}
leaf lsp-protection-state {
  type identityref {
    base te-types:lsp-protection-state;
  }
  default "te-types:normal";
  description
```

```
        "The state of the APS state machine controlling which
        tunnels is using the resources of the protecting LSP.";
    }
    leaf protection-group-ingress-node-id {
        type te-types:te-node-id;
        default "0.0.0.0";
        description
            "Indicates the te-node-id of the protection group
            ingress node when the APS state represents an external
            command (LoP, SF, MS) applied to it or a WTR timer
            running on it. If the external command is not applied to
            the ingress node or the WTR timer is not running on it,
            this attribute is not specified. A value 0.0.0.0 is used
            when the te-node-id of the protection group ingress node is
            unknown (e.g., because the ingress node is outside the scope
            of control of the server)";
    }
    leaf protection-group-egress-node-id {
        type te-types:te-node-id;
        default "0.0.0.0";
        description
            "Indicates the te-node-id of the protection group egress node
            when the APS state represents an external command (LoP, SF,
            MS) applied to it or a WTR timer running on it. If the
            external command is not applied to the ingress node or
            the WTR timer is not running on it, this attribute is not
            specified. A value 0.0.0.0 is used when the te-node-id of
            the protection group ingress node is unknown (e.g., because
            the ingress node is outside the scope of control of the
            server)";
    }
}

grouping protection-restoration-properties {
    description
        "Protection and restoration parameters.";
    container protection {
        description
            "Protection parameters.";
        leaf enable {
            type boolean;
            default "false";
            description
                "A flag to specify if LSP protection is enabled.";
            reference
                "RFC4427";
        }
        leaf protection-type {
```

```
    type identityref {
      base te-types:lsp-protection-type;
    }
    default "te-types:lsp-protection-unprotected";
    description
      "LSP protection type.";
  }
  leaf protection-reversion-disable {
    type boolean;
    default "false";
    description
      "Disable protection reversion to working path.";
  }
  leaf hold-off-time {
    type uint32;
    units "milli-seconds";
    default "0";
    description
      "The time between the declaration of an SF or SD condition
       and the initialization of the protection switching
       algorithm.";
    reference
      "RFC4427";
  }
  leaf wait-to-revert {
    type uint16;
    units "seconds";
    description
      "Time to wait before attempting LSP reversion.";
    reference
      "RFC4427";
  }
  leaf aps-signal-id {
    type uint8 {
      range "1..255";
    }
    default "1";
    description
      "The APS signal number used to reference the traffic of
       this tunnel. The default value for normal traffic is 1.
       The default value for extra-traffic is 255. If not
       specified, non-default values can be assigned by the
       server, if and only if, the server controls both
       endpoints.";
    reference
      "RFC4427";
  }
}
```

```
container restoration {
  description
    "Restoration parameters.";
  leaf enable {
    type boolean;
    default "false";
    description
      "A flag to specify if LSP restoration is enabled.";
    reference
      "RFC4427";
  }
  leaf restoration-type {
    type identityref {
      base te-types:lsp-restoration-type;
    }
    default "te-types:lsp-restoration-restore-any";
    description
      "LSP restoration type.";
  }
  leaf restoration-scheme {
    type identityref {
      base te-types:restoration-scheme-type;
    }
    default "te-types:restoration-scheme-preconfigured";
    description
      "LSP restoration scheme.";
  }
  leaf restoration-reversion-disable {
    type boolean;
    default "false";
    description
      "Disable restoration reversion to working path.";
  }
  leaf hold-off-time {
    type uint32;
    units "milli-seconds";
    description
      "The time between the declaration of an SF or SD condition
       and the initialization of the protection switching
       algorithm.";
    reference
      "RFC4427";
  }
  leaf wait-to-restore {
    type uint16;
    units "seconds";
    description
      "Time to wait before attempting LSP restoration.";
```

```
        reference
          "RFC4427";
      }
      leaf wait-to-revert {
        type uint16;
        units "seconds";
        description
          "Time to wait before attempting LSP reversion.";
        reference
          "RFC4427";
      }
    }
  }
}

grouping tunnel-associations-properties {
  description
    "TE tunnel association grouping.";
  container association-objects {
    description
      "TE tunnel associations.";
    list association-object {
      key "association-key";
      unique "type id source/id source/type";
      description
        "List of association base objects.";
      reference
        "RFC4872";
      leaf association-key {
        type string;
        description
          "Association key used to identify a specific
            association in the list";
      }
      leaf type {
        type identityref {
          base te-types:association-type;
        }
        description
          "Association type.";
        reference
          "RFC4872";
      }
      leaf id {
        type uint16;
        description
          "Association identifier.";
        reference
          "RFC4872";
      }
    }
  }
}
```

```
    }
    container source {
      uses te-generic-node-id;
      description
        "Association source.";
      reference
        "RFC4872";
    }
  }
  list association-object-extended {
    key "association-key";
    unique
      "type id source/id source/type global-source extended-id";
    description
      "List of extended association objects.";
    reference
      "RFC6780";
    leaf association-key {
      type string;
      description
        "Association key used to identify a specific
        association in the list";
    }
    leaf type {
      type identityref {
        base te-types:association-type;
      }
      description
        "Association type.";
      reference
        "RFC4872, RFC6780";
    }
    leaf id {
      type uint16;
      description
        "Association identifier.";
      reference
        "RFC4872, RFC6780";
    }
  }
  container source {
    uses te-generic-node-id;
    description
      "Association source.";
    reference
      "RFC4872, RFC6780";
  }
  leaf global-source {
    type uint32;
```



```
        description
            "Association global source.";
        reference
            "RFC6780";
    }
    leaf extended-id {
        type yang:hex-string;
        description
            "Association extended identifier.";
        reference
            "RFC6780";
    }
}
}

/* TE tunnel configuration/state grouping */
/* These grouping will be re-used in path-computation rpc */

grouping encoding-and-switching-type {
    description
        "Common grouping to define the LSP encoding and
        switching types";
    leaf encoding {
        type identityref {
            base te-types:lsp-encoding-types;
        }
        description
            "LSP encoding type.";
        reference
            "RFC3945";
    }
    leaf switching-type {
        type identityref {
            base te-types:switching-capabilities;
        }
        description
            "LSP switching type.";
        reference
            "RFC3945";
    }
}

grouping tunnel-common-attributes {
    description
        "Common grouping to define the TE tunnel parameters";
    leaf source {
        type te-types:te-node-id;
```

```
        description
            "TE tunnel source node ID.";
    }
    leaf destination {
        type te-types:te-node-id;
        description
            "TE tunnel destination node identifier.";
    }
    leaf src-tunnel-tp-id {
        type binary;
        description
            "TE tunnel source termination point identifier.";
    }
    leaf dst-tunnel-tp-id {
        type binary;
        description
            "TE tunnel destination termination point identifier.";
    }
    leaf bidirectional {
        type boolean;
        default "false";
        description
            "Indicates a bidirectional co-routed LSP.";
    }
}

grouping tunnel-hierarchy-properties {
    description
        "A grouping for TE tunnel hierarchy information.";
    container hierarchy {
        description
            "Container for TE hierarchy related information.";
        container dependency-tunnels {
            description
                "List of tunnels that this tunnel can be potentially
                dependent on.";
            list dependency-tunnel {
                key "name";
                description
                    "A tunnel entry that this tunnel can potentially depend
                    on.";
                leaf name {
                    type leafref {
                        path "/te:te/te:tunnels/te:tunnel/te:name";
                        require-instance false;
                    }
                }
                description
                    "Dependency tunnel name. The tunnel may not have been
```

```
        instantiated yet.";
    }
    uses encoding-and-switching-type;
}
}
container hierarchical-link {
    description
        "Identifies a hierarchical link (in client layer)
        that this tunnel is associated with.";
    reference
        "RFC4206";
    leaf local-te-node-id {
        type te-types:te-node-id;
        default "0.0.0.0";
        description
            "The local TE node identifier.";
    }
    leaf local-te-link-tp-id {
        type te-types:te-tp-id;
        default "0";
        description
            "The local TE link termination point identifier.";
    }
    leaf remote-te-node-id {
        type te-types:te-node-id;
        default "0.0.0.0";
        description
            "Remote TE node identifier.";
    }
    uses te-types:te-topology-identifier {
        description
            "The topology identifier where the hierarchical link
            supported by this TE tunnel is instantiated.";
    }
}
}
}

grouping tunnel-properties {
    description
        "Top level grouping for tunnel properties.";
    leaf name {
        type string;
        description
            "TE tunnel name.";
    }
    leaf alias {
        type string;
```

```
    description
      "An alternate name of the TE tunnel that can be modified
       anytime during its lifetime.";
  }
  leaf identifier {
    type uint32;
    description
      "TE tunnel Identifier.";
    reference
      "RFC3209";
  }
  leaf color {
    type uint32;
    description "The color associated with the TE tunnel.";
    reference "RFC9012";
  }
  leaf description {
    type string;
    default "None";
    description
      "Textual description for this TE tunnel.";
  }
  leaf admin-state {
    type identityref {
      base te-types:tunnel-admin-state-type;
    }
    default "te-types:tunnel-admin-state-up";
    description
      "TE tunnel administrative state.";
  }
  leaf operational-state {
    type identityref {
      base te-types:tunnel-state-type;
    }
    config false;
    description
      "TE tunnel operational state.";
  }
  uses encoding-and-switching-type;
  uses tunnel-common-attributes;
  container controller {
    description
      "Contains tunnel data relevant to external controller(s).
       This target node may be augmented by external module(s),
       for example, to add data for PCEP initiated and/or
       delegated tunnels.";
    leaf protocol-origin {
      type identityref {
```

```
        base protocol-origin-type;
    }
    description
        "The protocol origin for instantiating the tunnel.";
    }
    leaf controller-entity-id {
        type string;
        description
            "An identifier unique within the scope of visibility that
             associated with the entity that controls the tunnel";
        reference "RFC8232";
    }
}
leaf reoptimize-timer {
    type uint16;
    units "seconds";
    description
        "Frequency of reoptimization of a traffic engineered LSP.";
}
uses tunnel-associations-properties;
uses protection-restoration-properties;
uses te-types:tunnel-constraints;
uses tunnel-hierarchy-properties;
container primary-paths {
    description
        "The set of primary paths.";
    list primary-path {
        key "name";
        description
            "List of primary paths for this tunnel.";
        uses primary-path;
        container primary-reverse-path {
            description
                "The reverse primary path properties.";
            uses primary-reverse-path;
            container candidate-secondary-reverse-paths {
                description
                    "The set of referenced candidate reverse secondary
                     paths from the full set of secondary reverse paths
                     which may be used for this primary path.";
                list candidate-secondary-reverse-path {
                    key "secondary-path";
                    ordered-by user;
                    description
                        "List of candidate secondary reverse path(s)";
                    leaf secondary-path {
                        type leafref {
                            path "../.../.../.../.../..."
                        }
                    }
                }
            }
        }
    }
}
```

```

        + "te:secondary-reverse-paths/"
        + "te:secondary-reverse-path/te:name";
    }
    description
      "A reference to the secondary reverse path that
       should be utilised when the containing primary
       reverse path option is in use.";
  }
}
}
}
container candidate-secondary-paths {
  description
    "The set of candidate secondary paths which may be used
     for this primary path. When secondary paths are
     specified in the list the path of the secondary LSP in
     use must be restricted to those path options referenced.
     The priority of the secondary paths is specified within
     the list. Higher priority values are less preferred -
     that is to say that a path with priority 0 is the most
     preferred path. In the case that the list is empty, any
     secondary path option may be utilised when the current
     primary path is in use.";
  list candidate-secondary-path {
    key "secondary-path";
    ordered-by user;
    description
      "List of candidate secondary paths for this tunnel.";
    leaf secondary-path {
      type leafref {
        path "../../../../../te:secondary-paths/"
          + "te:secondary-path/te:name";
      }
      description
        "A reference to the secondary path that should be
         utilised when the containing primary path option is
         in use.";
    }
    leaf active {
      type boolean;
      config false;
      description
        "Indicates the current active path option that has
         been selected of the candidate secondary paths.";
    }
  }
}
}
}

```

```
    }
    container secondary-paths {
      description
        "The set of secondary paths.";
      list secondary-path {
        key "name";
        description
          "List of secondary paths for this tunnel.";
        uses secondary-path;
      }
    }
    container secondary-reverse-paths {
      description
        "The set of secondary reverse paths.";
      list secondary-reverse-path {
        key "name";
        description
          "List of secondary paths for this tunnel.";
        uses secondary-reverse-path;
      }
    }
  }
}

grouping tunnel-actions {
  description
    "Tunnel actions.";
  action tunnel-action {
    description
      "Tunnel action.";
    input {
      leaf action-type {
        type identityref {
          base tunnel-actions-type;
        }
        description
          "Tunnel action type.";
      }
    }
    output {
      leaf action-result {
        type identityref {
          base te-types:te-action-result;
        }
        description
          "The result of the tunnel action operation.";
      }
    }
  }
}
```

```
}

grouping tunnel-protection-actions {
  description
    "Protection external command actions.";
  action protection-external-commands {
    input {
      leaf protection-external-command {
        type identityref {
          base te-types:protection-external-commands;
        }
        description
          "Protection external command.";
      }
      leaf protection-group-ingress-node-id {
        type te-types:te-node-id;
        description
          "When specified, indicates whether the action is
          applied on ingress node.
          By default, if neither ingress nor egress node-id
          is set, the action applies to ingress node only.";
      }
      leaf protection-group-egress-node-id {
        type te-types:te-node-id;
        description
          "When specified, indicates whether the action is
          applied on egress node.
          By default, if neither ingress nor egress node-id
          is set, the action applies to ingress node only.";
      }
      leaf path-ref {
        type path-ref;
        description
          "Indicates to which path the external command applies
          to.";
      }
      leaf traffic-type {
        type enumeration {
          enum normal-traffic {
            description
              "The manual-switch or forced-switch command applies
              to the normal traffic (this Tunnel).";
          }
          enum null-traffic {
            description
              "The manual-switch or forced-switch command applies
              to the null traffic.";
          }
        }
      }
    }
  }
}
```



```
        enum extra-traffic {
            description
                "The manual-switch or forced-switch command applies
                to the extra traffic (the extra-traffic Tunnel
                sharing protection bandwidth with this Tunnel).";
        }
    }
    description
        "Indicates whether the manual-switch or forced-switch
        commands applies to the normal traffic, the null traffic
        or the extra-traffic.";
    reference
        "RFC4427";
}
leaf extra-traffic-tunnel-ref {
    type tunnel-ref;
    description
        "In case there are multiple extra-traffic tunnels sharing
        protection bandwidth with this Tunnel (m:n protection),
        represents which extra-traffic Tunnel the manual-switch
        or forced-switch to extra-traffic command applies to.";
}
}
}

/** End of TE tunnel groupings */
/**
 * LSP related generic groupings
 */

grouping lsp-record-route-information-state {
    description
        "LSP Recorded route information grouping.";
    container lsp-record-route-information {
        description
            "RSVP recorded route object information.";
        list lsp-record-route-information {
            when "../../origin-type = 'ingress'" {
                description
                    "Applicable on ingress LSPs only.";
            }
        }
        key "index";
        description
            "Record route list entry.";
        uses te-types:record-route-state;
    }
}
```

```
    }

    grouping lsp-grouping {
      description
        "LSPs state operational data grouping.";
      container lsp {
        config false;
        description
          "TE LSPs state container.";
        list lsp {
          key "tunnel-name lsp-id node";
          unique "source destination tunnel-id lsp-id "
            + "extended-tunnel-id";
          description
            "List of LSPs associated with the tunnel.";
          leaf tunnel-name {
            type string;
            description "The TE tunnel name.";
          }
          leaf lsp-id {
            type uint16;
            description
              "Identifier used in the SENDER_TEMPLATE and the
              FILTER_SPEC that can be changed to allow a sender to
              share resources with itself.";
            reference
              "RFC3209";
          }
          leaf node {
            type te-types:te-node-id;
            description
              "The node where the TE LSP state resides on.";
          }
          uses lsp-properties-state;
          uses lsp-record-route-information-state;
        }
      }
    }

    /*** End of TE LSP groupings ***/
    /**
     * TE global generic groupings
     */
    /* Global named admin-groups configuration data */

    grouping named-admin-groups-properties {
      description
        "Global named administrative groups configuration
```

```
        grouping.";
    leaf name {
        type string;
        description
            "A string name that uniquely identifies a TE
            interface named admin-group.";
    }
    leaf bit-position {
        type uint32;
        description
            "Bit position representing the administrative group.";
        reference
            "RFC3209 and RFC7308";
    }
}

grouping named-admin-groups {
    description
        "Global named administrative groups configuration
        grouping.";
    container named-admin-groups {
        description
            "TE named admin groups container.";
        list named-admin-group {
            if-feature "te-types:extended-admin-groups";
            if-feature "te-types:named-extended-admin-groups";
            key "name";
            description
                "List of named TE admin-groups.";
            uses named-admin-groups-properties;
        }
    }
}

/* Global named admin-srlgs configuration data */

grouping named-srlgs {
    description
        "Global named SRLGs configuration grouping.";
    container named-srlgs {
        description
            "TE named SRLGs container.";
        list named-srlg {
            if-feature "te-types:named-srlg-groups";
            key "name";
            description
                "A list of named SRLG groups.";
            leaf name {
```

```
        type string;
        description
            "A string name that uniquely identifies a TE
             interface named SRLG.";
    }
    leaf value {
        type te-types:srlg;
        description
            "An SRLG value.";
    }
    leaf cost {
        type uint32;
        description
            "SRLG associated cost. Used during path to append
             the path cost when traversing a link with this SRLG.";
    }
}
}
}

/* Global named paths constraints configuration data */

grouping path-constraints-common {
    description
        "Global named path constraints configuration
         grouping.";
    uses te-types:common-path-constraints-attributes {
        description
            "The constraints applicable to the path. This includes:
            - The path bandwidth constraint
            - The path link protection type constraint
            - The path setup/hold priority constraint
            - path signaling type constraint
            - path metric bounds constraint. The unit of path metric
              bound is interpreted in the context of the metric-type.
              For example for metric-type 'path-metric-loss', the bound
              is multiples of the basic unit 0.000003% as described
              in RFC7471 for OSPF, and RFC8570 for ISIS.
            - path affinity constraints
            - path SRLG constraints";
    }
    uses te-types:generic-path-disjointness;
    uses te-types:path-constraints-route-objects;
    container path-in-segment {
        presence "The end-to-end tunnel starts in a previous domain;
                 this tunnel is a segment in the current domain.";
        description
    }
}
```

```
        "If an end-to-end tunnel crosses multiple domains using
        the same technology, some additional constraints have to be
        taken in consideration in each domain.
        This TE tunnel segment is stitched to the upstream TE tunnel
        segment.";
    uses te-types:label-set-info;
}
container path-out-segment {
    presence
        "The end-to-end tunnel is not terminated in this domain;
        this tunnel is a segment in the current domain.";
    description
        "If an end-to-end tunnel crosses multiple domains using
        the same technology, some additional constraints have to be
        taken in consideration in each domain.
        This TE tunnel segment is stitched to the downstream TE
        tunnel segment.";
    uses te-types:label-set-info;
}
}

grouping named-path-constraints {
    description
        "Global named path constraints configuration
        grouping.";
    container named-path-constraints {
        description
            "TE named path constraints container.";
        list named-path-constraint {
            if-feature "te-types:named-path-constraints";
            key "name";
            leaf name {
                type string;
                description
                    "A string name that uniquely identifies a
                    path constraint set.";
            }
            uses path-constraints-common;
            description
                "A list of named path constraints.";
        }
    }
}

/* TE globals container data */

grouping globals-grouping {
    description
```

```
    "Globals TE system-wide configuration data grouping.";
    container globals {
      description
        "Globals TE system-wide configuration data container.";
      uses named-admin-groups;
      uses named-srlgs;
      uses named-path-constraints;
    }
  }

/* TE tunnels container data */

grouping tunnels-grouping {
  description
    "Tunnels TE configuration data grouping.";
  container tunnels {
    description
      "Tunnels TE configuration data container.";
    list tunnel {
      key "name";
      description
        "The list of TE tunnels.";
      uses tunnel-properties;
      uses tunnel-actions;
      uses tunnel-protection-actions;
    }
  }
}

/* TE LSPs ephemeral state container data */

grouping lsp-properties-state {
  description
    "LSPs state operational data grouping.";
  leaf source {
    type te-types:te-node-id;
    description
      "Tunnel sender address extracted from
       SENDER_TEMPLATE object.";
    reference
      "RFC3209";
  }
  leaf destination {
    type te-types:te-node-id;
    description
      "The tunnel endpoint address extracted from SESSION object.";
    reference
      "RFC3209";
  }
}
```

```
    }
    leaf tunnel-id {
      type uint16;
      description
        "The tunnel identifier used in the SESSION that remains
         constant over the life of the tunnel.";
      reference
        "RFC3209";
    }
    leaf extended-tunnel-id {
      type yang:dotted-quad;
      description
        "The LSP Extended Tunnel ID.";
      reference
        "RFC3209";
    }
    leaf operational-state {
      type identityref {
        base te-types:lsp-state-type;
      }
      description
        "The LSP operational state.";
    }
    leaf signaling-type {
      type identityref {
        base te-types:path-signaling-type;
      }
      description
        "The signaling protocol used to set up this LSP.";
    }
    leaf origin-type {
      type enumeration {
        enum ingress {
          description
            "Origin ingress.";
        }
        enum egress {
          description
            "Origin egress.";
        }
        enum transit {
          description
            "Origin transit.";
        }
      }
      default "ingress";
      description
        "The origin of the LSP relative to the location of the local
```

```
        switch in the path.";
    }
    leaf lsp-resource-status {
        type enumeration {
            enum primary {
                description
                "A primary LSP is a fully established LSP for which the
                resource allocation has been committed at the data
                plane.";
            }
            enum secondary {
                description
                "A secondary LSP is an LSP that has been provisioned
                in the control plane only; e.g. resource allocation
                has not been committed at the data plane.";
            }
        }
        default "primary";
        description
        "LSP resource allocation state.";
        reference
        "RFC4872, section 4.2.1";
    }
    uses protection-restoration-properties-state;
}

/** End of TE global groupings */
/**
 * TE container
 */

container te {
    presence "Enable TE feature.";
    description
    "TE global container.";
    /* TE Global Data */
    uses globals-grouping;

    /* TE Tunnel Data */
    uses tunnels-grouping;

    /* TE LSPs Data */
    uses lsps-grouping;
}

/* TE Tunnel RPCs/execution Data */

rpc tunnels-path-compute {
```



```
description
  "TE tunnels RPC nodes.";
input {
  container path-compute-info {
    /*
     * An external path compute module may augment this
     * target.
     */
    description
      "RPC input information.";
  }
}
output {
  container path-compute-result {
    /*
     * An external path compute module may augment this
     * target.
     */
    description
      "RPC output information.";
  }
}

rpc tunnels-actions {
  description
    "TE tunnels actions RPC";
  input {
    container tunnel-info {
      description
        "TE tunnel information.";
      choice filter-type {
        mandatory true;
        description
          "Filter choice.";
        case all-tunnels {
          leaf all {
            type empty;
            mandatory true;
            description
              "Apply action on all TE tunnels.";
          }
        }
        case one-tunnel {
          leaf tunnel {
            type tunnel-ref;
            description
              "Apply action on the specific TE tunnel.";
          }
        }
      }
    }
  }
}
```

```

        }
    }
}
container action-info {
    description
        "TE tunnel action information.";
    leaf action {
        type identityref {
            base tunnel-actions-type;
        }
        description
            "The action type.";
    }
    leaf disruptive {
        when "derived-from-or-self(..../action, "
            + "'te:tunnel-action-reoptimize')";
        type empty;
        description
            "Specifies whether or not the reoptimization action
            is allowed to be disruptive.";
    }
}
}
output {
    leaf action-result {
        type identityref {
            base te-types:te-action-result;
        }
        description
            "The result of the tunnel action operation.";
    }
}
}
}
<CODE ENDS>

```

Figure 9: TE Tunnel data model YANG module

## 6. TE Device YANG Model

The device TE YANG module ('ietf-te-device') models data that is specific to managing a TE device. This module augments the generic TE YANG module.

## 6.1. Module Structure

### 6.1.1. TE Interfaces

This branch of the model manages TE interfaces that are present on a device. Examples of TE interface properties are:

- \* Maximum reservable bandwidth, bandwidth constraints (BC)
- \* Flooding parameters
  - Flooding intervals and threshold values
- \* interface attributes
  - (Extended) administrative groups
  - SRLG values
  - TE metric value
- \* Fast reroute backup tunnel properties (such as static, auto-tunnel)

The derived state associated with interfaces is grouped under the interface "state" sub-container as shown in Figure 10. This covers state data such as:

- \* Bandwidth information: maximum bandwidth, available bandwidth at different priorities and for each class-type (CT)
- \* List of admitted LSPs
  - Name, bandwidth value and pool, time, priority
- \* Statistics: state counters, flooding counters, admission counters (accepted/rejected), preemption counters
- \* Adjacency information
  - Neighbor address
  - Metric value

```

module: ietf-te-device
  augment /te:te:
    +--rw interfaces
    .
    +-- rw te-dev:te-attributes
       <<intended configuration>>
    .
    +-- ro state
       <<derived state associated with the TE interface>>

```

Figure 10: TE interface state YANG subtree

## 6.2. Tree Diagram

Figure 11 shows the tree diagram of the device TE YANG model defined in modules 'ietf-te.yang'.

```

module: ietf-te-device
  augment /te:te:
    +--rw interfaces
    |
    |   +--rw threshold-type?          enumeration
    |   +--rw delta-percentage?       rt-types:percentage
    |   +--rw threshold-specification? enumeration
    |   +--rw up-thresholds*          rt-types:percentage
    |   +--rw down-thresholds*       rt-types:percentage
    |   +--rw up-down-thresholds*    rt-types:percentage
    |   +--rw interface* [interface]
    |       +--rw interface          if:interface-ref
    |       +--rw te-metric?
    |           |
    |           |   te-types:te-metric
    |       +--rw (admin-group-type)?
    |           |
    |           |   +--:(value-admin-groups)
    |           |   |
    |           |   |   +--rw (value-admin-group-type)?
    |           |   |   |
    |           |   |   |   +--:(admin-groups)
    |           |   |   |   |
    |           |   |   |   |   +--rw admin-group?
    |           |   |   |   |   |
    |           |   |   |   |   |   te-types:admin-group
    |           |   |   |   |   +--:(extended-admin-groups)
    |           |   |   |   |   |   {te-types:extended-admin-groups}?
    |           |   |   |   |   |   +--rw extended-admin-group?
    |           |   |   |   |   |   |
    |           |   |   |   |   |   |   te-types:extended-admin-group
    |           |   |   |   |   +--:(named-admin-groups)
    |           |   |   |   |   |   +--rw named-admin-groups* [named-admin-group]
    |           |   |   |   |   |   |
    |           |   |   |   |   |   |   {te-types:extended-admin-groups, te-types:named-
    |           |   |   |   |   |   |   extended-admin-groups}?
    |           |   |   |   |   |   |   +--rw named-admin-group    leafref
    |           |   |   |   |   +--rw (srlg-type)?
    |           |   |   |   |   |
    |           |   |   |   |   |   +--:(value-srlgs)
    |           |   |   |   |   |   |
    |           |   |   |   |   |   |   +--rw values* [value]

```

```

|         +---rw value      uint32
+---:(named-srlgs)
|         +---rw named-srlgs* [named-srlg]
|             {te-types:named-srlg-groups}?
|         +---rw named-srlg    leafref
+---rw threshold-type?          enumeration
+---rw delta-percentage?
|         rt-types:percentage
+---rw threshold-specification?  enumeration
+---rw up-thresholds*
|         rt-types:percentage
+---rw down-thresholds*
|         rt-types:percentage
+---rw up-down-thresholds*
|         rt-types:percentage
+---rw switching-capabilities* [switching-capability]
|         +---rw switching-capability    identityref
|         +---rw encoding?                identityref
+---ro state
|         +---ro te-advertisements-state
|             +---ro flood-interval?      uint32
|             +---ro last-flooded-time?   uint32
|             +---ro next-flooded-time?   uint32
|             +---ro last-flooded-trigger? enumeration
|             +---ro advertised-level-areas* [level-area]
|                 +---ro level-area      uint32
+---rw performance-thresholds
augment /te:te/te:globals:
+---rw lsp-install-interval?      uint32
+---rw lsp-cleanup-interval?      uint32
+---rw lsp-invalidation-interval? uint32
augment /te:te/te:tunnels/te:tunnel:
+---rw path-invalidation-action?  identityref
+---rw lsp-install-interval?      uint32
+---rw lsp-cleanup-interval?      uint32
+---rw lsp-invalidation-interval? uint32
augment /te:te/te:lsps/te:lsp:
+---ro lsp-timers
|         +---ro life-time?      uint32
|         +---ro time-to-install? uint32
|         +---ro time-to-destroy? uint32
+---ro downstream-info
|         +---ro nhop?            te-types:te-tp-id
|         +---ro outgoing-interface? if:interface-ref
|         +---ro neighbor
|             +---ro id?          te-gen-node-id
|             +---ro type?        enumeration
+---ro label?                    rt-types:generalized-label

```

```

+--ro upstream-info
  +--ro phop?      te-types:te-tp-id
  +--ro neighbor
    | +--ro id?      te-gen-node-id
    | +--ro type?    enumeration
  +--ro label?      rt-types:generalized-label

rpcs:
  +---x link-state-update
    +---w input
      +---w (filter-type)
        +---:(match-all)
          | +---w all          empty
        +---:(match-one-interface)
          +---w interface?    if:interface-ref

```

Figure 11: TE Tunnel device model YANG tree diagram

### 6.3. YANG Module

The device TE YANG module 'ietf-te-device' imports the following module(s):

- \* ietf-yang-types and ietf-inet-types defined in [RFC6991]
- \* ietf-interfaces defined in [RFC8343]
- \* ietf-routing-types defined in [RFC8294]
- \* ietf-te-types defined in [RFC8776]
- \* ietf-te defined in this document

```

<CODE BEGINS> file "ietf-te-device@2021-10-22.yang"
module ietf-te-device {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-device";

  /* Replace with IANA when assigned */

  prefix te-dev;

  /* Import TE module */

  import ietf-te {
    prefix te;
    reference
      "draft-ietf-teas-yang-te: A YANG Data Model for Traffic

```

```
    Engineering Tunnels and Interfaces";
}

/* Import TE types */

import ietf-te-types {
    prefix te-types;
    reference
        "RFC8776: Common YANG Data Types for Traffic Engineering.";
}
import ietf-interfaces {
    prefix if;
    reference
        "RFC8343: A YANG Data Model for Interface Management";
}
import ietf-routing-types {
    prefix rt-types;
    reference
        "RFC8294: Common YANG Data Types for the Routing Area";
}

organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
    Working Group";
contact
    "WG Web:    <http://tools.ietf.org/wg/teas/>
    WG List:    <mailto:teas@ietf.org>

    Editor:     Tarek Saad
                <mailto:tsaad@juniper.net>

    Editor:     Rakesh Gandhi
                <mailto:rgandhi@cisco.com>

    Editor:     Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

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

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

    Editor:     Igor Bryskin
                <mailto:i_bryskin@yahoo.com>";
description
    "YANG data module for TE device configurations,
    state, and RPCs. The model fully conforms to the
```

Network Management Datastore Architecture (NMDA).

Copyright (c) 2019 IETF Trust and the persons  
identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or  
without modification, is permitted pursuant to, and subject  
to the license terms contained in, the Simplified BSD License  
set forth in Section 4.c of the IETF Trust's Legal Provisions  
Relating to IETF Documents  
(<https://trustee.ietf.org/license-info>).  
This version of this YANG module is part of RFC XXXX; see  
the RFC itself for full legal notices.";

```
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
```

```
revision 2021-10-22 {
  description
    "Latest update to TE device YANG module.";
  reference
    "RFCXXXX: A YANG Data Model for Traffic Engineering Tunnels
    and Interfaces";
}
```

```
/**
 * TE LSP device state grouping
 */
```

```
grouping lsps-device-info {
  description
    "TE LSP device state grouping.";
  container lsp-timers {
    when "../te:origin-type = 'ingress'" {
      description
        "Applicable to ingress LSPs only.";
    }
    description
      "Ingress LSP timers.";
    leaf life-time {
      type uint32;
      units "seconds";
      description
        "TE LSP lifetime.";
    }
    leaf time-to-install {
```



```
        type uint32;
        units "seconds";
        description
            "TE LSP installation delay time.";
    }
    leaf time-to-destroy {
        type uint32;
        units "seconds";
        description
            "TE LSP expiration delay time.";
    }
}
container downstream-info {
    when "../te:origin-type != 'egress'" {
        description
            "Downstream information of the LSP.";
    }
    description
        "downstream information.";
    leaf nhop {
        type te-types:te-tp-id;
        description
            "downstream next-hop address.";
    }
    leaf outgoing-interface {
        type if:interface-ref;
        description
            "downstream interface.";
    }
    container neighbor {
        uses te:te-generic-node-id;
        description
            "downstream neighbor address.";
    }
    leaf label {
        type rt-types:generalized-label;
        description
            "downstream label.";
    }
}
container upstream-info {
    when "../te:origin-type != 'ingress'" {
        description
            "Upstream information of the LSP.";
    }
    description
        "upstream information.";
    leaf phop {
```

```
        type te-types:te-tp-id;
        description
            "upstream next-hop or previous-hop address.";
    }
    container neighbor {
        uses te:te-generic-node-id;
        description
            "upstream neighbor address.";
    }
    leaf label {
        type rt-types:generalized-label;
        description
            "upstream label.";
    }
}

/**
 * Device general groupings.
 */

grouping lsp-device-timers {
    description
        "Device TE LSP timers configs.";
    leaf lsp-install-interval {
        type uint32;
        units "seconds";
        description
            "TE LSP installation delay time.";
    }
    leaf lsp-cleanup-interval {
        type uint32;
        units "seconds";
        description
            "TE LSP cleanup delay time.";
    }
    leaf lsp-invalidation-interval {
        type uint32;
        units "seconds";
        description
            "TE LSP path invalidation before taking action delay time.";
    }
}

/**
 * TE global device groupings
 */
/* TE interface container data */
```

```
grouping interfaces-grouping {
  description
    "TE interface configuration data grouping.";
  container interfaces {
    description
      "Configuration data model for TE interfaces.";
    uses te-all-attributes;
    list interface {
      key "interface";
      description
        "TE interfaces.";
      leaf interface {
        type if:interface-ref;
        description
          "TE interface name.";
      }
      /* TE interface parameters */
      uses te-attributes;
    }
  }
}

/**
 * TE interface device groupings
 */

grouping te-admin-groups-config {
  description
    "TE interface affinities grouping.";
  choice admin-group-type {
    description
      "TE interface administrative groups
      representation type.";
    case value-admin-groups {
      choice value-admin-group-type {
        description
          "choice of admin-groups.";
        case admin-groups {
          description
            "Administrative group/Resource
            class/Color.";
          leaf admin-group {
            type te-types:admin-group;
            description
              "TE interface administrative group.";
          }
        }
      }
    case extended-admin-groups {
```

```
        if-feature "te-types:extended-admin-groups";
        description
            "Extended administrative group/Resource
            class/Color.";
        leaf extended-admin-group {
            type te-types:extended-admin-group;
            description
                "TE interface extended administrative group.";
        }
    }
}
}
case named-admin-groups {
    list named-admin-groups {
        if-feature "te-types:extended-admin-groups";
        if-feature "te-types:named-extended-admin-groups";
        key "named-admin-group";
        description
            "A list of named admin-group entries.";
        leaf named-admin-group {
            type leafref {
                path "../..../te:globals/"
                    + "te:named-admin-groups/te:named-admin-group/"
                    + "te:name";
            }
            description
                "A named admin-group entry.";
        }
    }
}
}
}

/* TE interface SRLGs */

grouping te-srlgs-config {
    description
        "TE interface SRLG grouping.";
    choice srlg-type {
        description
            "Choice of SRLG configuration.";
        case value-srlgs {
            list values {
                key "value";
                description
                    "List of SRLG values that
                    this link is part of.";
                leaf value {
```

```
        type uint32 {
            range "0..4294967295";
        }
        description
            "Value of the SRLG";
    }
}
}
case named-srlgs {
    list named-srlgs {
        if-feature "te-types:named-srlg-groups";
        key "named-srlg";
        description
            "A list of named SRLG entries.";
        leaf named-srlg {
            type leafref {
                path "../..../te:globals/"
                    + "te:named-srlgs/te:named-srlg/te:name";
            }
            description
                "A named SRLG entry.";
        }
    }
}
}
}
}

grouping te-igp-flooding-bandwidth-config {
    description
        "Configurable items for igp flooding bandwidth
        threshold configuration.";
    leaf threshold-type {
        type enumeration {
            enum delta {
                description
                    "'delta' indicates that the local
                    system should flood IGP updates when a
                    change in reserved bandwidth >= the specified
                    delta occurs on the interface.";
            }
            enum threshold-crossed {
                description
                    "THRESHOLD-CROSSED indicates that
                    the local system should trigger an update (and
                    hence flood) the reserved bandwidth when the
                    reserved bandwidth changes such that it crosses,
                    or becomes equal to one of the threshold values.";
            }
        }
    }
}
```

```
}
description
  "The type of threshold that should be used to specify the
  values at which bandwidth is flooded. 'delta' indicates that
  the local system should flood IGP updates when a change in
  reserved bandwidth >= the specified delta occurs on the
  interface. Where 'threshold-crossed' is specified, the local
  system should trigger an update (and hence flood) the
  reserved bandwidth when the reserved bandwidth changes such
  that it crosses, or becomes equal to one of the threshold
  values."
}
leaf delta-percentage {
  when "../threshold-type = 'delta'" {
    description
      "The percentage delta can only be specified when the
      threshold type is specified to be a percentage delta of
      the reserved bandwidth."
  }
  type rt-types:percentage;
  description
    "The percentage of the maximum-reservable-bandwidth
    considered as the delta that results in an IGP update
    being flooded."
}
leaf threshold-specification {
  when "../threshold-type = 'threshold-crossed'" {
    description
      "The selection of whether mirrored or separate threshold
      values are to be used requires user specified thresholds
      to be set."
  }
  type enumeration {
    enum mirrored-up-down {
      description
        "mirrored-up-down indicates that a single set of
        threshold values should be used for both increasing
        and decreasing bandwidth when determining whether
        to trigger updated bandwidth values to be flooded
        in the IGP TE extensions."
    }
    enum separate-up-down {
      description
        "separate-up-down indicates that a separate
        threshold values should be used for the increasing
        and decreasing bandwidth when determining whether
        to trigger updated bandwidth values to be flooded
        in the IGP TE extensions."
    }
  }
}
```

```
    }
  }
  description
    "This value specifies whether a single set of threshold
    values should be used for both increasing and decreasing
    bandwidth when determining whether to trigger updated
    bandwidth values to be flooded in the IGP TE extensions.
    'mirrored-up-down' indicates that a single value (or set of
    values) should be used for both increasing and decreasing
    values, where 'separate-up-down' specifies that the
    increasing and decreasing values will be separately
    specified.";
}
leaf-list up-thresholds {
  when "../threshold-type = 'threshold-crossed'"
    + "and ../threshold-specification = 'separate-up-down'" {
    description
      "A list of up-thresholds can only be specified when the
      bandwidth update is triggered based on crossing a
      threshold and separate up and down thresholds are
      required.";
  }
  type rt-types:percentage;
  description
    "The thresholds (expressed as a percentage of the maximum
    reservable bandwidth) at which bandwidth updates are to be
    triggered when the bandwidth is increasing.";
}
leaf-list down-thresholds {
  when "../threshold-type = 'threshold-crossed'"
    + "and ../threshold-specification = 'separate-up-down'" {
    description
      "A list of down-thresholds can only be specified when the
      bandwidth update is triggered based on crossing a
      threshold and separate up and down thresholds are
      required.";
  }
  type rt-types:percentage;
  description
    "The thresholds (expressed as a percentage of the maximum
    reservable bandwidth) at which bandwidth updates are to be
    triggered when the bandwidth is decreasing.";
}
leaf-list up-down-thresholds {
  when "../threshold-type = 'threshold-crossed'"
    + "and ../threshold-specification = 'mirrored-up-down'" {
    description
      "A list of thresholds corresponding to both increasing
```

```
        and decreasing bandwidths can be specified only when an
        update is triggered based on crossing a threshold, and
        the same up and down thresholds are required.";
    }
    type rt-types:percentage;
    description
        "The thresholds (expressed as a percentage of the maximum
        reservable bandwidth of the interface) at which bandwidth
        updates are flooded - used both when the bandwidth is
        increasing and decreasing.";
}
}

/* TE interface metric */

grouping te-metric-config {
    description
        "TE interface metric grouping.";
    leaf te-metric {
        type te-types:te-metric;
        description
            "TE interface metric.";
    }
}

/* TE interface switching capabilities */

grouping te-switching-cap-config {
    description
        "TE interface switching capabilities.";
    list switching-capabilities {
        key "switching-capability";
        description
            "List of interface capabilities for this interface.";
        leaf switching-capability {
            type identityref {
                base te-types:switching-capabilities;
            }
            description
                "Switching Capability for this interface.";
        }
        leaf encoding {
            type identityref {
                base te-types:lsp-encoding-types;
            }
            description
                "Encoding supported by this interface.";
        }
    }
}
```



```
    }  
  }  
  
  grouping te-advertisements-state {  
    description  
      "TE interface advertisements state grouping.";  
    container te-advertisements-state {  
      description  
        "TE interface advertisements state container.";  
      leaf flood-interval {  
        type uint32;  
        description  
          "The periodic flooding interval.";  
      }  
      leaf last-flooded-time {  
        type uint32;  
        units "seconds";  
        description  
          "Time elapsed since last flooding in seconds.";  
      }  
      leaf next-flooded-time {  
        type uint32;  
        units "seconds";  
        description  
          "Time remained for next flooding in seconds.";  
      }  
      leaf last-flooded-trigger {  
        type enumeration {  
          enum link-up {  
            description  
              "Link-up flooding trigger.";  
          }  
          enum link-down {  
            description  
              "Link-down flooding trigger.";  
          }  
          enum threshold-up {  
            description  
              "Bandwidth reservation up threshold.";  
          }  
          enum threshold-down {  
            description  
              "Bandwidth reservation down threshold.";  
          }  
          enum bandwidth-change {  
            description  
              "Bandwidth capacity change.";  
          }  
        }  
      }  
    }  
  }
```

```
        enum user-initiated {
            description
                "Initiated by user.";
        }
        enum srlg-change {
            description
                "SRLG property change.";
        }
        enum periodic-timer {
            description
                "Periodic timer expired.";
        }
    }
    default "periodic-timer";
    description
        "Trigger for the last flood.";
}
list advertised-level-areas {
    key "level-area";
    description
        "List of level-areas that the TE interface is advertised
        in.";
    leaf level-area {
        type uint32;
        description
            "The IGP area or level where the TE interface link state
            is advertised in.";
    }
}
}
}

/* TE interface attributes grouping */

grouping te-attributes {
    description
        "TE attributes configuration grouping.";
    uses te-metric-config;
    uses te-admin-groups-config;
    uses te-srlgs-config;
    uses te-igp-flooding-bandwidth-config;
    uses te-switching-cap-config;
    container state {
        config false;
        description
            "State parameters for interface TE metric.";
        uses te-advertisements-state;
    }
}
```

```
}

grouping te-all-attributes {
  description
    "TE attributes configuration grouping for all
    interfaces.";
  uses te-igp-flooding-bandwidth-config;
}

/** End of TE interfaces device groupings */
/**
 * TE device augmentations
 */

augment "/te:te" {
  description
    "TE global container.";
  /* TE Interface Configuration Data */
  uses interfaces-grouping;
  container performance-thresholds {
    description
      "Performance parameters configurable thresholds.";
  }
}

/* TE globals device augmentation */

augment "/te:te/te:globals" {
  description
    "Global TE device specific configuration parameters.";
  uses lsp-device-timers;
}

/* TE tunnels device configuration augmentation */

augment "/te:te/te:tunnels/te:tunnel" {
  description
    "Tunnel device dependent augmentation.";
  leaf path-invalidation-action {
    type identityref {
      base te-types:path-invalidation-action-type;
    }
    description
      "Tunnel path invalidation action.";
  }
  uses lsp-device-timers;
}
```

```

/* TE LSPs device state augmentation */

augment "/te:te/te:lsps/te:lsp" {
  description
    "TE LSP device dependent augmentation.";
  uses lsp-device-info;
}

/* TE interfaces RPCs/execution Data */

rpc link-state-update {
  description
    "Triggers a link state update for the specific interface.";
  input {
    choice filter-type {
      mandatory true;
      description
        "Filter choice.";
      case match-all {
        leaf all {
          type empty;
          mandatory true;
          description
            "Match all TE interfaces.";
        }
      }
      case match-one-interface {
        leaf interface {
          type if:interface-ref;
          description
            "Match a specific TE interface.";
        }
      }
    }
  }
}
}

<CODE ENDS>

```

Figure 12: TE device data model YANG module

## 7. Notifications

Notifications are a key component of any topology data model.

[RFC8639] and [RFC8641] define a subscription mechanism and a push mechanism for YANG datastores. These mechanisms currently allow the user to:

- \* Subscribe to notifications on a per-client basis.
- \* Specify subtree filters or XML Path Language (XPath) filters so that only contents of interest will be sent.
- \* Specify either periodic or on-demand notifications.

## 8. TE Generic and Helper YANG Modules

## 9. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made.

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

URI: urn:ietf:params:xml:ns:yang:ietf-te-device  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

Name: ietf-te  
Namespace: urn:ietf:params:xml:ns:yang:ietf-te  
Prefix: te  
Reference: RFCXXXX

Name: ietf-te-device  
Namespace: urn:ietf:params:xml:ns:yang:ietf-te-device  
Prefix: te-device  
Reference: RFCXXXX

## 10. Security Considerations

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

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

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

`"/te/globals"`: This module specifies the global TE configurations on a device. Unauthorized access to this container could cause the device to ignore packets it should receive and process.

`"/te/tunnels"`: This list specifies the configuration and state of TE Tunnels present on the device or controller. Unauthorized access to this list could cause the device to ignore packets it should receive and process. An attacker may also use state to derive information about the network topology, and subsequently orchestrate further attacks.

`"/te/interfaces"`: This list specifies the configuration and state TE interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

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:

`"/te/lspss"`: this list contains information state about established LSPs in the network. An attacker can use this information to derive information about the network topology, and subsequently orchestrate further attacks.

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:

`"/te/tunnels-actions"`: using this RPC, an attacker can modify existing paths that may be carrying live traffic, and hence result to interruption to services carried over the network.

`"/te/tunnels-path-compute"`: using this RPC, an attacker can retrieve secured information about the network provider which can be used to orchestrate further attacks.

The security considerations spelled out in the YANG 1.1 specification [RFC7950] apply for this document as well.

## 11. Acknowledgement

The authors would like to thank the members of the multi-vendor YANG design team who are involved in the definition of this model.

The authors would like to thank Tom Petch for reviewing and providing useful feedback about the document. The authors would also like to thank Loa Andersson, Lou Berger, Sergio Belotti, Italo Busi, Carlo Perocchio, Francesco Lazzeri, Aihua Guo, Dhruv Dhody, and Raqib Jones for providing useful feedback on this document.

## 12. Contributors

Himanshu Shah  
Ciena

Email: [hshah@ciena.com](mailto:hshah@ciena.com)

Xia Chen  
Huawei Technologies

Email: [jescia.chenxia@huawei.com](mailto:jescia.chenxia@huawei.com)

Bin Wen  
Comcast

Email: [Bin\\_Wen@cable.comcast.com](mailto:Bin_Wen@cable.comcast.com)

## 13. Appendix A: Data Tree Examples

This section contains examples of use of the model with RESTCONF [RFC8040] and JSON encoding.

For the example we will use a 4 node MPLS network where RSVP-TE MPLS Tunnels can be setup. The loopbacks of each router are shown. The network in Figure 13 will be used in the examples described in the following sections.

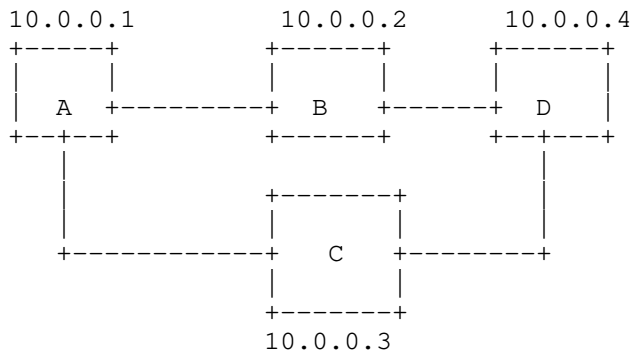


Figure 13: TE network used in data tree examples

### 13.1. Basic Tunnel Setup

This example uses the TE Tunnel YANG data model defined in this document to create an RSVP-TE signaled Tunnel of packet LSP encoding type. First, the TE Tunnel is created with no specific restrictions or constraints (e.g., protection or restoration). The TE Tunnel ingresses on router A and egresses on router D.

In this case, the TE Tunnel is created without specifying additional information about the primary paths.

```

POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
Host: example.com
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "ietf-te:tunnel": [
    {
      "name": "Example_LSP_Tunnel_A_2",
      "encoding": "te-types:lsp-encoding-packet",
      "admin-state": "te-types:tunnel-state-up",
      "source": "10.0.0.1",
      "destination": "10.0.0.4",
      "bidirectional": "false",
      "signaling-type": "te-types:path-setup-rsvp"
    }
  ]
}

```



### 13.2. Global Named Path Constraints

This example uses the YANG data model to create a 'named path constraint' that can be reference by TE Tunnels. The path constraint, in this case, limits the TE Tunnel hops for the computed path.

```
POST /restconf/data/ietf-te:te/globals/named-path-constraints HTTP/1.1
```

```
Host: example.com
```

```
Accept: application/yang-data+json
```

```
Content-Type: application/yang-data+json
```

```
{
  "ietf-te:named-path-constraint": {
    "name": "max-hop-3",
    "path-metric-bounds": {
      "path-metric-bound": {
        "metric-type": "te-types:path-metric-hop",
        "upper-bound": "3"
      }
    }
  }
}
```

### 13.3. Tunnel with Global Path Constraint

In this example, the previously created 'named path constraint' is applied to the TE Tunnel created in Section 13.1.

```
POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
Host: example.com
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "ietf-te:ietf-tunnel": [
    {
      "name": "Example_LSP_Tunnel_A_4_1",
      "encoding": "te-types:lsp-encoding-packet",
      "description": "Simple_LSP_with_named_path",
      "admin-state": "te-types:tunnel-state-up",
      "source": "10.0.0.1",
      "destination": "10.0.0.4",
      "signaling-type": "path-setup-rsvp",
      "bidirectional": "false",
      "primary-paths": [
        {
          "primary-path": {
            "name": "Simple_LSP_1",
            "use-path-computation": "true",
            "named-path-constraint": "max-hop-3"
          }
        }
      ]
    }
  ]
}
```

#### 13.4. Tunnel with Per-tunnel Path Constraint

In this example, the a per tunnel path constraint is explicitly indicated under the TE Tunnel created in Section 13.1 to constrain the computed path for the tunnel.

```
POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
Host: example.com
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "ietf-te:tunnel": [
    {
      "name": "Example_LSP_Tunnel_A_4_2",
      "encoding": "te-types:lsp-encoding-packet",
      "admin-state": "te-types:tunnel-state-up",
      "source": "10.0.0.1",
      "destination": "10.0.0.4",
      "bidirectional": "false",
      "signaling-type": "te-types:path-setup-rsvp",
      "primary-paths": {
        "primary-path": [
          {
            "name": "path1",
            "path-metric-bounds": {
              "path-metric-bound": [
                {
                  "metric-type": "te-types:path-metric-hop",
                  "upper-bound": "3"
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```

### 13.5. Tunnel State

In this example, the 'GET' query is sent to return the state stored about the tunnel.

```
GET /restconf/data/ietf-te:te/tunnels/tunnel="Example_LSP_Tunnel_A_4_1"
/p2p-primary-paths/ HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

The request, with status code 200 would include, for example, the following json:

```

{
  "ietf-te:primary-paths": {
    "primary-path": [
      {
        "name": "path1",
        "path-computation-method": "te-types:path-locally-computed",
        "computed-paths-properties": {
          "computed-path-properties": [
            {
              "k-index": "1",
              "path-properties": {
                "path-route-objects": {
                  "path-route-object": [
                    {
                      "index": "1",
                      "numbered-node-hop": {
                        "node-id": "10.0.0.2"
                      }
                    },
                    {
                      "index": "2",
                      "numbered-node-hop": {
                        "node-id": "10.0.0.4"
                      }
                    }
                  ]
                }
              }
            }
          ]
        }
      }
    ]
  },
  "lsp": {
    "lsp": [
      {
        "tunnel-name": "Example_LSP_Tunnel_A_4_1",
        "node": "10.0.0.1 ",
        "lsp-id": "25356"
      }
    ]
  }
}

```

## 14. References

### 14.1. 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>>.
- [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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3945] Mannie, E., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, DOI 10.17487/RFC3945, October 2004, <<https://www.rfc-editor.org/info/rfc3945>>.
- [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, DOI 10.17487/RFC4206, October 2005, <<https://www.rfc-editor.org/info/rfc4206>>.
- [RFC4872] Lang, J.P., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <<https://www.rfc-editor.org/info/rfc4872>>.
- [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>>.
- [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>>.

- [RFC6107] Shiomoto, K., Ed. and A. Farrel, Ed., "Procedures for Dynamically Signaled Hierarchical Label Switched Paths", RFC 6107, DOI 10.17487/RFC6107, February 2011, <<https://www.rfc-editor.org/info/rfc6107>>.
- [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>>.
- [RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP ASSOCIATION Object Extensions", RFC 6780, DOI 10.17487/RFC6780, October 2012, <<https://www.rfc-editor.org/info/rfc6780>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <<https://www.rfc-editor.org/info/rfc7308>>.
- [RFC7551] Zhang, F., Ed., Jing, R., and R. Gandhi, Ed., "RSVP-TE Extensions for Associated Bidirectional Label Switched Paths (LSPs)", RFC 7551, DOI 10.17487/RFC7551, May 2015, <<https://www.rfc-editor.org/info/rfc7551>>.
- [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>>.
- [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>>.

- [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>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [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>>.
- [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>>.
- [RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <<https://www.rfc-editor.org/info/rfc8639>>.
- [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <<https://www.rfc-editor.org/info/rfc8641>>.
- [RFC8776] Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "Common YANG Data Types for Traffic Engineering", RFC 8776, DOI 10.17487/RFC8776, June 2020, <<https://www.rfc-editor.org/info/rfc8776>>.
- [RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Gonzalez de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", RFC 8795, DOI 10.17487/RFC8795, August 2020, <<https://www.rfc-editor.org/info/rfc8795>>.

#### 14.2. Informative References

- [I-D.ietf-spring-segment-routing-policy]  
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and  
P. Mattes, "Segment Routing Policy Architecture", Work in  
Progress, Internet-Draft, draft-ietf-spring-segment-  
routing-policy-16, 28 January 2022,  
<[https://www.ietf.org/archive/id/draft-ietf-spring-  
segment-routing-policy-16.txt](https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-policy-16.txt)>.
- [I-D.ietf-teas-yang-rsvp]  
Beeram, V. P., Saad, T., Gandhi, R., Liu, X., and I.  
Bryskin, "A YANG Data Model for Resource Reservation  
Protocol (RSVP)", Work in Progress, Internet-Draft, draft-  
ietf-teas-yang-rsvp-17, 9 January 2022,  
<[https://www.ietf.org/archive/id/draft-ietf-teas-yang-  
rsvp-17.txt](https://www.ietf.org/archive/id/draft-ietf-teas-yang-rsvp-17.txt)>.
- [RFC4427] Mannie, E., Ed. and D. Papadimitriou, Ed., "Recovery  
(Protection and Restoration) Terminology for Generalized  
Multi-Protocol Label Switching (GMPLS)", RFC 4427,  
DOI 10.17487/RFC4427, March 2006,  
<<https://www.rfc-editor.org/info/rfc4427>>.
- [RFC8800] Litkowski, S., Sivabalan, S., Barth, C., and M. Negi,  
"Path Computation Element Communication Protocol (PCEP)  
Extension for Label Switched Path (LSP) Diversity  
Constraint Signaling", RFC 8800, DOI 10.17487/RFC8800,  
July 2020, <<https://www.rfc-editor.org/info/rfc8800>>.
- [RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder,  
"The BGP Tunnel Encapsulation Attribute", RFC 9012,  
DOI 10.17487/RFC9012, April 2021,  
<<https://www.rfc-editor.org/info/rfc9012>>.

#### Authors' Addresses

Tarek Saad  
Juniper Networks  
  
Email: [tsaad@juniper.net](mailto:tsaad@juniper.net)

Rakesh Gandhi  
Cisco Systems Inc  
  
Email: [rgandhi@cisco.com](mailto:rgandhi@cisco.com)



Xufeng Liu  
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Vishnu Pavan Beeram  
Juniper Networks

Email: vbeeram@juniper.net

Igor Bryskin  
Individual

Email: i\_bryskin@yahoo.com

Oscar Gonzalez de Dios  
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

TEAS Working Group  
Internet Draft  
Intended status: Standards Track

Xufeng Liu  
Volta Networks  
Igor Bryskin  
Huawei Technologies  
Vishnu Pavan Beeram  
Tarek Saad  
Juniper Networks  
Himanshu Shah  
Ciena  
Oscar Gonzalez De Dios  
Telefonica

Expires: December 19, 2019

June 19, 2019

YANG Data Model for Traffic Engineering (TE) Topologies  
draft-ietf-teas-yang-te-topo-22

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on December 19, 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

(<http://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.

## Abstract

This document defines a YANG data model for representing, retrieving and manipulating Traffic Engineering (TE) Topologies. The model serves as a base model that other technology specific TE Topology models can augment.

## Conventions used in this document

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.

## Table of Contents

1. Introduction.....	3
1.1. Terminology.....	4
1.2. Tree Structure.....	4
1.3. Prefixes in Data Node Names.....	5
2. Characterizing TE Topologies.....	5
3. Modeling Abstractions and Transformations.....	7
3.1. TE Topology.....	7
3.2. TE Node.....	7
3.3. TE Link.....	8
3.4. Transitional TE Link for Multi-Layer Topologies.....	8
3.5. TE Link Termination Point (LTP).....	10
3.6. TE Tunnel Termination Point (TTP).....	10
3.7. TE Node Connectivity Matrix.....	11
3.8. TTP Local Link Connectivity List (LLCL).....	11
3.9. TE Path.....	11
3.10. TE Inter-Layer Lock.....	12
3.11. Underlay TE topology.....	13
3.12. Overlay TE topology.....	13
3.13. Abstract TE topology.....	13
4. Model Applicability.....	14
4.1. Native TE Topologies.....	14

4.2. Customized TE Topologies.....	16
4.3. Merging TE Topologies Provided by Multiple Providers.....	19
4.4. Dealing with Multiple Abstract TE Topologies Provided by the Same Provider.....	22
5. Modeling Considerations.....	25
5.1. Network topology building blocks.....	25
5.2. Technology agnostic TE Topology model.....	25
5.3. Model Structure.....	26
5.4. Topology Identifiers.....	27
5.5. Generic TE Link Attributes.....	27
5.6. Generic TE Node Attributes.....	28
5.7. TED Information Sources.....	29
5.8. Overlay/Underlay Relationship.....	30
5.9. Templates.....	31
5.10. Scheduling Parameters.....	32
5.11. Notifications.....	33
6. Guidance for Writing Technology Specific TE Topology Augmentations .....	33
7. TE Topology YANG Module.....	46
8. Security Considerations.....	92
9. IANA Considerations.....	94
10. References.....	95
10.1. Normative References.....	95
10.2. Informative References.....	96
11. Acknowledgments.....	100
Appendix A. Complete Model Tree Structure.....	101
Appendix B. Companion YANG Model for Non-NMDA Compliant Implementations.....	163
Appendix C. Example: YANG Model for Technology Specific Augmentations .....	172
Contributors.....	210
Authors' Addresses.....	210

## 1. Introduction

The Traffic Engineering Database (TED) is an essential component of Traffic Engineered (TE) systems that are based on MPLS-TE [RFC2702] and GMPLS [RFC3945]. The TED is a collection of all TE information about all TE nodes and TE links in the network. The TE Topology is a schematic arrangement of TE nodes and TE links present in a given TED. There could be one or more TE Topologies present in a given Traffic Engineered system. A TE Topology is the topology on which path computational algorithms are run to compute Traffic Engineered Paths (TE Paths).

This document defines a YANG [RFC7950] data model for representing and manipulating TE Topologies. This model contains technology

agnostic TE Topology building blocks that can be augmented and used by other technology-specific TE Topology models.

### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The reader is assumed to be familiar with general body of work captured in currently available TE related RFCs. [RFC7926] serves as a good starting point for those who may be less familiar with Traffic Engineering related RFCs.

Some of the key terms used in this document are:

**TED:** The Traffic Engineering Database is a collection of all TE information about all TE nodes and TE links in a given network.

**TE-Topology:** The TE Topology is a schematic arrangement of TE nodes and TE links in a given TED. It forms the basis for a graph suitable for TE path computations.

**Native TE Topology:** Native TE Topology is a topology that is native to a given provider network. Native TE topology could be discovered via various routing protocols and/or subscribe/publish techniques. This is the topology on which path computational algorithms are run to compute TE Paths.

**Customized TE Topology:** Customized TE Topology is a custom topology that is produced by a provider for a given client. This topology typically makes abstractions on the provider's Native TE Topology, and is provided to the client. The client receives the Customized TE Topology, and merges it into the client's Native TE Topology. The client's path computational algorithms aren't typically run on the Customized TE Topology; they are run on the client's Native TE Topology after the merge.

### 1.2. Tree Structure

A simplified graphical representation of the data model is presented in Appendix A. of this document. The tree format defined in [RFC8340] is used for the YANG data model tree representation.

### 1.3. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
nw	ietf-network	[RFC6991]
nt	ietf-network-topology	[RFC8345]
te-types	ietf-te-types	[I-D.ietf-teas-yang-te-types]

Table 1: Prefixes and corresponding YANG modules

## 2. Characterizing TE Topologies

The data model proposed by this document takes the following characteristics of TE Topologies into account:

- TE Topology is an abstract control-plane representation of the data-plane topology. Hence attributes specific to the data-plane must make their way into the corresponding TE Topology modeling. The TE Topology comprises of dynamic auto-discovered data as well as fairly static data associated with data-plane nodes and links. The dynamic data may change frequently, such as unreserved bandwidth available on data-plane links. The static data rarely changes, such as layer network identification, switching and adaptation capabilities and limitations, fate sharing, and administrative colors. It is possible for a single TE Topology to encompass TE information at multiple switching layers.
- TE Topologies are protocol independent. Information about topological elements may be learnt via link-state protocols, but the topology can exist without being dependent on any particular protocol.
- TE Topology may not be congruent to the routing topology in a given TE System. The routing topology is constructed based on routing adjacencies. There isn't always a one-to-one association between a TE-link and a routing adjacency. For example, the presence of a TE link between a pair of nodes doesn't necessarily imply the existence of a routing-adjacency between these nodes. To

learn more, see [I-D.ietf-teas-te-topo-and-tunnel-modeling] and [I-D.ietf-teas-yang-l3-te-topo].

- Each TE Topological element has at least one information source associated with it. In some scenarios, there could be more than one information source associated with any given topological element.
- TE Topologies can be hierarchical. Each node and link of a given TE Topology can be associated with respective underlay topology. This means that each node and link of a given TE Topology can be associated with an independent stack of supporting TE Topologies.
- TE Topologies can be customized. TE topologies of a given network presented by the network provider to its client could be customized on per-client request basis. This customization could be performed by provider, by client or by provider/client negotiation. The relationship between a customized topology and provider's native topology could be captured as hierarchical (overlay-underlay), but otherwise the two topologies are decoupled from each other. A customized topology is presented to the client, while provider's native topology is known in its entirety to the provider itself.

### 3. Modeling Abstractions and Transformations

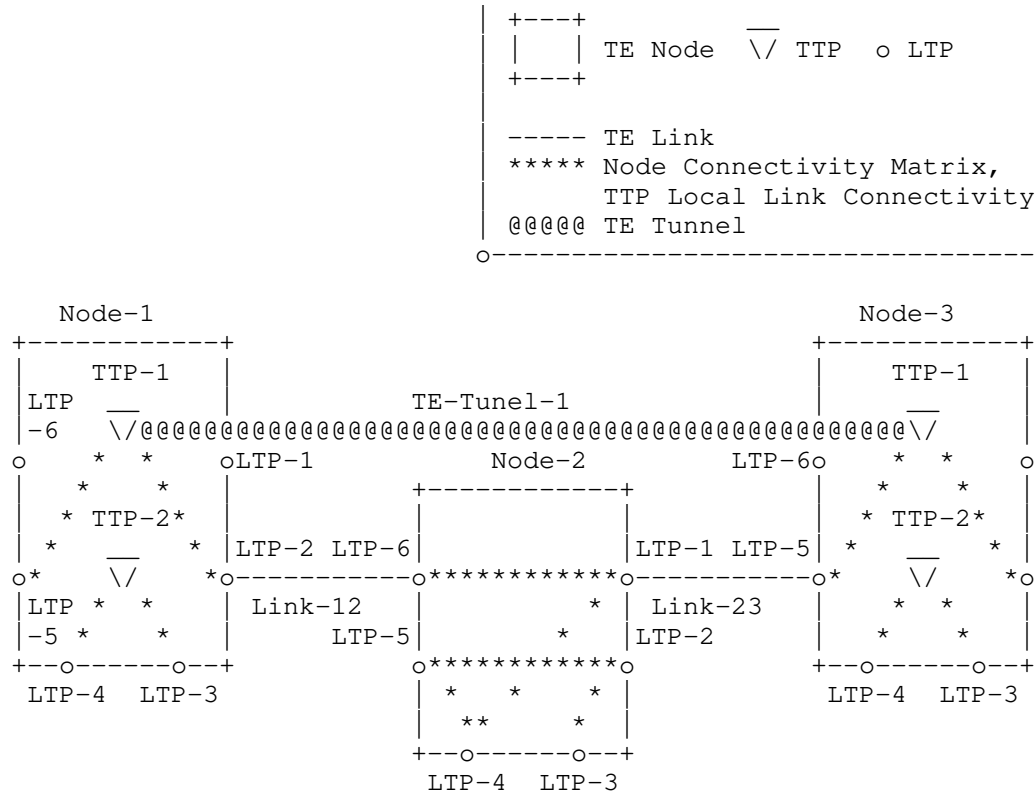


Figure 1: TE Topology Modeling Abstractions

### 3.1. TE Topology

TE topology is a traffic engineering representation of one or more layers of network topologies. TE topology is comprised of TE nodes (TE graph vertices) interconnected via TE links (TE graph edges). A TE topology is mapped to a TE graph.

### 3.2. TE Node

TE node is an element of a TE topology, presented as a vertex on TE graph. TE node represents one or several nodes, or a fraction of a node, which can be a switch or router that is physical or virtual. TE node belongs to and is fully defined in exactly one TE topology. TE node is assigned a unique ID within the TE topology scope. TE node attributes include information related to the data plane aspects of



the associated node(s) (e.g. connectivity matrix), as well as configuration data (such as TE node name). A given TE node can be reached on the TE graph over one of TE links terminated by the TE node.

Multi-layer TE nodes providing switching functions at multiple network layers are an example where a physical node can be decomposed into multiple logical TE nodes, which are fractions of the physical node. Some of these (logical) TE nodes may reside in the client layer TE topology while the remaining TE nodes belong to the server layer TE topology.

In Figure 1, Node-1, Node-2, and Node-3 are TE nodes.

### 3.3. TE Link

TE link is an element of a TE topology, presented as an edge on TE graph. The arrows on an edge indicate one or both directions of the TE link. When there are a pair of parallel links of opposite directions, an edge without arrows is also used. TE link represents one or several (physical) links or a fraction of a link. TE link belongs to and is fully defined in exactly one TE topology. TE link is assigned a unique ID within the TE topology scope. TE link attributes include parameters related to the data plane aspects of the associated link(s) (e.g. unreserved bandwidth, resource maps/pools, etc.), as well as the configuration data (such as remote node/link IDs, SRLGs, administrative colors, etc.). TE link is connected to TE node, terminating the TE link via exactly one TE link termination point (LTP).

In Figure 1, Link-12 and Link-23 are TE links.

### 3.4. Transitional TE Link for Multi-Layer Topologies

Networks are typically composed of multiple network layers where one or multiple signals in the client layer network can be multiplexed and encapsulated into a server layer signal [RFC5212] [G.805]. The server layer signal can be carried in the server layer network across multiple nodes until the server layer signal is terminated and the client layer signals reappear in the node that terminates the server layer signal. Examples of multi-layer networks are: IP over MPLS over Ethernet, low order Optical Data Unit-k (ODUk) signals multiplexed into a high order ODUL (l>k) carried over an Optical Channel (OCh) signal in an optical transport network as defined in [G.872] and [G.709].

TE links as defined in Section 3.3. can be used to represent links within a network layer. In case of a multi-layer network, TE nodes and TE links only allow representation of each network layer as a separate TE topology. Each of these single layer TE topologies would be isolated from their client and their server layer TE topology, if present. The highest and the lowest network layer in the hierarchy only have a single adjacent layer below or above, respectively. Multiplexing of client layer signals and encapsulating them into a server layer signal requires a function that is provided inside a node (typically realized in hardware). This function is also called layer transition.

One of the key requirements for path computation is to be able to calculate a path between two endpoints across a multi-layer network based on the TE topology representing this multi-layer network. This means that an additional TE construct is needed that represents potential layer transitions in the multi-layer TE-topology that connects the TE-topologies representing each separate network layer. The so-called transitional TE link is such a construct and it represents the layer transition function residing inside a node that is decomposed into multiple logical nodes that are represented as TE nodes (see also the transitional link definition in [G.8080] for the optical transport network). Hence, a transitional TE link connects a client layer node with a server layer node. A TE link as defined in 3.3. has LTPs of exactly the same kind on each link end whereas the transitional TE link has client layer LTPs on the client side of the transitional link and in most cases a single server layer LTP on the server side. It should be noted that transitional links are a helper construct in the multi-layer TE topology and they only exist as long as they are not in use, as they represent potential connectivity. When the server layer trail has been established between the server layer LTP of two transitional links in the server layer network, the resulting client layer link in the data plane will be represented as a normal TE link in the client layer topology. The transitional TE links will re-appear when the server layer trail has been torn down.

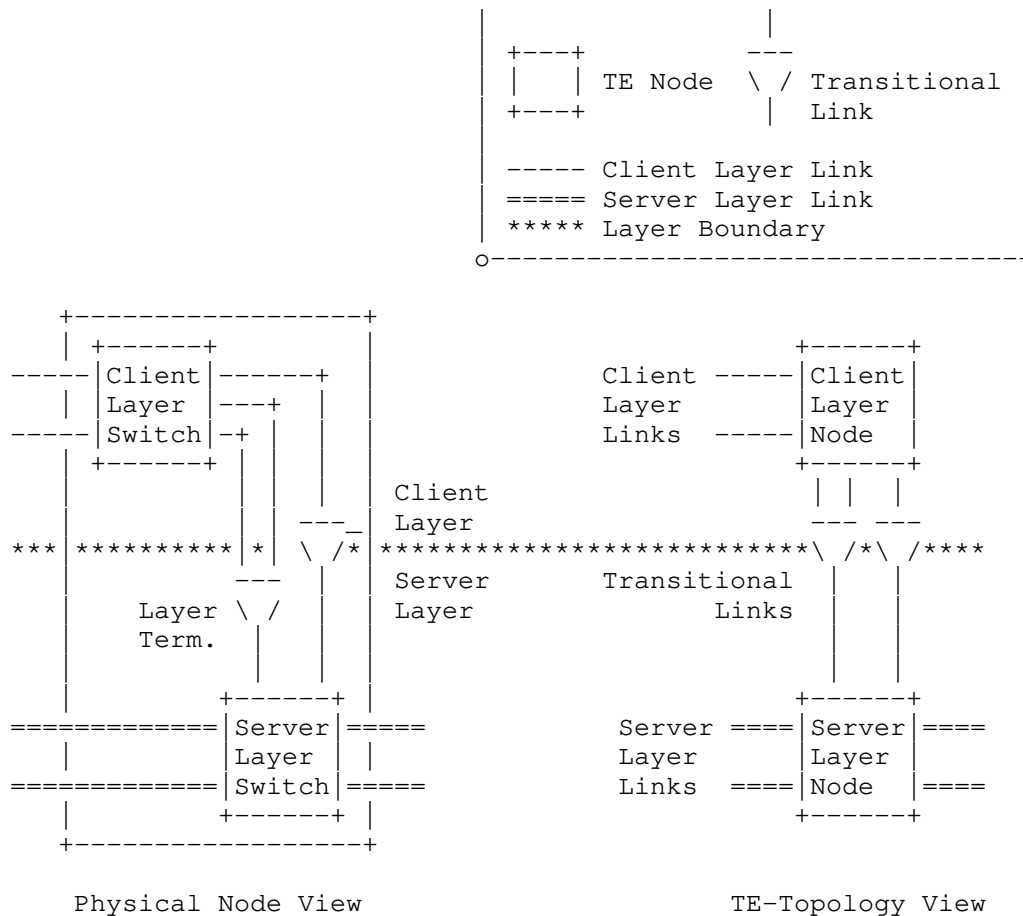


Figure 2: Modeling a Multi-Layer Node (Dual-Layer Example)

### 3.5. TE Link Termination Point (LTP)

TE link termination point (LTP) is a conceptual point of connection of a TE node to one of the TE links, terminated by the TE node. Cardinality between an LTP and the associated TE link is 1:0..1.

In Figure 1, Node-2 has six LTPs: LTP-1 to LTP-6.

### 3.6. TE Tunnel Termination Point (TTP)

TE tunnel termination point (TTP) is an element of TE topology representing one or several of potential transport service termination points (i.e. service client adaptation points such as

WDM/OCh transponder). TTP is associated with (hosted by) exactly one TE node. TTP is assigned a unique ID within the TE node scope. Depending on the TE node's internal constraints, a given TTP hosted by the TE node could be accessed via one, several or all TE links terminated by the TE node.

In Figure 1, Node-1 has two TTPs: TTP-1 and TTP-2.

### 3.7. TE Node Connectivity Matrix

TE node connectivity matrix is a TE node's attribute describing the TE node's switching limitations in a form of valid switching combinations of the TE node's LTPs (see below). From the point of view of a potential TE path arriving at the TE node at a given inbound LTP, the node's connectivity matrix describes valid (permissible) outbound LTPs for the TE path to leave the TE node from.

In Figure 1, the connectivity matrix on Node-2 is:  
{<LTP-6, LTP-1>, <LTP-5, LTP-2>, <LTP-5, LTP-4>, <LTP-4, LTP-1>, <LTP-3, LTP-2>}

### 3.8. TTP Local Link Connectivity List (LLCL)

TTP Local Link Connectivity List (LLCL) is a List of TE links terminated by the TTP hosting TE node (i.e. list of the TE link LTPs), which the TTP could be connected to. From the point of view of a potential TE path, LLCL provides a list of valid TE links the TE path needs to start/stop on for the connection, taking the TE path, to be successfully terminated on the TTP in question.

In Figure 1, the LLCL on Node-1 is:  
{<TTP-1, LTP-5>, <TTP-1, LTP-2>, <TTP-2, LTP-3>, <TTP-2, LTP4>}

### 3.9. TE Path

TE path is an ordered list of TE links and/or TE nodes on the TE topology graph, inter-connecting a pair of TTPs to be taken by a potential connection. TE paths, for example, could be a product of successful path computation performed for a given transport service.

In Figure 1, the TE Path for TE-Tunnel-1 is:  
{Node-1:TTP-1, Link-12, Node-2, Link-23, Node-3:TTP1}

## 3.10. TE Inter-Layer Lock

TE inter-layer lock is a modeling concept describing client-server layer adaptation relationships and hence important for the multi-layer traffic engineering. It is an association of M client layer LTPs and N server layer TTPs, within which data arriving at any of the client layer LTPs could be adopted onto any of the server layer TTPs. TE inter-layer lock is identified by inter-layer lock ID, which is unique across all TE topologies provided by the same provider. The client layer LTPs and the server layer TTPs associated within a given TE inter-layer lock are annotated with the same inter-layer lock ID attribute.

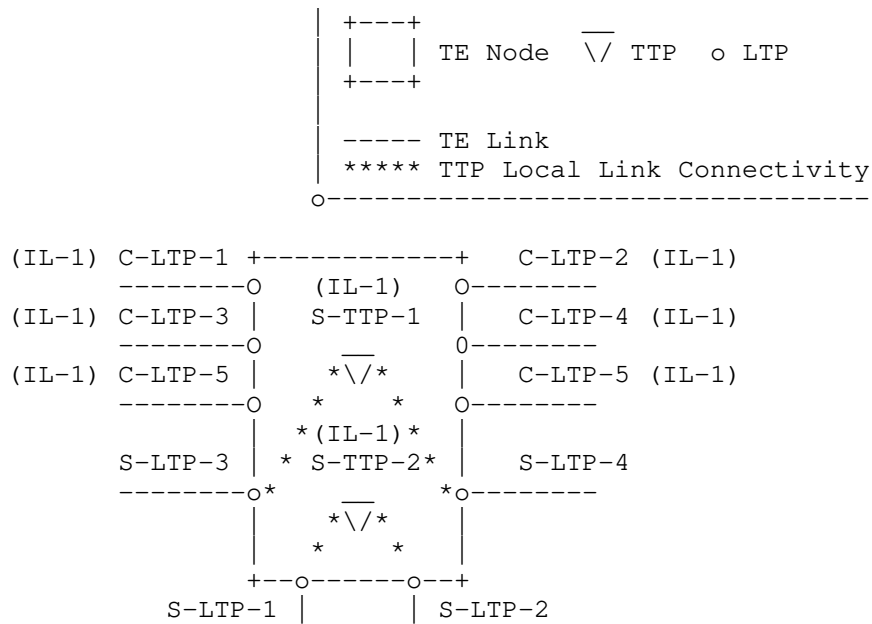


Figure 3: TE Inter-Layer Lock ID Associations

On the picture above a TE inter-layer lock with IL\_1 ID associates 6 client layer LTPs (C-LTP-1 - C-LTP-6) with two server layer TTPs (S-TTP-1 and S-TTP-2). They all have the same attribute - TE inter-layer lock ID: IL-1, which is the only thing that indicates the association. A given LTP may have 0, 1 or more inter-layer lock IDs. In the latter case this means that the data arriving at the LTP may be adopted onto any of TTPs associated with all specified inter-layer locks. For example, C-LTP-1 could have two inter-layer lock IDs - IL-1 and IL-2. This would mean that C-LTP-1 for adaptation purposes could use not just TTPs associated with inter-layer lock IL-1 (i.e.

S-TTP-1 and S-TTP-2 on the picture), but any of TTPs associated with inter-layer lock IL-2 as well. Likewise, a given TTP may have one or more inter-layer lock IDs, meaning that it can offer the adaptation service to any of client layer LTPs with inter-layer lock ID matching one of its own. Additionally, each TTP has an attribute - Unreserved Adaptation Bandwidth, which announces its remaining adaptation resources sharable between all potential client LTPs.

LTPs and TTPs associated within the same TE inter-layer lock may be hosted by the same (hybrid, multi-layer) TE node or multiple TE nodes located in the same or separate TE topologies. The latter is especially important since TE topologies of different layer networks could be modeled by separate augmentations of the basic (common to all layers) TE topology model.

### 3.11. Underlay TE topology

Underlay TE topology is a TE topology that serves as a base for constructing of overlay TE topologies

### 3.12. Overlay TE topology

Overlay TE topology is a TE topology constructed based on one or more underlay TE topologies. Each TE node of the overlay TE topology represents an arbitrary segment of an underlay TE topology; each TE link of the overlay TE topology represents an arbitrary TE path in one of the underlay TE topologies. The overlay TE topology and the supporting underlay TE topologies may represent distinct layer networks (e.g. OTN/ODUk and WDM/OCh respectively) or the same layer network.

### 3.13. Abstract TE topology

Abstract TE topology is a topology that contains abstract topological elements (nodes, links, tunnel termination points). Abstract TE topology is an overlay TE topology created by a topology provider and customized for a topology provider's client based on one or more of the provider's native TE topologies (underlay TE topologies), the provider's policies and the client's preferences. For example, a first level topology provider (such as Domain Controller) can create an abstract TE topology for its client (e.g. Multi-Domain Service Coordinator) based on the provider's one or more native TE topologies, local policies/profiles and the client's TE topology configuration requests

Figure 4 shows an example of abstract TE topology.

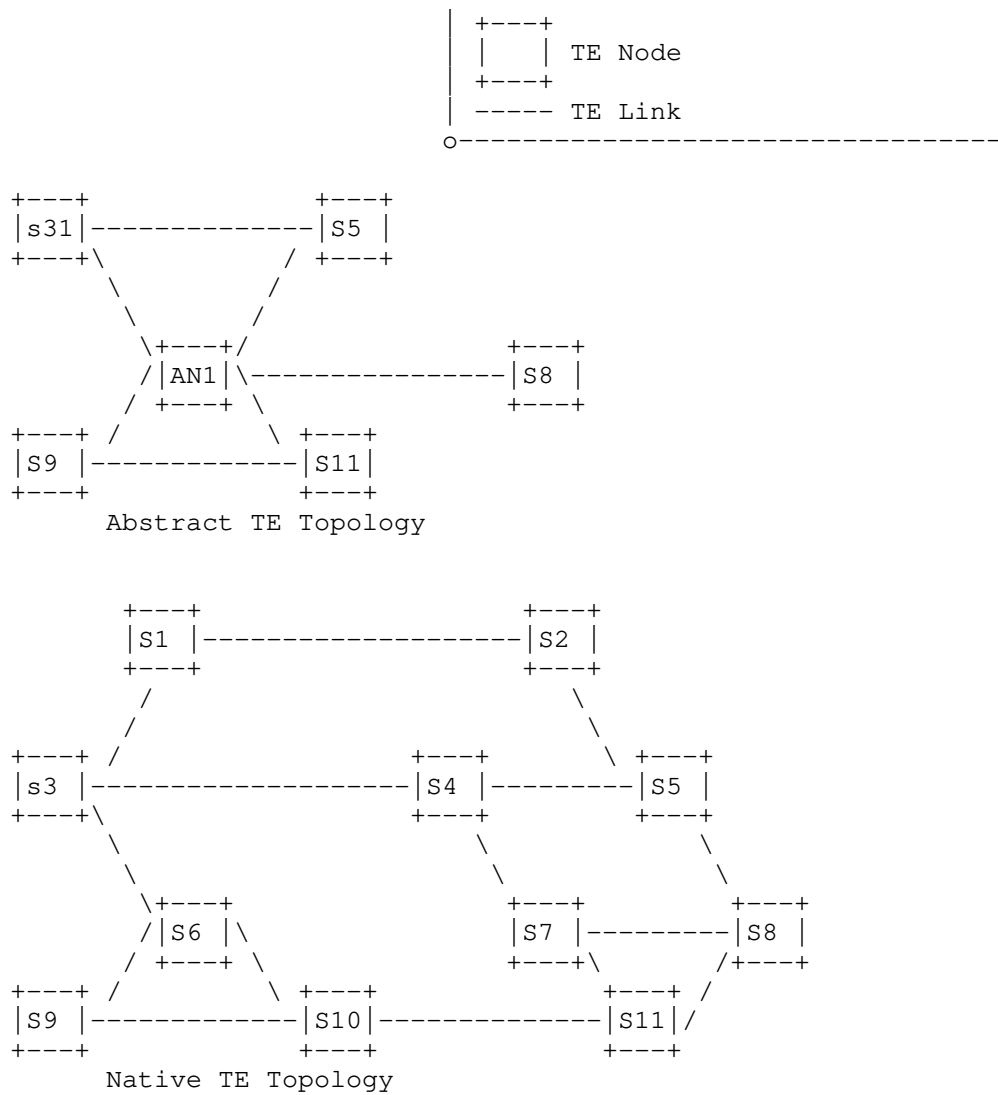


Figure 4: Abstract TE Topology

#### 4. Model Applicability

##### 4.1. Native TE Topologies

The model discussed in this draft can be used to represent and retrieve native TE topologies on a given TE system.

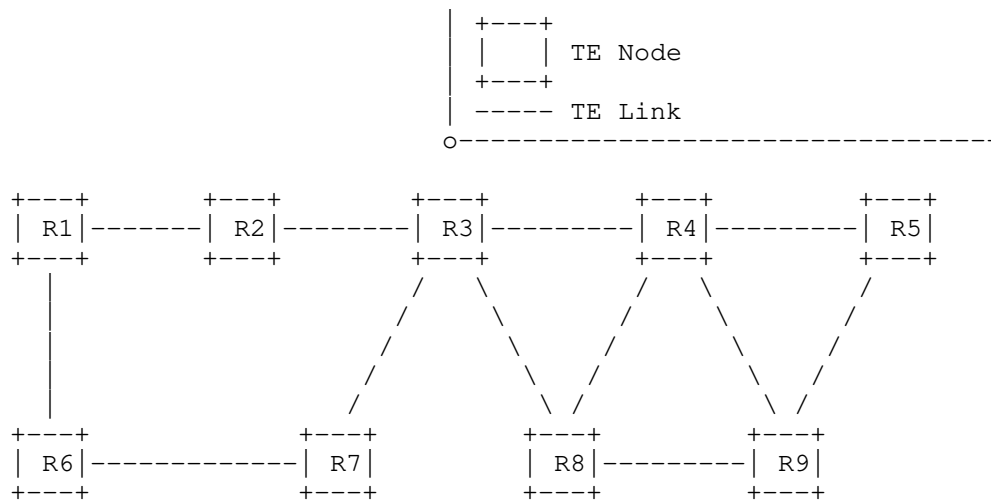


Figure 5a: Example Network Topology

Consider the network topology depicted in Figure 5a. R1 .. R9 are nodes representing routers. An implementation MAY choose to construct a native TE Topology using all nodes and links present in the given TED as depicted in Figure 5b. The data model proposed in this document can be used to retrieve/represent this TE topology.

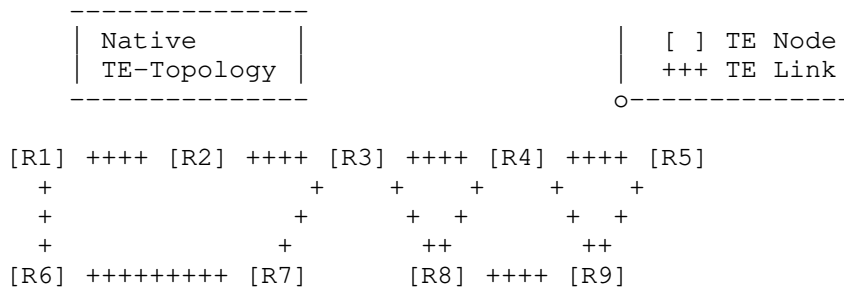


Figure 5b: Native TE Topology as seen on Node R3

Consider the case of the topology being split in a way that some nodes participate in OSPF-TE while others participate in ISIS-TE (Figure 6a). An implementation MAY choose to construct separate TE Topologies based on the information source. The native TE Topologies constructed using only nodes and links that were learnt via a specific information source are depicted in Figure 6b. The data model proposed in this document can be used to retrieve/represent these TE topologies.



Similarly, the data model can be used to represent/retrieve a TE Topology that is constructed using only nodes and links that belong to a particular technology layer. The data model is flexible enough to retrieve and represent many such native TE Topologies.

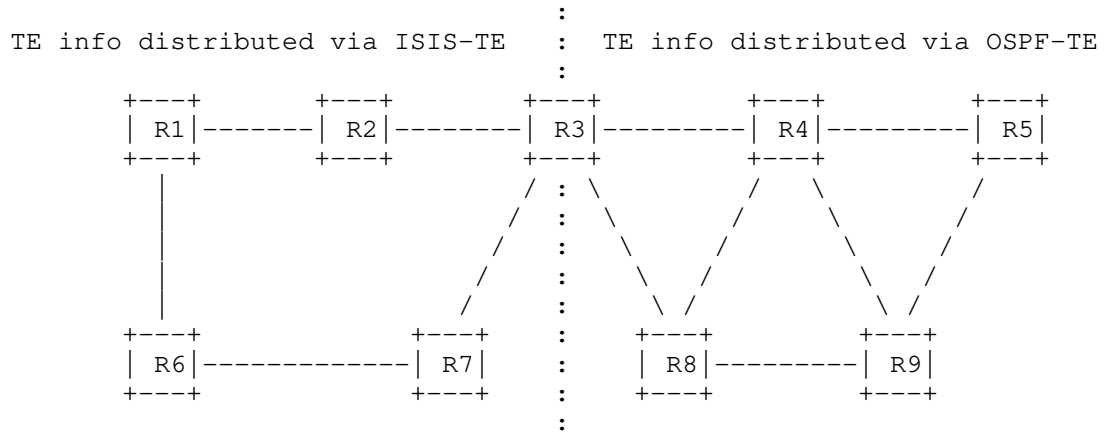


Figure 6a: Example Network Topology

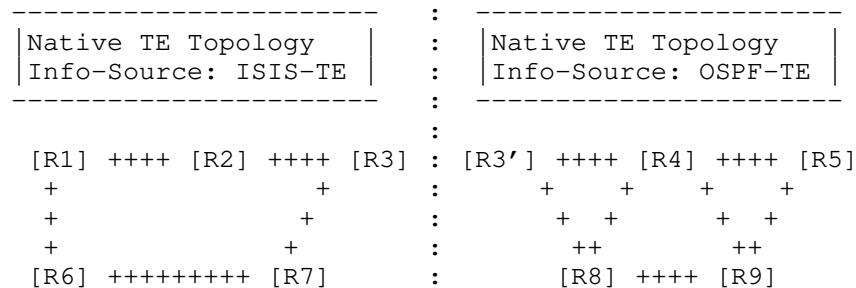


Figure 6b: Native TE Topologies as seen on Node R3

## 4.2. Customized TE Topologies

Customized TE topology is a topology that was modified by the provider to honor a particular client's requirements or preferences. The model discussed in this draft can be used to represent, retrieve and manipulate customized TE Topologies. The model allows the provider to present the network in abstract TE Terms on a per client

basis. These customized topologies contain sufficient information for the path computing client to select paths according to its policies.

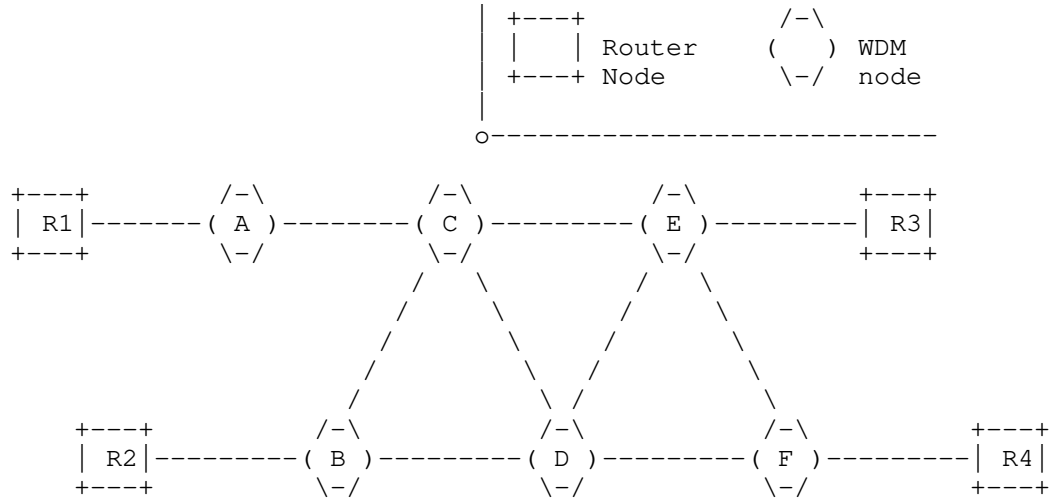


Figure 7: Example packet optical topology

Consider the network topology depicted in Figure 7. This is a typical packet optical transport deployment scenario where the WDM layer network domain serves as a Server Network Domain providing transport connectivity to the packet layer network Domain (Client Network Domain). Nodes R1, R2, R3 and R4 are IP routers that are connected to an Optical WDM transport network. A, B, C, D, E and F are WDM nodes that constitute the Server Network Domain.

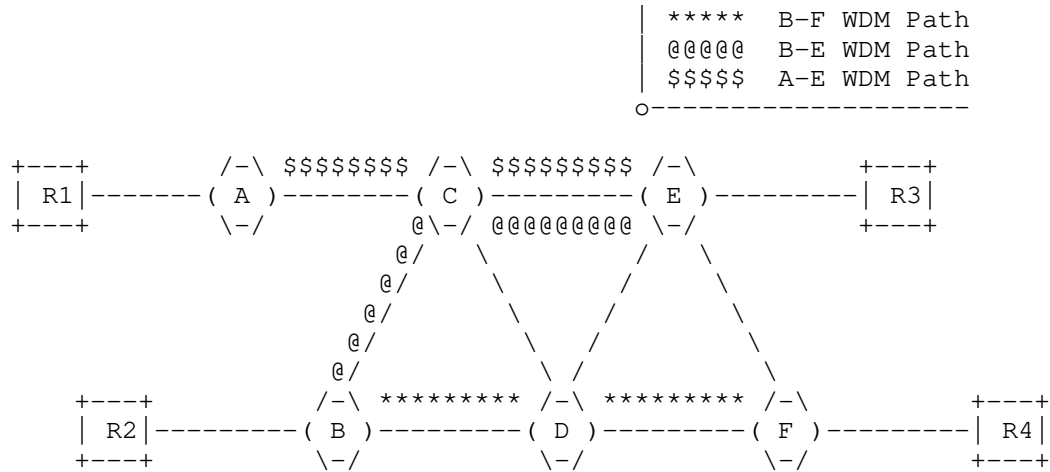


Figure 8a: Paths within the provider domain

```

+++++++ [A] ++++++ [E] ++++++
                ++++++
                ++++++
                ++++++
                ++++++
                ++++++
+++++++ [B] ++++++ [F] ++++++

```

Figure 8b: Customized TE Topology provided to the Client

The goal here is to augment the Client TE Topology with a customized TE Topology provided by the WDM network. Given the availability of the paths A-E, B-F and B-E (Figure 8a), a customized TE Topology as depicted in Figure 8b is provided to the Client. This customized TE Topology is merged with the Client's Native TE Topology and the resulting topology is depicted in Figure 8c.

```

[R1] ++++++ [A] ++++++ [E] ++++++ [R3]
                ++++++
                ++++++
                ++++++
                ++++++
                ++++++
[R2] ++++++ [B] ++++++ [F] ++++++ [R4]

```

Figure 8c: Customized TE Topology merged with the Client's Native TE Topology

The data model proposed in this document can be used to retrieve/represent/manipulate the customized TE Topology depicted in Figure 8b.

A customized TE topology is not necessarily an abstract TE topology. The provider may produce, for example, an abstract TE topology of certain type (e.g. single-abstract-node-with-connectivity-matrix topology, a border-nodes-connected-via-mesh-of-abstract-links topology, etc.) and expose it to all/some clients in expectation that the clients will use it without customization. On the other hand, a client may request a customized version of the provider's native TE topology (e.g. by requesting removal of TE links

which belong to certain layers, are too slow, not protected and/or have a certain affinity). Note that the resulting TE topology will not be abstract (because it will not contain abstract elements), but customized (modified upon client's instructions).

The client ID field in the TE topology identifier (Section 5.4. ) indicates which client the TE topology is customized for. Although an authorized client MAY receive a TE topology with the client ID field matching some other client, the client can customize only TE topologies with the client ID field either 0 or matching the ID of the client in question. If the client starts reconfiguration of a topology its client ID will be automatically set in the topology ID field for all future configurations and updates wrt. the topology in question.

The provider MAY tell the client that a given TE topology cannot be re-negotiated, by setting its own (provider's) ID in the client ID field of the topology ID.

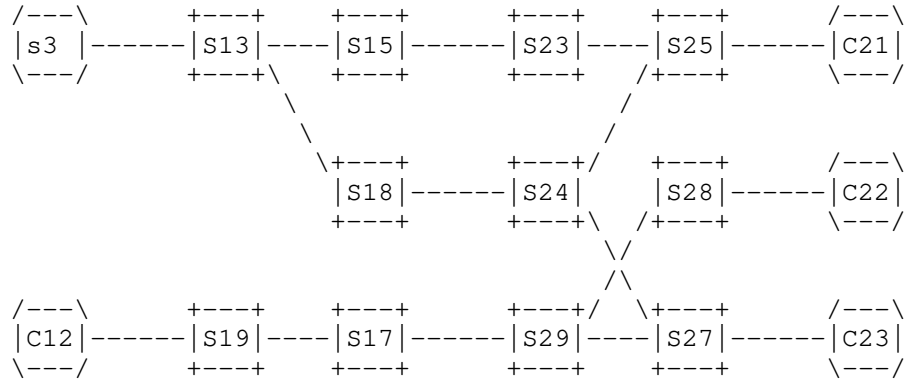
Even though this data model allows to access TE topology information across clients, implementations MAY restrict access for particular clients to particular data fields. The Network Configuration Access Control Model (NACM) [RFC8341] provides such a mechanism.

#### 4.3. Merging TE Topologies Provided by Multiple Providers

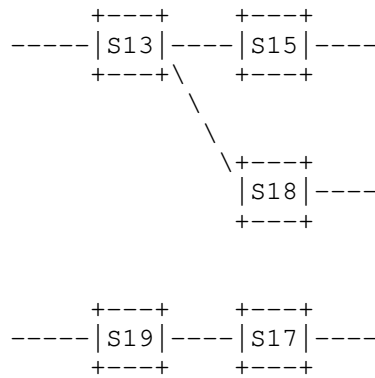
A client may receive TE topologies provided by multiple providers, each of which managing a separate domain of multi-domain network. In order to make use of said topologies, the client is expected to merge the provided TE topologies into one or more client's native TE topologies, each of which homogeneously representing the multi-domain network. This makes it possible for the client to select end-to-end TE paths for its services traversing multiple domains.

In particular, the process of merging TE topologies includes:

- Identifying neighboring domains and locking their topologies horizontally by connecting their inter-domain open-ended TE links;
- Renaming TE node, link, and SRLG IDs to ones allocated from a separate name space; this is necessary because all TE topologies are considered to be, generally speaking, independent with a possibility of clashes among TE node, link or SRLG IDs;
- Locking, vertically, TE topologies associated with different layer networks, according to provided topology inter-layer locks; this is to facilitate inter-layer path computations across multiple TE topologies provided by the same topology provider.



Domain 1 TE Topology



Domain 2 TE Topology

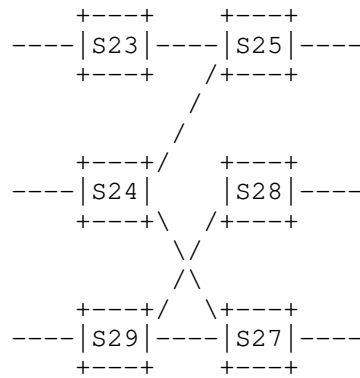


Figure 9: Merging Domain TE Topologies

Figure 9 illustrates the process of merging, by the client, of TE topologies provided by the client's providers. In the Figure, each of the two providers caters to the client (abstract or native) TE topology, describing the network domain under the respective provider's control. The client, by consulting the attributes of the inter-domain TE links - such as inter-domain plug IDs or remote TE node/link IDs (as defined by the TE Topology model) - is able to determine that:

- a) the two domains are adjacent and are inter-connected via three inter-domain TE links, and;

- b) each domain is connected to a separate customer site, connecting the left domain in the Figure to customer devices C-11 and C-12, and the right domain to customer devices C-21, C-22 and C-23.

Therefore, the client inter-connects the open-ended TE links, as shown on the upper part of the Figure.

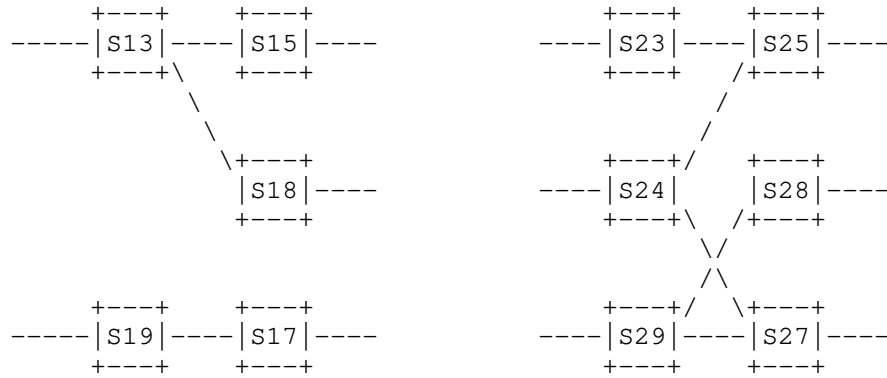
As mentioned, one way to inter-connect the open-ended inter-domain TE links of neighboring domains is to mandate the providers to specify remote nodeID/linkID attribute in the provided inter-domain TE links. This, however, may prove to be not flexible. For example, the providers may not know the respective remote nodeIDs/ linkIDs. More importantly, this option does not allow for the client to mix-n-match multiple (more than one) topologies catered by the same providers (see below). Another, more flexible, option to resolve the open-ended inter-domain TE links is by annotating them with the inter-domain plug ID attribute. Inter-domain plug ID is a network-wide unique number that identifies on the network a connectivity supporting a given inter-domain TE link. Instead of specifying remote node ID/link ID, an inter-domain TE link may provide a non-zero inter-domain plug ID. It is expected that two neighboring domain TE topologies (provided by separate providers) will have each at least one open-ended inter-domain TE link with an inter-domain plug ID matching to one provided by its neighbor. For example, the inter-domain TE link originating from node S15 of the Domain 1 TE topology (Figure 9) and the inter-domain TE link coming from node S23 of Domain 2 TE topology may specify matching inter-domain plug ID (e.g. 175344). This allows for the client to identify adjacent nodes in the separate neighboring TE topologies and resolve the inter-domain TE links connecting them regardless of their respective nodeIDs/linkIDs (which, as mentioned, could be allocated from independent name spaces). Inter-domain plug IDs may be assigned and managed by a central network authority. Alternatively, inter-domain plug IDs could be dynamically auto-discovered (e.g. via LMP protocol).

Furthermore, the client renames the TE nodes, links and SRLGs offered in the abstract TE topologies by assigning to them IDs allocated from a separate name space managed by the client. Such renaming is necessary, because the two abstract TE topologies may have their own name spaces, generally speaking, independent one from another; hence, ID overlaps/clashes are possible. For example, both TE topologies have TE nodes named S7, which, after renaming, appear in the merged TE topology as S17 and S27, respectively.

Once the merging process is complete, the client can use the merged TE topology for path computations across both domains, for example, to compute a TE path connecting C-11 to C-23.

#### 4.4. Dealing with Multiple Abstract TE Topologies Provided by the Same Provider

Domain 1 Abstract TE Topology 1      Domain 2 Abstract TE Topology 1



Domain 1 Abstract TE Topology 1      Domain 2 Abstract TE Topology 1

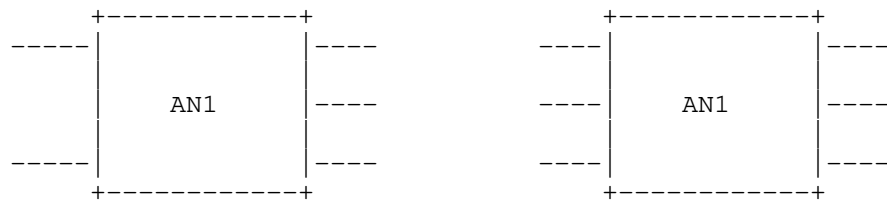


Figure 10: Merging Domain TE Topologies

Based on local configuration, templates and/or policies pushed by the client, a given provider may expose more than one abstract TE topology to the client. For example, one abstract TE topology could be optimized based on a lowest-cost criterion, while another one could be based on best possible delay metrics, while yet another one could be based on maximum bandwidth availability for the client services. Furthermore, the client may request all or some providers to expose additional abstract TE topologies, possibly of a different type and/or optimized differently, as compared to already-provided TE topologies. In any case, the client should be prepared for a provider to offer to the client more than one abstract TE topology.

It should be up to the client (based on the client's local configuration and/or policies conveyed to the client by the client's

clients) to decide how to mix-and-match multiple abstract TE topologies provided by each or some of the providers, as well as how to merge them into the client's native TE topologies. The client also decides how many such merged TE topologies it needs to produce and maintain. For example, in addition to the merged TE topology depicted in the upper part of Figure 9, the client may merge the abstract TE topologies received from the two providers, as shown in Figure 10, into the client's additional native TE topologies, as shown in Figure 11.

Note that allowing for the client mix-n-matching of multiple TE topologies assumes that inter-domain plug IDs (rather than remote nodeID/linkID) option is used for identifying neighboring domains and inter-domain TE link resolution.



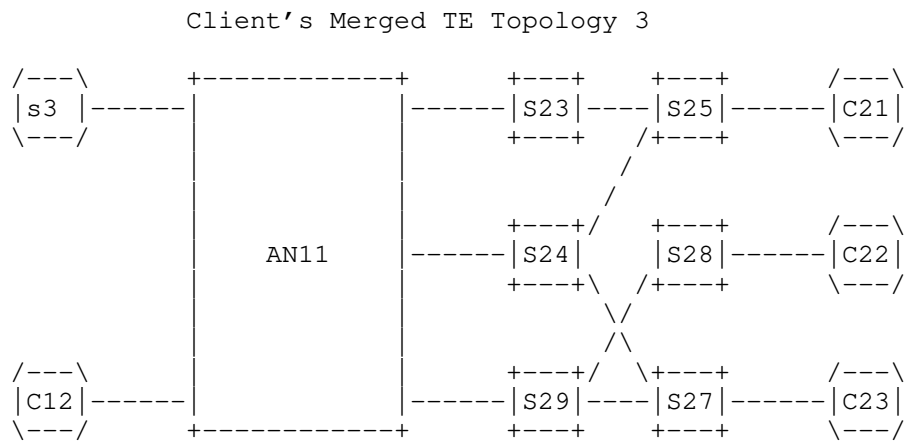
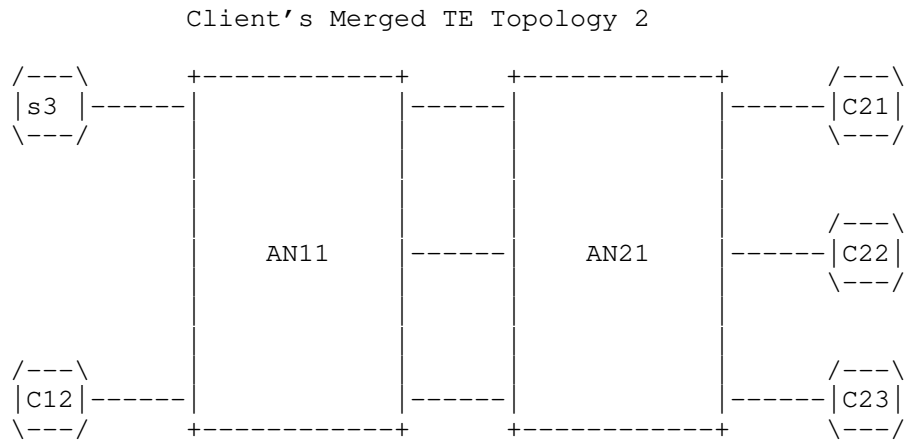


Figure 11: Multiple Native (Merged) Client's TE Topologies

It is important to note that each of the three native (merged) TE topologies could be used by the client for computing TE paths for any of the multi-domain services. The choice as to which topology to use for a given service depends on the service parameters/requirements and the topology's style, optimization criteria and the level of details.

## 5. Modeling Considerations

### 5.1. Network topology building blocks

The network topology building blocks are discussed in [RFC8345]. The TE Topology model proposed in this document augments and uses the `ietf-network-topology` module defined in [RFC8345].

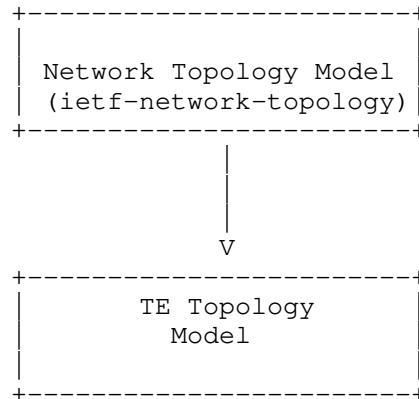


Figure 12: Augmenting the Network Topology Model

### 5.2. Technology agnostic TE Topology model

The TE Topology model proposed in this document is meant to be network technology agnostic. Other technology specific TE Topology models can augment and use the building blocks provided by the proposed model.

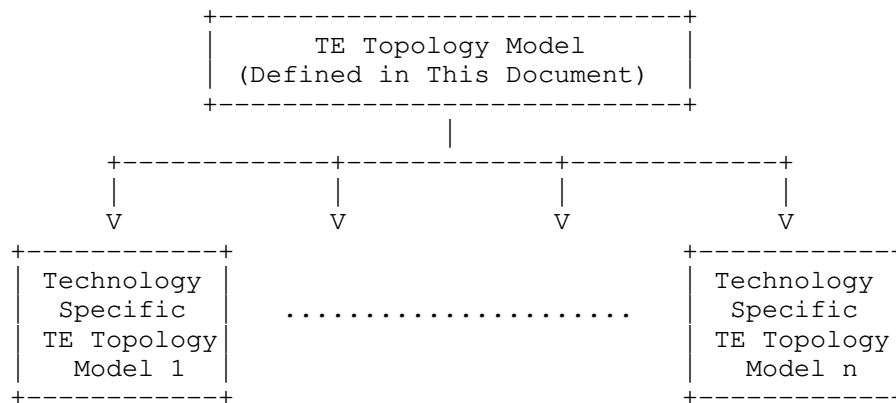


Figure 13: Augmenting the Technology agnostic TE Topology model

### 5.3. Model Structure

The high-level model structure proposed by this document is as shown below:

```

module: ietf-te-topology
augment /nw:networks/nw:network/nw:network-types:
  +--rw te-topology!

augment /nw:networks:
  +--rw te!
  +--rw templates
    +--rw node-template* [name] {template}?
    | .....
    +--rw link-template* [name] {template}?
    .....

augment /nw:networks/nw:network:
  +--rw te-topology-identifier
  | +--rw provider-id?    te-global-id
  | +--rw client-id?     te-global-id
  | +--rw topology-id?   te-topology-id
  +--rw te!
  | .....

augment /nw:networks/nw:network/nw:node:
  +--rw te-node-id?     te-types:te-node-id
  +--rw te!
  | .....
  +--rw tunnel-termination-point* [tunnel-tp-id]

```

```

    +--rw tunnel-tp-id      binary
    | .....
    +--rw supporting-tunnel-termination-point* [node-ref tunnel-
tp-ref]
    | .....

```

```

augment /nw:networks/nw:network/nt:link:
  +--rw te!
  | .....

```

```

augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--rw te-tp-id?   te-types:te-tp-id
  +--rw te!
  | .....

```

#### 5.4. Topology Identifiers

The TE-Topology is uniquely identified by a key that has 3 constituents - topology-id, provider-id and client-id. The combination of provider-id and topology-id uniquely identifies a native TE Topology on a given provider. The client-id is used only when Customized TE Topologies come into play; a value of "0" is used as the client-id for native TE Topologies.

```

augment /nw:networks/nw:network:
  +--rw te-topology-identifier
  |   +--rw provider-id?   te-global-id
  |   +--rw client-id?     te-global-id
  |   +--rw topology-id?   te-topology-id
  +--rw te!
  | .....

```

#### 5.5. Generic TE Link Attributes

The model covers the definitions for generic TE Link attributes - bandwidth, admin groups, SRLGs, switching capabilities, TE metric extensions etc.

```

+--rw te-link-attributes
  .....
  +--rw admin-status?                                te-admin-status
  | .....
  +--rw link-index?                                  uint64
  +--rw administrative-group?                         te-types:admin-groups
  +--rw link-protection-type?                        enumeration
  +--rw max-link-bandwidth?                           te-bandwidth

```

```

+--rw max-resv-link-bandwidth?          te-bandwidth
+--rw unreserved-bandwidth* [priority]
|   .....
+--rw te-default-metric?                  uint32
|   .....
+--rw te-srlgs
+--rw te-nsrlgs {nsrlg}?                  .....

```

## 5.6. Generic TE Node Attributes

The model covers the definitions for generic TE Node attributes.

The definition of a generic connectivity matrix is shown below:

```

+--rw te-node-attributes
|   .....
|   +--rw connectivity-matrices
|   |   .....
|   |   +--rw connectivity-matrix* [id]
|   |   |   +--rw id                uint32
|   |   |   +--rw from
|   |   |   |   +--rw tp-ref?         leafref
|   |   |   |   +--rw label-restrictions
|   |   |   +--rw to
|   |   |   |   +--rw tp-ref?         leafref
|   |   |   |   +--rw label-restrictions
|   |   |   +--rw is-allowed?        boolean
|   |   .....
|   |   +--rw underlay! {te-topology-hierarchy}?
|   |   .....
|   |   +--rw path-constraints
|   |   .....
|   |   +--rw optimizations
|   |   .....
|   |   +--rw path-properties
|   |   .....
|   .....

```

The definition of a TTP Local Link Connectivity List is shown below:

```

+--rw tunnel-termination-point* [tunnel-tp-id]
|   +--rw tunnel-tp-id                binary
|   +--rw admin-status?                te-types:te-admin-status
|   +--rw name?                        string
|   +--rw switching-capability?        identityref
|   +--rw encoding?                    identityref
|   +--rw inter-layer-lock-id*         uint32

```

```

    +--rw protection-type?          Identityref
    +--rw client-layer-adaptation
    .....
    +--rw local-link-connectivities
    .....
    |   +--rw local-link-connectivity* [link-tp-ref]
    |       +--rw link-tp-ref          leafref
    |       +--rw label-restrictions
    .....
    |       +--rw is-allowed?           boolean
    |       +--rw underlay {te-topology-hierarchy}?
    .....
    |       +--rw path-constraints
    .....
    |       +--rw optimizations
    .....
    |       +--ro path-properties
    .....
    +--rw supporting-tunnel-termination-point* [node-ref tunnel-tp-
ref]
        +--rw node-ref                inet:uri
        +--rw tunnel-tp-ref            binary

```

The attributes directly under container connectivity-matrices are the default attributes for all connectivity-matrix entries when the per entry corresponding attribute is not specified. When a per entry attribute is specified, it overrides the corresponding attribute directly under the container connectivity-matrices. The same rule applies to the attributes directly under container local-link-connectivities.

Each TTP (Tunnel Termination Point) MAY be supported by one or more supporting TTPs. If the TE node hosting the TTP in question refers to a supporting TE node, then the supporting TTPs are hosted by the supporting TE node. If the TE node refers to an underlay TE topology, the supporting TTPs are hosted by one or more specified TE nodes of the underlay TE topology.

### 5.7. TED Information Sources

The model allows each TE topological element to have multiple TE information sources (OSPF-TE, ISIS-TE, BGP-LS, User-Configured, System-Processed, Other). Each information source is associated with a credibility preference to indicate precedence. In scenarios where a customized TE Topology is merged into a Client's native TE Topology, the merged topological elements would point to the corresponding customized TE Topology as its information source.

```

augment /nw:networks/nw:network/nw:node:
  +--rw te!
    .....
    +--ro information-source?          te-info-source
    +--ro information-source-instance? string
    +--ro information-source-state
      |
      +--ro credibility-preference?    uint16
      +--ro logical-network-element?   string
      +--ro network-instance?          string
      +--ro topology
        |
        +--ro node-ref?                leafref
        +--ro network-ref?            leafref
    +--ro information-source-entry*
      |
      [information-source information-source-instance]
      +--ro information-source          te-info-source
      +--ro information-source-instance string
      .....

augment /nw:networks/nw:network/nt:link:
  +--rw te!
    .....
    +--ro information-source?          te-info-source
    +--ro information-source-instance? string
    +--ro information-source-state
      |
      +--ro credibility-preference?    uint16
      +--ro logical-network-element?   string
      +--ro network-instance?          string
      +--ro topology
        |
        +--ro link-ref?                leafref
        +--ro network-ref?            leafref
    +--ro information-source-entry*
      |
      [information-source information-source-instance]
      +--ro information-source          te-info-source
      +--ro information-source-instance string
      .....

```

## 5.8. Overlay/Underlay Relationship

The model captures overlay and underlay relationship for TE nodes/links. For example - in networks where multiple TE Topologies are built hierarchically, this model allows the user to start from a specific topological element in the top most topology and traverse all the way down to the supporting topological elements in the bottom most topology.

This relationship is captured via the "underlay-topology" field for the node and via the "underlay" field for the link. The use of these

fields is optional and this functionality is tagged as a "feature" ("te-topology-hierarchy").

```
augment /nw:networks/nw:network/nw:node:
  +--rw te-node-id?   te-types:te-node-id
  +--rw te!
    +--rw te-node-template*          leafref {template}?
    +--rw te-node-attributes
      | .....
      | +--rw underlay-topology {te-topology-hierarchy}?
      |   +--rw network-ref?   leafref
    +--rw te-link-attributes
      | .....
      | +--rw underlay {te-topology-hierarchy}?
      |   +--rw enabled?          boolean
      |   +--rw primary-path
      |     | +--rw network-ref?   leafref
      |     | .....
      |   +--rw backup-path* [index]
      |     | +--rw index          uint32
      |     | +--rw network-ref?   leafref
      |     | .....
      |   +--rw protection-type?    identityref
      |   +--rw tunnel-termination-points
      |     | +--rw source?        binary
      |     | +--rw destination?   binary
      |   +--rw tunnels
      |     | .....
      |     |
```

## 5.9. Templates

The data model provides the users with the ability to define templates and apply them to link and node configurations. The use of "template" configuration is optional and this functionality is tagged as a "feature" ("template").

```
augment /nw:networks/nw:network/nw:node:
  +--rw te-node-id?   te-types:te-node-id
  +--rw te!
    +--rw te-node-template*
      | -> ../../../../te/templates/node-template/name
      | {template}?
```



```

augment /nw:networks/nw:network/nt:link:
  +--rw te!
    +--rw te-link-template*
      |      -> ../../../../te/templates/link-template/name
      |      {template}?

augment /nw:networks:
  +--rw te!
    +--rw templates
      +--rw node-template* [name] {template}?
        +--rw name
          |      te-types:te-template-name
        +--rw priority?          uint16
        +--rw reference-change-policy?  enumeration
        +--rw te-node-attributes
        .....
      +--rw link-template* [name] {template}?
        +--rw name
          |      te-types:te-template-name
        +--rw priority?          uint16
        +--rw reference-change-policy?  enumeration
        +--rw te-link-attributes
        .....

```

Multiple templates can be specified to a configuration element. When two or more templates specify values for the same configuration field, the value from the template with the highest priority is used. The range of the priority is from 0 to 65535, with a lower number indicating a higher priority. The reference-change-policy specifies the action that needs to be taken when the template changes on a configuration element that has a reference to this template. The choices of action include taking no action, rejecting the change to the template and applying the change to the corresponding configuration.

#### 5.10. Scheduling Parameters

The model allows time scheduling parameters to be specified for each topological element or for the topology as a whole. These parameters allow the provider to present different topological views to the client at different time slots. The use of "scheduling parameters" is optional.

The YANG data model for configuration scheduling is defined in [I-D.liu-netmod-yang-schedule], which allows specifying configuration schedules without altering this data model.

### 5.11. Notifications

Notifications are a key component of any topology data model.

[I-D.ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] define a subscription and push mechanism for YANG datastores. This mechanism currently allows the user to:

- Subscribe notifications on a per client basis
- Specify subtree filters or xpath filters so that only interested contents will be sent.
- Specify either periodic or on-demand notifications.

### 6. Guidance for Writing Technology Specific TE Topology Augmentations

The TE topology model defined in this document is technology agnostic as it defines concepts, abstractions and attributes that are common across multiple network technologies. It is envisioned that this base model will be widely used when defining technology specific TE topology models for various layer networks.

[I-D.ietf-ccamp-wson-yang], [I-D.ietf-ccamp-otn-topo-yang], and [I-D.ietf-teas-yang-l3-te-topo] are some examples of technology specific TE Topology models. Writers of such models are encouraged to augment the basic TE topology model's containers, such as TE Topology, TE Node, TE Link, Link Termination Point (LTP), Tunnel Termination Point (TTP), Bandwidth and Label with the layer specific attributes instead of defining new containers.

Consider the following technology specific example-topology model:

```
module: example-topology
  augment /nw:networks/nw:network/nw:network-types/tet:te-topology:
    +--rw example-topology!
  augment /nw:networks/nw:network/tet:te:
    +--rw attributes
      +--rw attribute-1?  uint8
  augment /nw:networks/nw:network/nw:node/tet:te
    /tet:te-node-attributes:
      +--rw attributes
        +--rw attribute-2?  uint8
  augment /nw:networks/nw:network/nw:node/tet:te
    /tet:te-node-attributes/tet:connectivity-matrices:
      +--rw attributes
        +--rw attribute-3?  uint8
  augment /nw:networks/nw:network/nw:node/tet:te
```

```

        /tet:te-node-attributes/tet:connectivity-matrices
        /tet:connectivity-matrix:
+--rw attributes
  +--rw attribute-3?  uint8
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point:
+--rw attributes
  +--rw attribute-4?  uint8
augment /nw:networks/nw:network/nw:node/nt:termination-point
  /tet:te:
+--rw attributes
  +--rw attribute-5?  uint8
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:te-link-attributes:
+--rw attributes
  +--rw attribute-6?  uint8

```

The technology specific TE bandwidth for this example topology can be specified using the following augment statements:

```

augment /nw:networks/tet:te/tet:templates/tet:link-template
  /tet:te-link-attributes
  /tet:interface-switching-capability/tet:max-lsp-bandwidth
  /tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?  uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template
  /tet:te-link-attributes/tet:max-link-bandwidth
  /tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?  uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template
  /tet:te-link-attributes/tet:max-resv-link-bandwidth
  /tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?  uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template

```

```

        /tet:te-link-attributes/tet:unreserved-bandwidth
        /tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:path-constraints/tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:path-constraints
  /tet:te-bandwidth/tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:path-constraints/tet:te-bandwidth/tet:technology:
+--:(example)
  +--ro example
    +--ro bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:path-constraints
  /tet:te-bandwidth/tet:technology:
+--:(example)
  +--ro example
    +--ro bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point/tet:client-layer-adaptation
  /tet:switching-capability/tet:te-bandwidth
  /tet:technology:
+--:(example)
  +--rw example
    +--rw bandwidth-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities/tet:path-constraints

```

```

        /tet:te-bandwidth/tet:technology:
+---: (example)
+---rw example
+---rw bandwidth-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
        /tet:tunnel-termination-point
        /tet:local-link-connectivities
        /tet:local-link-connectivity/tet:path-constraints
        /tet:te-bandwidth/tet:technology:
+---: (example)
+---rw example
+---rw bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
        /tet:te-link-attributes
        /tet:interface-switching-capability/tet:max-lsp-bandwidth
        /tet:te-bandwidth/tet:technology:
+---: (example)
+---rw example
+---rw bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
        /tet:te-link-attributes/tet:max-link-bandwidth
        /tet:te-bandwidth/tet:technology:
+---: (example)
+---rw example
+---rw bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
        /tet:te-link-attributes/tet:max-resv-link-bandwidth
        /tet:te-bandwidth/tet:technology:
+---: (example)
+---rw example
+---rw bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
        /tet:information-source-entry
        /tet:interface-switching-capability/tet:max-lsp-bandwidth
        /tet:te-bandwidth/tet:technology:
+---: (example)
+---ro example
+---ro bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
        /tet:information-source-entry/tet:max-link-bandwidth
        /tet:te-bandwidth/tet:technology:

```

```

    +---:(example)
      +---ro example
        +---ro bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
      /tet:information-source-entry/tet:max-resv-link-bandwidth
      /tet:te-bandwidth/tet:technology:
    +---:(example)
      +---ro example
        +---ro bandwidth-1?    uint32
augment /nw:networks/nw:network/nt:link/tet:te
      /tet:information-source-entry/tet:unreserved-bandwidth
      /tet:te-bandwidth/tet:technology:
    +---:(example)
      +---ro example
        +---ro bandwidth-1?    uint32
augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te
      /tet:interface-switching-capability/tet:max-lsp-bandwidth
      /tet:te-bandwidth/tet:technology:
    +---:(example)
      +---rw example
        +---rw bandwidth-1?    uint32

```

The technology specific TE label for this example topology can be specified using the following augment statements:

```

augment /nw:networks/tet:te/tet:templates/tet:link-template
      /tet:te-link-attributes/tet:underlay/tet:primary-path
      /tet:path-element/tet:type/tet:label/tet:label-hop
      /tet:te-label/tet:technology:
    +---:(example)
      +---rw example
        +---rw label-1?    uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template
      /tet:te-link-attributes/tet:underlay/tet:backup-path
      /tet:path-element/tet:type/tet:label/tet:label-hop
      /tet:te-label/tet:technology:
    +---:(example)
      +---rw example
        +---rw label-1?    uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template

```

```

        /tet:te-link-attributes/tet:label-restrictions
        /tet:label-restriction/tet:label-start/tet:te-label
        /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/tet:te/tet:templates/tet:link-template
  /tet:te-link-attributes/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:label-restrictions/tet:label-restriction
  /tet:label-start/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:label-restrictions/tet:label-restriction
  /tet:label-end/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:underlay/tet:primary-path/tet:path-element/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:underlay/tet:backup-path/tet:path-element/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32

```

```

augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:path-properties/tet:path-route-objects
  /tet:path-route-object/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:from/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:from/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:to/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:to/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te

```



```

        /tet:te-node-attributes/tet:connectivity-matrices
        /tet:connectivity-matrix/tet:underlay/tet:primary-path
        /tet:path-element/tet:type/tet:label/tet:label-hop
        /tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:underlay/tet:backup-path
  /tet:path-element/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:path-properties
  /tet:path-route-objects/tet:path-route-object/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:label-restrictions/tet:label-restriction
  /tet:label-start/tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:label-restrictions/tet:label-restriction
  /tet:label-end/tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:underlay/tet:primary-path/tet:path-element/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:

```

```

    +--:(example)
      +--ro example
        +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:underlay/tet:backup-path/tet:path-element/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:path-properties/tet:path-route-objects
  /tet:path-route-object/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:from/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:from/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?    uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:information-source-entry/tet:connectivity-matrices
  /tet:connectivity-matrix/tet:to/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--ro example
```

```

      +---ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
      /tet:information-source-entry/tet:connectivity-matrices
      /tet:connectivity-matrix/tet:to/tet:label-restrictions
      /tet:label-restriction/tet:label-end/tet:te-label
      /tet:technology:
+---:(example)
  +---ro example
    +---ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
      /tet:information-source-entry/tet:connectivity-matrices
      /tet:connectivity-matrix/tet:underlay/tet:primary-path
      /tet:path-element/tet:type/tet:label/tet:label-hop
      /tet:te-label/tet:technology:
+---:(example)
  +---ro example
    +---ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
      /tet:information-source-entry/tet:connectivity-matrices
      /tet:connectivity-matrix/tet:underlay/tet:backup-path
      /tet:path-element/tet:type/tet:label/tet:label-hop
      /tet:te-label/tet:technology:
+---:(example)
  +---ro example
    +---ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
      /tet:information-source-entry/tet:connectivity-matrices
      /tet:connectivity-matrix/tet:path-properties
      /tet:path-route-objects/tet:path-route-object/tet:type
      /tet:label/tet:label-hop/tet:te-label/tet:technology:
+---:(example)
  +---ro example
    +---ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
      /tet:tunnel-termination-point
      /tet:local-link-connectivities/tet:label-restrictions
      /tet:label-restriction/tet:label-start/tet:te-label
      /tet:technology:
+---:(example)
  +---rw example
    +---rw label-1?   uint32

```

```
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities/tet:underlay
  /tet:primary-path/tet:path-element/tet:type/tet:label
  /tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities/tet:underlay
  /tet:backup-path/tet:path-element/tet:type/tet:label
  /tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities/tet:path-properties
  /tet:path-route-objects/tet:path-route-object/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?   uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities
  /tet:local-link-connectivity/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?   uint32
```

```

augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities
  /tet:local-link-connectivity/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
  +--:(example)
    +--rw example
      +--rw label-1?  uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities
  /tet:local-link-connectivity/tet:underlay
  /tet:primary-path/tet:path-element/tet:type/tet:label
  /tet:label-hop/tet:te-label/tet:technology:
  +--:(example)
    +--rw example
      +--rw label-1?  uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities
  /tet:local-link-connectivity/tet:underlay/tet:backup-path
  /tet:path-element/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
  +--:(example)
    +--rw example
      +--rw label-1?  uint32
augment /nw:networks/nw:network/nw:node/tet:te
  /tet:tunnel-termination-point
  /tet:local-link-connectivities
  /tet:local-link-connectivity/tet:path-properties
  /tet:path-route-objects/tet:path-route-object/tet:type
  /tet:label/tet:label-hop/tet:te-label/tet:technology:
  +--:(example)
    +--ro example
      +--ro label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:te-link-attributes/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
  +--:(example)

```

```

    +--rw example
      +--rw label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:te-link-attributes/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:te-link-attributes/tet:underlay/tet:primary-path
  /tet:path-element/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:te-link-attributes/tet:underlay/tet:backup-path
  /tet:path-element/tet:type/tet:label/tet:label-hop
  /tet:te-label/tet:technology:
+--:(example)
  +--rw example
    +--rw label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:information-source-entry/tet:label-restrictions
  /tet:label-restriction/tet:label-start/tet:te-label
  /tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?  uint32
augment /nw:networks/nw:network/nt:link/tet:te
  /tet:information-source-entry/tet:label-restrictions
  /tet:label-restriction/tet:label-end/tet:te-label
  /tet:technology:
+--:(example)
  +--ro example
    +--ro label-1?  uint32

```

The YANG module to implement the above example topology can be seen in Appendix C.

## 7. TE Topology YANG Module

This module references [RFC1195], [RFC3209], [RFC3272], [RFC3471], [RFC3630], [RFC3785], [RFC4201], [RFC4202], [RFC4203], [RFC4206], [RFC4872], [RFC5152], [RFC5212], [RFC5305], [RFC5316], [RFC5329], [RFC5392], [RFC6001], [RFC6241], [RFC6991], [RFC7308], [RFC7471], [RFC7579], [RFC7752], [RFC8345], and [I-D.ietf-teas-yang-te-types].

```
<CODE BEGINS> file "ietf-te-topology@2019-02-07.yang"
module ietf-te-topology {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology";

  prefix "tet";

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

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

  import ietf-te-types {
    prefix "te-types";
    reference
      "I-D.ietf-teas-yang-te-types: Traffic Engineering Common YANG
      Types";
  }

  import ietf-network {
    prefix "nw";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
  }

  import ietf-network-topology {
    prefix "nt";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
  }
}
```

## organization

"IETF Traffic Engineering Architecture and Signaling (TEAS)  
Working Group";

## contact

"WG Web: <<http://tools.ietf.org/wg/teas/>>

WG List: <<mailto:teas@ietf.org>>

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

Editor: Igor Bryskin  
<<mailto:Igor.Bryskin@huawei.com>>

Editor: Vishnu Pavan Beeram  
<<mailto:vbeeram@juniper.net>>

Editor: Tarek Saad  
<<mailto:tsaad@juniper.net>>

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

Editor: Oscar Gonzalez De Dios  
<<mailto:oscar.gonzalezdedios@telefonica.com>>;

## description

"TE topology model for representing and manipulating technology  
agnostic TE Topologies.

Copyright (c) 2019 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.";

revision "2019-02-07" {
    description "Initial revision";
    reference "RFC XXXX: YANG Data Model for TE Topologies";
    // RFC Ed.: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Features
 */
feature nsrlg {
    description
        "This feature indicates that the system supports NSRLG
        (Not Sharing Risk Link Group).";
}

feature te-topology-hierarchy {
    description
        "This feature indicates that the system allows underlay
        and/or overlay TE topology hierarchy.";
}

feature template {
    description
        "This feature indicates that the system supports
        template configuration.";
}

/*
 * Typedefs
 */
typedef geographic-coordinate-degree {
    type decimal64 {
        fraction-digits 8;
    }
    description
        "Decimal degree (DD) used to express latitude and longitude
        geographic coordinates.";
} // geographic-coordinate-degree
```

```
typedef te-info-source {
  type enumeration {
    enum "unknown" {
      description "The source is unknown.";
    }
    enum "locally-configured" {
      description "Configured entity.";
    }
    enum "ospfv2" {
      description "OSPFv2.";
    }
    enum "ospfv3" {
      description "OSPFv3.";
    }
    enum "isis" {
      description "ISIS.";
    }
    enum "bgp-ls" {
      description "BGP-LS.";
      reference
        "RFC 7752: North-Bound Distribution of Link-State and
        Traffic Engineering (TE) Information Using BGP";
    }
    enum "system-processed" {
      description "System processed entity.";
    }
    enum "other" {
      description "Other source.";
    }
  }
  description
    "Describing the type of source that has provided the
    related information, and the source credibility.";
} // te-info-source

/*
 * Groupings
 */
grouping connectivity-matrix-entry-path-attributes {
  description
```

```
    "Attributes of connectivity matrix entry.";
  leaf is-allowed {
    type boolean;
    description
      "true - switching is allowed,
       false - switching is disallowed.";
  }
  container underlay {
    if-feature te-topology-hierarchy;
    description "Attributes of the te-link underlay.";
    reference
      "RFC 4206: Label Switched Paths (LSP) Hierarchy with
       Generalized Multi-Protocol Label Switching (GMPLS)
       Traffic Engineering (TE)";

    uses te-link-underlay-attributes;
  } // underlay

  uses te-types:generic-path-constraints;
  uses te-types:generic-path-optimization;
  uses te-types:generic-path-properties;
} // connectivity-matrix-entry-path-attributes

grouping geolocation-container {
  description
    "A container containing a GPS location.";
  container geolocation{
    config false;
    description
      "A container containing a GPS location.";
    leaf altitude {
      type int64;
      units millimeter;
      description
        "Distance above the sea level.";
    }
    leaf latitude {
      type geographic-coordinate-degree {
        range "-90..90";
      }
      description

```

```
        "Relative position north or south on the Earth's surface.";
    }
    leaf longitude {
        type geographic-coordinate-degree {
            range "-180..180";
        }
        description
            "Angular distance east or west on the Earth's surface.";
    }
} // gps-location
} // geolocation-container

grouping information-source-state-attributes {
    description
        "The attributes identifying source that has provided the
        related information, and the source credibility.";
    leaf credibility-preference {
        type uint16;
        description
            "The preference value to calculate the traffic
            engineering database credibility value used for
            tie-break selection between different
            information-source values.
            Higher value is more preferable.";
    }
    leaf logical-network-element {
        type string;
        description
            "When applicable, this is the name of a logical network
            element from which the information is learned.";
    } // logical-network-element
    leaf network-instance {
        type string;
        description
            "When applicable, this is the name of a network-instance
            from which the information is learned.";
    } // network-instance
} // information-source-state-attributes

grouping information-source-per-link-attributes {
    description
```

```
    "Per node container of the attributes identifying source that
      has provided the related information, and the source
      credibility.";
  leaf information-source {
    type te-info-source;
    config false;
    description
      "Indicates the type of the information source.";
  }
  leaf information-source-instance {
    type string;
    config false;
    description
      "The name indicating the instance of the information
        source.";
  }
  container information-source-state {
    config false;
    description
      "The container contains state attributes related to
        the information source.";
    uses information-source-state-attributes;
    container topology {
      description
        "When the information is processed by the system,
          the attributes in this container indicate which topology
          is used to process to generate the result information.";
      uses nt:link-ref;
    } // topology
  } // information-source-state
} // information-source-per-link-attributes

grouping information-source-per-node-attributes {
  description
    "Per node container of the attributes identifying source that
      has provided the related information, and the source
      credibility.";
  leaf information-source {
    type te-info-source;
    config false;
    description
```

```
        "Indicates the type of the information source.";
    }
    leaf information-source-instance {
        type string;
        config false;
        description
            "The name indicating the instance of the information
             source.";
    }
    container information-source-state {
        config false;
        description
            "The container contains state attributes related to
             the information source.";
        uses information-source-state-attributes;
        container topology {
            description
                "When the information is processed by the system,
                 the attributes in this container indicate which topology
                 is used to process to generate the result information.";
            uses nw:node-ref;
        } // topology
    } // information-source-state
} // information-source-per-node-attributes

grouping interface-switching-capability-list {
    description
        "List of Interface Switching Capabilities Descriptors (ISCD)";
    list interface-switching-capability {
        key "switching-capability encoding";
        description
            "List of Interface Switching Capabilities Descriptors (ISCD)
             for this link.";
        reference
            "RFC 3471: Generalized Multi-Protocol Label Switching (GMPLS)
             Signaling Functional Description.
             RFC 4203: OSPF Extensions in Support of Generalized
             Multi-Protocol Label Switching (GMPLS).";
    }
    leaf switching-capability {
        type identityref {
            base te-types:switching-capabilities;
        }
    }
}
```

```
    }
    description
      "Switching Capability for this interface.";
  }
  leaf encoding {
    type identityref {
      base te-types:lsp-encoding-types;
    }
    description
      "Encoding supported by this interface.";
  }
  uses te-link-iscd-attributes;
} // interface-switching-capability
} // interface-switching-capability-list

grouping statistics-per-link {
  description
    "Statistics attributes per TE link.";
  leaf discontinuity-time {
    type yang:date-and-time;
    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.";
  }
  /* Administrative attributes */
  leaf disables {
    type yang:counter32;
    description
      "Number of times that link was disabled.";
  }
  leaf enables {
    type yang:counter32;
    description
      "Number of times that link was enabled.";
  }
  leaf maintenance-clears {
    type yang:counter32;
```

```
        description
            "Number of times that link was put out of maintenance.";
    }
    leaf maintenance-sets {
        type yang:counter32;
        description
            "Number of times that link was put in maintenance.";
    }
    leaf modifies {
        type yang:counter32;
        description
            "Number of times that link was modified.";
    }
    /* Operational attributes */
    leaf downs {
        type yang:counter32;
        description
            "Number of times that link was set to operational down.";
    }
    leaf ups {
        type yang:counter32;
        description
            "Number of times that link was set to operational up.";
    }
    /* Recovery attributes */
    leaf fault-clears {
        type yang:counter32;
        description
            "Number of times that link experienced fault clear event.";
    }
    leaf fault-detects {
        type yang:counter32;
        description
            "Number of times that link experienced fault detection.";
    }
    leaf protection-switches {
        type yang:counter32;
        description
            "Number of times that link experienced protection
            switchover.";
    }
}
```



```
leaf protection-reverts {
  type yang:counter32;
  description
    "Number of times that link experienced protection
    reversion.";
}
leaf restoration-failures {
  type yang:counter32;
  description
    "Number of times that link experienced restoration
    failure.";
}
leaf restoration-starts {
  type yang:counter32;
  description
    "Number of times that link experienced restoration
    start.";
}
leaf restoration-successes {
  type yang:counter32;
  description
    "Number of times that link experienced restoration
    success.";
}
leaf restoration-reversion-failures {
  type yang:counter32;
  description
    "Number of times that link experienced restoration reversion
    failure.";
}
leaf restoration-reversion-starts {
  type yang:counter32;
  description
    "Number of times that link experienced restoration reversion
    start.";
}
leaf restoration-reversion-successes {
  type yang:counter32;
  description
    "Number of times that link experienced restoration reversion
    success.";
```

```
    }  
  } // statistics-per-link  
  
  grouping statistics-per-node {  
    description  
      "Statistics attributes per TE node.";  
    leaf discontinuity-time {  
      type yang:date-and-time;  
      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 node {  
      description  
        "Containing TE node level statistics attributes.";  
      leaf disables {  
        type yang:counter32;  
        description  
          "Number of times that node was disabled.";  
      }  
      leaf enables {  
        type yang:counter32;  
        description  
          "Number of times that node was enabled.";  
      }  
      leaf maintenance-sets {  
        type yang:counter32;  
        description  
          "Number of times that node was put in maintenance.";  
      }  
      leaf maintenance-clears {  
        type yang:counter32;  
        description  
          "Number of times that node was put out of maintenance.";  
      }  
      leaf modifies {  
        type yang:counter32;
```

```
        description
            "Number of times that node was modified.";
    }
} // node
container connectivity-matrix-entry {
    description
        "Containing connectivity matrix entry level statistics
        attributes.";
    leaf creates {
        type yang:counter32;
        description
            "Number of times that a connectivity matrix entry was
            created.";
        reference
            "RFC 6241. Section 7.2 for 'create' operation. ";
    }
    leaf deletes {
        type yang:counter32;
        description
            "Number of times that a connectivity matrix entry was
            deleted.";
        reference
            "RFC 6241. Section 7.2 for 'delete' operation. ";
    }
    leaf disables {
        type yang:counter32;
        description
            "Number of times that a connectivity matrix entry was
            disabled.";
    }
    leaf enables {
        type yang:counter32;
        description
            "Number of times that a connectivity matrix entry was
            enabled.";
    }
    leaf modifies {
        type yang:counter32;
        description
            "Number of times that a connectivity matrix entry was
            modified.";
```

```
    }
  } // connectivity-matrix-entry
} // statistics-per-node

grouping statistics-per-ttp {
  description
    "Statistics attributes per TE TTP (Tunnel Termination Point).";
  leaf discontinuity-time {
    type yang:date-and-time;
    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 tunnel-termination-point {
    description
      "Containing TE TTP (Tunnel Termination Point) level
      statistics attributes.";
    /* Administrative attributes */
    leaf disables {
      type yang:counter32;
      description
        "Number of times that TTP was disabled.";
    }
    leaf enables {
      type yang:counter32;
      description
        "Number of times that TTP was enabled.";
    }
    leaf maintenance-clears {
      type yang:counter32;
      description
        "Number of times that TTP was put out of maintenance.";
    }
    leaf maintenance-sets {
      type yang:counter32;
      description
        "Number of times that TTP was put in maintenance.";
    }
  }
}
```

```
    }
    leaf modifies {
        type yang:counter32;
        description
            "Number of times that TTP was modified.";
    }
    /* Operational attributes */
    leaf downs {
        type yang:counter32;
        description
            "Number of times that TTP was set to operational down.";
    }
    leaf ups {
        type yang:counter32;
        description
            "Number of times that TTP was set to operational up.";
    }
    leaf in-service-clears {
        type yang:counter32;
        description
            "Number of times that TTP was taken out of service
            (TE tunnel was released).";
    }
    leaf in-service-sets {
        type yang:counter32;
        description
            "Number of times that TTP was put in service by a TE
            tunnel (TE tunnel was set up).";
    }
} // tunnel-termination-point

container local-link-connectivity {
    description
        "Containing TE LLCL (Local Link Connectivity List) level
        statistics attributes.";
    leaf creates {
        type yang:counter32;
        description
            "Number of times that an LLCL entry was created.";
        reference
            "RFC 6241. Section 7.2 for 'create' operation.";
    }
}
```

```
    }
    leaf deletes {
      type yang:counter32;
      description
        "Number of times that an LLCL entry was deleted.";
      reference
        "RFC 6241. Section 7.2 for 'delete' operation.";
    }
    leaf disables {
      type yang:counter32;
      description
        "Number of times that an LLCL entry was disabled.";
    }
    leaf enables {
      type yang:counter32;
      description
        "Number of times that an LLCL entry was enabled.";
    }
    leaf modifies {
      type yang:counter32;
      description
        "Number of times that an LLCL entry was modified.";
    }
  } // local-link-connectivity
} // statistics-per-ttp

grouping te-link-augment {
  description
    "Augmentation for TE link.";
  uses te-link-config;
  uses te-link-state-derived;
  container statistics {
    config false;
    description
      "Statistics data.";
    uses statistics-per-link;
  } // statistics
} // te-link-augment

grouping te-link-config {
  description
```

```
"TE link configuration grouping.";
choice bundle-stack-level {
  description
    "The TE link can be partitioned into bundled
    links, or component links.";
  case bundle {
    container bundled-links {
      description
        "A set of bundled links.";
      reference
        "RFC 4201: Link Bundling in MPLS Traffic Engineering
        (TE).";
      list bundled-link {
        key "sequence";
        description
          "Specify a bundled interface that is
          further partitioned.";
        leaf sequence {
          type uint32;
          description
            "Identify the sequence in the bundle.";
        }
      } // list bundled-link
    }
  }
  case component {
    container component-links {
      description
        "A set of component links";
      list component-link {
        key "sequence";
        description
          "Specify a component interface that is
          sufficient to unambiguously identify the
          appropriate resources";

        leaf sequence {
          type uint32;
          description
            "Identify the sequence in the bundle.";
        }
      }
    }
  }
}
```

```
        leaf src-interface-ref {
            type string;
            description
                "Reference to component link interface on the
                 source node.";
        }
        leaf des-interface-ref {
            type string;
            description
                "Reference to component link interface on the
                 destination node.";
        }
    }
}
} // bundle-stack-level

leaf-list te-link-template {
    if-feature template;
    type leafref {
        path "../.../te/templates/link-template/name";
    }
    description
        "The reference to a TE link template.";
}
uses te-link-config-attributes;
} // te-link-config

grouping te-link-config-attributes {
    description
        "Link configuration attributes in a TE topology.";
    container te-link-attributes {
        description "Link attributes in a TE topology.";
        leaf access-type {
            type te-types:te-link-access-type;
            description
                "Link access type, which can be point-to-point or
                 multi-access.";
        }
        container external-domain {
            description
```



```
    "For an inter-domain link, specify the attributes of
      the remote end of link, to facilitate the signalling at
      local end.";
  uses nw:network-ref;
  leaf remote-te-node-id {
    type te-types:te-node-id;
    description
      "Remote TE node identifier, used together with
        remote-te-link-id to identify the remote link
        termination point in a different domain.";
  }
  leaf remote-te-link-tp-id {
    type te-types:te-tp-id;
    description
      "Remote TE link termination point identifier, used
        together with remote-te-node-id to identify the remote
        link termination point in a different domain.";
  }
}
leaf is-abstract {
  type empty;
  description "Present if the link is abstract.";
}
leaf name {
  type string;
  description "Link Name.";
}
container underlay {
  if-feature te-topology-hierarchy;
  description "Attributes of the te-link underlay.";
  reference
    "RFC 4206: Label Switched Paths (LSP) Hierarchy with
      Generalized Multi-Protocol Label Switching (GMPLS)
      Traffic Engineering (TE)";

  uses te-link-underlay-attributes;
} // underlay
leaf admin-status {
  type te-types:te-admin-status;
  description
    "The administrative state of the link.";
```

```
    }

    uses te-link-info-attributes;
  } // te-link-attributes
} // te-link-config-attributes

grouping te-link-info-attributes {
  description
    "Advertised TE information attributes.";
  leaf link-index {
    type uint64;
    description
      "The link identifier.  If OSPF is used, this represents an
      ospfLsdbID.  If IS-IS is used, this represents an isisLSPID.
      If a locally configured link is used, this object represents
      a unique value, which is locally defined in a router.";
  }
  leaf administrative-group {
    type te-types:admin-groups;
    description
      "Administrative group or color of the link.
      This attribute covers both administrative group (defined in
      RFC 3630, RFC 5305 and RFC 5329), and extended
      administrative group (defined in RFC 7308).";
  }
}

uses interface-switching-capability-list;
uses te-types:label-set-info;

leaf link-protection-type {
  type identityref {
    base te-types:link-protection-type;
  }
  description
    "Link Protection Type desired for this link.";
  reference
    "RFC 4202: Routing Extensions in Support of
    Generalized Multi-Protocol Label Switching (GMPLS).";
}

container max-link-bandwidth {
```

```
    uses te-types:te-bandwidth;
    description
        "Maximum bandwidth that can be seen on this link in this
        direction. Units in bytes per second.";
    reference
        "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
        Version 2.
        RFC 5305: IS-IS Extensions for Traffic Engineering.";
}
container max-resv-link-bandwidth {
    uses te-types:te-bandwidth;
    description
        "Maximum amount of bandwidth that can be reserved in this
        direction in this link. Units in bytes per second.";
    reference
        "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
        Version 2.
        RFC 5305: IS-IS Extensions for Traffic Engineering.";
}
list unreserved-bandwidth {
    key "priority";
    max-elements "8";
    description
        "Unreserved bandwidth for 0-7 priority levels. Units in
        bytes per second.";
    reference
        "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
        Version 2.
        RFC 5305: IS-IS Extensions for Traffic Engineering.";
    leaf priority {
        type uint8 {
            range "0..7";
        }
        description "Priority.";
    }
    uses te-types:te-bandwidth;
}
leaf te-default-metric {
    type uint32;
    description
        "Traffic engineering metric.";
```

```
    reference
      "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
      Version 2.
      RFC 5305: IS-IS Extensions for Traffic Engineering.";
  }
  leaf te-delay-metric {
    type uint32;
    description
      "Traffic engineering delay metric.";
    reference
      "RFC 7471: OSPF Traffic Engineering (TE) Metric Extensions.";
  }
  leaf te-igp-metric {
    type uint32;
    description
      "IGP metric used for traffic engineering.";
    reference
      "RFC 3785: Use of Interior Gateway Protocol (IGP) Metric as a
      Second MPLS Traffic Engineering (TE) Metric.";
  }
  container te-srlgs {
    description
      "Containing a list of SLRGs.";
    leaf-list value {
      type te-types:srlg;
      description "SRLG value.";
      reference
        "RFC 4202: Routing Extensions in Support of
        Generalized Multi-Protocol Label Switching (GMPLS).";
    }
  }
  container te-nsrlgs {
    if-feature nsrlg;
    description
      "Containing a list of NSRLGs (Not Sharing Risk Link
      Groups).
      When an abstract TE link is configured, this list specifies
      the request that underlay TE paths need to be mutually
      disjoint with other TE links in the same groups.";
    leaf-list id {
      type uint32;
```

```
        description
            "NSRLG ID, uniquely configured within a topology.";
        reference
            "RFC 4872: RSVP-TE Extensions in Support of End-to-End
            Generalized Multi-Protocol Label Switching (GMPLS)
            Recovery";
    }
}
} // te-link-info-attributes

grouping te-link-iscd-attributes {
    description
        "TE link ISCD (Interface Switching Capability Descriptor)
        attributes.";
    reference
        "Sec 1.4, RFC 4203: OSPF Extensions in Support of Generalized
        Multi-Protocol Label Switching (GMPLS). Section 1.4.";
    list max-lsp-bandwidth {
        key "priority";
        max-elements "8";
        description
            "Maximum LSP Bandwidth at priorities 0-7.";
        leaf priority {
            type uint8 {
                range "0..7";
            }
            description "Priority.";
        }
        uses te-types:te-bandwidth;
    }
} // te-link-iscd-attributes

grouping te-link-state-derived {
    description
        "Link state attributes in a TE topology.";
    leaf oper-status {
        type te-types:te-oper-status;
        config false;
        description
            "The current operational state of the link.";
    }
}
```

```
leaf is-transitional {
  type empty;
  config false;
  description
    "Present if the link is transitional, used as an
     alternative approach in lieu of inter-layer-lock-id
     for path computation in a TE topology covering multiple
     layers or multiple regions.";
  reference
    "RFC 5212: Requirements for GMPLS-Based Multi-Region and
     Multi-Layer Networks (MRN/MLN).
     RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
     for Multi-Layer and Multi-Region Networks (MLN/MRN).";
}
uses information-source-per-link-attributes;
list information-source-entry {
  key "information-source information-source-instance";
  config false;
  description
    "A list of information sources learned, including the one
     used.";
  uses information-source-per-link-attributes;
  uses te-link-info-attributes;
}
container recovery {
  config false;
  description
    "Status of the recovery process.";
  leaf restoration-status {
    type te-types:te-recovery-status;
    description
      "Restoration status.";
  }
  leaf protection-status {
    type te-types:te-recovery-status;
    description
      "Protection status.";
  }
}
container underlay {
  if-feature te-topology-hierarchy;
```

```
    config false;
    description "State attributes for te-link underlay.";
    leaf dynamic {
        type boolean;
        description
            "true if the underlay is dynamically created.";
    }
    leaf committed {
        type boolean;
        description
            "true if the underlay is committed.";
    }
}
} // te-link-state-derived

grouping te-link-underlay-attributes {
    description "Attributes for te-link underlay.";
    reference
        "RFC 4206: Label Switched Paths (LSP) Hierarchy with
        Generalized Multi-Protocol Label Switching (GMPLS)
        Traffic Engineering (TE)";
    leaf enabled {
        type boolean;
        description
            "'true' if the underlay is enabled.
            'false' if the underlay is disabled.";
    }
}
container primary-path {
    description
        "The service path on the underlay topology that
        supports this link.";
    uses nw:network-ref;
    list path-element {
        key "path-element-id";
        description
            "A list of path elements describing the service path.";
        leaf path-element-id {
            type uint32;
            description "To identify the element in a path.";
        }
    }
    uses te-path-element;
```

```
    }  
  } // primary-path  
  list backup-path {  
    key "index";  
    description  
      "A list of backup service paths on the underlay topology that  
      protect the underlay primary path. If the primary path is  
      not protected, the list contains zero elements. If the  
      primary path is protected, the list contains one or more  
      elements.";  
    leaf index {  
      type uint32;  
      description  
        "A sequence number to identify a backup path.";  
    }  
    uses nw:network-ref;  
    list path-element {  
      key "path-element-id";  
      description  
        "A list of path elements describing the backup service  
        path";  
      leaf path-element-id {  
        type uint32;  
        description "To identify the element in a path.";  
      }  
      uses te-path-element;  
    }  
  } // underlay-backup-path  
  leaf protection-type {  
    type identityref {  
      base te-types:lsp-protection-type;  
    }  
    description  
      "Underlay protection type desired for this link.";  
  }  
  container tunnel-termination-points {  
    description  
      "Underlay TTP (Tunnel Termination Points) desired for this  
      link.";  
    leaf source {  
      type binary;  
    }  
  }  
}
```



```
        description
            "Source tunnel termination point identifier.";
    }
    leaf destination {
        type binary;
        description
            "Destination tunnel termination point identifier.";
    }
}
container tunnels {
    description
        "Underlay TE tunnels supporting this TE link.";
    leaf sharing {
        type boolean;
        default true;
        description
            "'true' if the underlay tunnel can be shared with other
            TE links;
            'false' if the underlay tunnel is dedicated to this
            TE link.
            This leaf is the default option for all TE tunnels,
            and may be overridden by the per TE tunnel value.";
    }
    list tunnel {
        key "tunnel-name";
        description
            "Zero, one or more underlay TE tunnels that support this TE
            link.";
        leaf tunnel-name {
            type string;
            description
                "A tunnel name uniquely identifies an underlay TE tunnel,
                used together with the source-node of this link.
                The detailed information of this tunnel can be retrieved
                from the ietf-te model.";
            reference "RFC 3209";
        }
    }
    leaf sharing {
        type boolean;
        description
            "'true' if the underlay tunnel can be shared with other
```

```
        TE links;
        'false' if the underlay tunnel is dedicated to this
        TE link.";
    }
} // tunnel
} // tunnels
} // te-link-underlay-attributes

grouping te-node-augment {
  description
    "Augmentation for TE node.";
  uses te-node-config;
  uses te-node-state-derived;
  container statistics {
    config false;
    description
      "Statistics data.";
    uses statistics-per-node;
  } // statistics

  list tunnel-termination-point {
    key "tunnel-tp-id";
    description
      "A termination point can terminate a tunnel.";
    leaf tunnel-tp-id {
      type binary;
      description
        "Tunnel termination point identifier.";
    }

    uses te-node-tunnel-termination-point-config;
    leaf oper-status {
      type te-types:te-oper-status;
      config false;
      description
        "The current operational state of the tunnel
        termination point.";
    }
    uses geolocation-container;
    container statistics {
      config false;
```

```
    description
      "Statistics data.";
    uses statistics-per-ttp;
  } // statistics

  // Relations to other tunnel termination points
  list supporting-tunnel-termination-point {
    key "node-ref tunnel-tp-ref";
    description
      "Identifies the tunnel termination points, that this
       tunnel termination point is depending on.";
    leaf node-ref {
      type inet:uri;
      description
        "This leaf identifies the node in which the supporting
         tunnel termination point is present.
         This node is either the supporting node or a node in
         an underlay topology.";
    }
    leaf tunnel-tp-ref {
      type binary;
      description
        "Reference to a tunnel termination point, which is
         either in the supporting node or a node in an
         underlay topology.";
    }
  } // supporting-tunnel-termination-point
} // tunnel-termination-point
} // te-node-augment

grouping te-node-config {
  description "TE node configuration grouping.";
  leaf-list te-node-template {
    if-feature template;
    type leafref {
      path "../.../.../te/templates/node-template/name";
    }
    description
      "The reference to a TE node template.";
  }
  uses te-node-config-attributes;
```

```
    } // te-node-config

    grouping te-node-config-attributes {
      description "Configuration node attributes in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-connectivity-matrices;
        uses te-node-info-attributes;
      } // te-node-attributes
    } // te-node-config-attributes

    grouping te-node-config-attributes-template {
      description
        "Configuration node attributes for template in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-info-attributes;
      } // te-node-attributes
    } // te-node-config-attributes-template

    grouping te-node-connectivity-matrices {
      description "Connectivity matrix on a TE node.";
      container connectivity-matrices {
        description
          "Containing connectivity matrix on a TE node.";
        leaf number-of-entries {
          type uint16;
          description
            "The number of connectivity matrix entries.
             If this number is specified in the configuration request,
             the number is requested number of entries, which may not
```

```
        all be listed in the list;
        if this number is reported in the state data,
        the number is the current number of operational entries.";
    }
    uses te-types:label-set-info;
    uses connectivity-matrix-entry-path-attributes;
    list connectivity-matrix {
        key "id";
        description
            "Represents node's switching limitations, i.e. limitations
            in interconnecting network TE links across the node.";
        reference
            "RFC 7579: General Network Element Constraint Encoding
            for GMPLS-Controlled Networks.";
        leaf id {
            type uint32;
            description "Identifies the connectivity-matrix entry.";
        }
    } // connectivity-matrix
} // connectivity-matrices
} // te-node-connectivity-matrices

grouping te-node-connectivity-matrix-attributes {
    description
        "Termination point references of a connectivity matrix entry.";
    container from {
        description
            "Reference to source link termination point.";
        leaf tp-ref {
            type leafref {
                path "../..//../..//../nt:termination-point/nt:tp-id";
            }
            description
                "Relative reference to a termination point.";
        }
        uses te-types:label-set-info;
    }
    container to {
        description
            "Reference to destination link termination point.";
        leaf tp-ref {
```

```
    type leafref {
      path "../.../.../.../nt:termination-point/nt:tp-id";
    }
    description
      "Relative reference to a termination point.";
  }
  uses te-types:label-set-info;
}
uses connectivity-matrix-entry-path-attributes;
} // te-node-connectivity-matrix-attributes

grouping te-node-info-attributes {
  description
    "Advertised TE information attributes.";
  leaf domain-id {
    type uint32;
    description
      "Identifies the domain that this node belongs.
       This attribute is used to support inter-domain links.";
    reference
      "RFC 5152: A Per-Domain Path Computation Method for
       Establishing Inter-Domain Traffic Engineering (TE)
       Label Switched Paths (LSPs).
       RFC 5392: OSPF Extensions in Support of Inter-Autonomous
       System (AS) MPLS and GMPLS Traffic Engineering.
       RFC 5316: ISIS Extensions in Support of Inter-Autonomous
       System (AS) MPLS and GMPLS Traffic Engineering.";
  }
  leaf is-abstract {
    type empty;
    description
      "Present if the node is abstract, not present if the node
       is actual.";
  }
  leaf name {
    type string;
    description "Node name.";
  }
  leaf-list signaling-address {
    type inet:ip-address;
    description "Node signaling address.";
```

```
    }
    container underlay-topology {
      if-feature te-topology-hierarchy;
      description
        "When an abstract node encapsulates a topology,
         the attributes in this container point to said topology.";
      uses nw:network-ref;
    }
  } // te-node-info-attributes

grouping te-node-state-derived {
  description "Node state attributes in a TE topology.";
  leaf oper-status {
    type te-types:te-oper-status;
    config false;
    description
      "The current operational state of the node.";
  }
  uses geolocation-container;
  leaf is-multi-access-dr {
    type empty;
    config false;
    description
      "The presence of this attribute indicates that this TE node
       is a pseudonode elected as a designated router.";
    reference
      "RFC 3630: Traffic Engineering (TE) Extensions to OSPF
       Version 2.
       RFC 1195: Use of OSI IS-IS for Routing in TCP/IP and Dual
       Environments.";
  }
  uses information-source-per-node-attributes;
  list information-source-entry {
    key "information-source information-source-instance";
    config false;
    description
      "A list of information sources learned, including the one
       used.";
    uses information-source-per-node-attributes;
    uses te-node-connectivity-matrices;
    uses te-node-info-attributes;
  }
}
```

```
    }
  } // te-node-state-derived

  grouping te-node-tunnel-termination-point-config {
    description
      "Termination capability of a tunnel termination point on a
      TE node.";
    uses te-node-tunnel-termination-point-config-attributes;
    container local-link-connectivities {
      description
        "Containing local link connectivity list for
        a tunnel termination point on a TE node.";
      leaf number-of-entries {
        type uint16;
        description
          "The number of local link connectivity list entries.
          If this number is specified in the configuration request,
          the number is requested number of entries, which may not
          all be listed in the list;
          if this number is reported in the state data,
          the number is the current number of operational entries.";
      }
      uses te-types:label-set-info;
      uses connectivity-matrix-entry-path-attributes;
    } // local-link-connectivities
  } // te-node-tunnel-termination-point-config

  grouping te-node-tunnel-termination-point-config-attributes {
    description
      "Configuration attributes of a tunnel termination point on a
      TE node.";
    leaf admin-status {
      type te-types:te-admin-status;
      description
        "The administrative state of the tunnel termination point.";
    }
    leaf name {
      type string;
      description
        "A descriptive name for the tunnel termination point.";
    }
  }
```



```
leaf switching-capability {
  type identityref {
    base te-types:switching-capabilities;
  }
  description
    "Switching Capability for this interface.";
}
leaf encoding {
  type identityref {
    base te-types:lsp-encoding-types;
  }
  description
    "Encoding supported by this interface.";
}
leaf-list inter-layer-lock-id {
  type uint32;
  description
    "Inter layer lock ID, used for path computation in a TE
    topology covering multiple layers or multiple regions.";
  reference
    "RFC 5212: Requirements for GMPLS-Based Multi-Region and
    Multi-Layer Networks (MRN/MLN).
    RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
    for Multi-Layer and Multi-Region Networks (MLN/MRN).";
}
leaf protection-type {
  type identityref {
    base te-types:lsp-protection-type;
  }
  description
    "The protection type that this tunnel termination point
    is capable of.";
}

container client-layer-adaptation {
  description
    "Containing capability information to support a client layer
    adaption in multi-layer topology.";
  list switching-capability {
    key "switching-capability encoding";
    description
```

```
    "List of supported switching capabilities";
  reference
    "RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
    for Multi-Layer and Multi-Region Networks (MLN/MRN).
    RFC 4202: Routing Extensions in Support of
    Generalized Multi-Protocol Label Switching (GMPLS).";
  leaf switching-capability {
    type identityref {
      base te-types:switching-capabilities;
    }
    description
      "Switching Capability for the client layer adaption.";
  }
  leaf encoding {
    type identityref {
      base te-types:lsp-encoding-types;
    }
    description
      "Encoding supported by the client layer adaption.";
  }
  uses te-types:te-bandwidth;
}
}
} // te-node-tunnel-termination-point-config-attributes

grouping te-node-tunnel-termination-point-llc-list {
  description
    "Local link connectivity list of a tunnel termination
    point on a TE node.";
  list local-link-connectivity {
    key "link-tp-ref";
    description
      "The termination capabilities between
      tunnel-termination-point and link termination-point.
      The capability information can be used to compute
      the tunnel path.
      The Interface Adjustment Capability Descriptors (IACD)
      (defined in RFC 6001) on each link-tp can be derived from
      this local-link-connectivity list.";
  }
  reference
    "RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
```

```
        for Multi-Layer and Multi-Region Networks (MLN/MRN).";

    leaf link-tp-ref {
        type leafref {
            path "../..../nt:termination-point/nt:tp-id";
        }
        description
            "Link termination point.";
    }
    uses te-types:label-set-info;
    uses connectivity-matrix-entry-path-attributes;
} // local-link-connectivity
} // te-node-tunnel-termination-point-config

grouping te-path-element {
    description
        "A group of attributes defining an element in a TE path
        such as TE node, TE link, TE atomic resource or label.";
    uses te-types:explicit-route-hop;
} // te-path-element

grouping te-termination-point-augment {
    description
        "Augmentation for TE termination point.";
    leaf te-tp-id {
        type te-types:te-tp-id;
        description
            "An identifier to uniquely identify a TE termination
            point.";
    }
    container te {
        must "../te-tp-id";
        presence "TE support.";
        description
            "Indicates TE support.";

        uses te-termination-point-config;
        leaf oper-status {
            type te-types:te-oper-status;
            config false;
            description
```

```
        "The current operational state of the link termination
        point.";
    }
    uses geolocation-container;
} // te
} // te-termination-point-augment

grouping te-termination-point-config {
    description
        "TE termination point configuration grouping.";
    leaf admin-status {
        type te-types:te-admin-status;
        description
            "The administrative state of the link termination point.";
    }
    leaf name {
        type string;
        description
            "A descriptive name for the link termination point.";
    }
    uses interface-switching-capability-list;
    leaf inter-domain-plug-id {
        type binary;
        description
            "A topology-wide unique number that identifies on the
            network a connectivity supporting a given inter-domain
            TE link. This is more flexible alternative to specifying
            remote-te-node-id and remote-te-link-tp-id on a TE link,
            when the provider does not know remote-te-node-id and
            remote-te-link-tp-id or need to give client the
            flexibility to mix-n-match multiple topologies.";
    }
    leaf-list inter-layer-lock-id {
        type uint32;
        description
            "Inter layer lock ID, used for path computation in a TE
            topology covering multiple layers or multiple regions.";
        reference
            "RFC 5212: Requirements for GMPLS-Based Multi-Region and
            Multi-Layer Networks (MRN/MLN).
            RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
```

```
        for Multi-Layer and Multi-Region Networks (MLN/MRN).";
    }
} // te-termination-point-config

grouping te-topologies-augment {
  description
    "Augmentation for TE topologies.";
  container te {
    presence "TE support.";
    description
      "Indicates TE support.";

    container templates {
      description
        "Configuration parameters for templates used for TE
        topology.";

      list node-template {
        if-feature template;
        key "name";
        leaf name {
          type te-types:te-template-name;
          description
            "The name to identify a TE node template.";
        }
        description
          "The list of TE node templates used to define sharable
          and reusable TE node attributes.";
        uses template-attributes;
        uses te-node-config-attributes-template;
      } // node-template

      list link-template {
        if-feature template;
        key "name";
        leaf name {
          type te-types:te-template-name;
          description
            "The name to identify a TE link template.";
        }
        description

```

```
        "The list of TE link templates used to define sharable
        and reusable TE link attributes.";
    uses template-attributes;
    uses te-link-config-attributes;
    } // link-template
} // templates
} // te
} // te-topologies-augment

grouping te-topology-augment {
    description
        "Augmentation for TE topology.";
    uses te-types:te-topology-identifier;

    container te {
        must "../te-topology-identifier/provider-id"
            + " and ../te-topology-identifier/client-id"
            + " and ../te-topology-identifier/topology-id";
        presence "TE support.";
        description
            "Indicates TE support.";

        uses te-topology-config;
        uses geolocation-container;
    } // te
} // te-topology-augment

grouping te-topology-config {
    description
        "TE topology configuration grouping.";
    leaf name {
        type string;
        description
            "Name of the TE topology. This attribute is optional and can
            be specified by the operator to describe the TE topology,
            which can be useful when network-id is not descriptive
            and not modifiable because of being generated by the
            system.";
    }
    leaf preference {
        type uint8 {
```

```
        range "1..255";
    }
    description
        "Specifies a preference for this topology. A lower number
        indicates a higher preference.";
}
leaf optimization-criterion {
    type identityref {
        base te-types:objective-function-type;
    }
    description
        "Optimization criterion applied to this topology.";
    reference
        "RFC 3272: Overview and Principles of Internet Traffic
        Engineering.";
}
list nsrlg {
    if-feature nsrlg;
    key "id";
    description
        "List of NSRLGs (Not Sharing Risk Link Groups).";
    reference
        "RFC 4872: RSVP-TE Extensions in Support of End-to-End
        Generalized Multi-Protocol Label Switching (GMPLS)
        Recovery";
    leaf id {
        type uint32;
        description
            "Identify the NSRLG entry.";
    }
    leaf disjointness {
        type te-types:te-path-disjointness;
        description
            "The type of resource disjointness.";
    }
} // nsrlg
} // te-topology-config

grouping template-attributes {
    description
        "Common attributes for all templates.";
```

```
leaf priority {
  type uint16;
  description
    "The preference value to resolve conflicts between different
    templates. When two or more templates specify values for
    one configuration attribute, the value from the template
    with the highest priority is used.
    A lower number indicates a higher priority. The highest
    priority is 0.";
}
leaf reference-change-policy {
  type enumeration {
    enum no-action {
      description
        "When an attribute changes in this template, the
        configuration node referring to this template does
        not take any action.";
    }
    enum not-allowed {
      description
        "When any configuration object has a reference to this
        template, changing this template is not allowed.";
    }
    enum cascade {
      description
        "When an attribute changes in this template, the
        configuration object referring to this template applies
        the new attribute value to the corresponding
        configuration.";
    }
  }
  description
    "This attribute specifies the action taken to a configuration
    node that has a reference to this template.";
}
} // template-attributes

/*
 * Data nodes
 */
augment "/nw:networks/nw:network/nw:network-types" {
```



```
    description
      "Introduce new network type for TE topology.";
    container te-topology {
      presence "Indicates TE topology.";
      description
        "Its presence identifies the TE topology type.";
    }
  }

  augment "/nw:networks" {
    description
      "Augmentation parameters for TE topologies.";
    uses te-topologies-augment;
  }

  augment "/nw:networks/nw:network" {
    when "nw:network-types/tet:te-topology" {
      description
        "Augmentation parameters apply only for networks with
        TE topology type.";
    }
    description
      "Configuration parameters for TE topology.";
    uses te-topology-augment;
  }

  augment "/nw:networks/nw:network/nw:node" {
    when "../nw:network-types/tet:te-topology" {
      description
        "Augmentation parameters apply only for networks with
        TE topology type.";
    }
    description
      "Configuration parameters for TE at node level.";
    leaf te-node-id {
      type te-types:te-node-id;
      description
        "The identifier of a node in the TE topology.
        A node is specific to a topology to which it belongs.";
    }
  }
  container te {
```

```
    must "../te-node-id" {
      description
        "te-node-id is mandatory.";
    }
    must "count(..../nw:supporting-node)<=1" {
      description
        "For a node in a TE topology, there cannot be more
        than 1 supporting node. If multiple nodes are abstracted,
        the underlay-topology is used.";
    }
    presence "TE support.";
    description
      "Indicates TE support.";
    uses te-node-augment;
  } // te
}

augment "/nw:networks/nw:network/nt:link" {
  when "../nw:network-types/tet:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Configuration parameters for TE at link level.";
  container te {
    must "count(..../nt:supporting-link)<=1" {
      description
        "For a link in a TE topology, there cannot be more
        than 1 supporting link. If one or more link paths are
        abstracted, the underlay is used.";
    }
    presence "TE support.";
    description
      "Indicates TE support.";
    uses te-link-augment;
  } // te
}

augment "/nw:networks/nw:network/nw:node/"
  + "nt:termination-point" {
```

```
when "../../../nw:network-types/tet:te-topology" {
  description
    "Augmentation parameters apply only for networks with
    TE topology type.";
}
description
  "Configuration parameters for TE at termination point level.";
uses te-termination-point-augment;
}

augment
  "/nw:networks/nw:network/nt:link/te/bundle-stack-level/"
+ "bundle/bundled-links/bundled-link" {
  when "../../../nw:network-types/tet:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Augment TE link bundled link.";
  leaf src-tp-ref {
    type leafref {
      path "../../../nw:node[nw:node-id = "
        + "current()/../../../../nt:source/"
        + "nt:source-node]/"
        + "nt:termination-point/nt:tp-id";
      require-instance true;
    }
    description
      "Reference to another TE termination point on the
      same source node.";
  }
  leaf des-tp-ref {
    type leafref {
      path "../../../nw:node[nw:node-id = "
        + "current()/../../../../nt:destination/"
        + "nt:dest-node]/"
        + "nt:termination-point/nt:tp-id";
      require-instance true;
    }
    description

```

```
        "Reference to another TE termination point on the
        same destination node.";
    }
}

augment
  "/nw:networks/nw:network/nw:node/te/"
+ "information-source-entry/connectivity-matrices/"
+ "connectivity-matrix" {
  when "../..../nw:network-types/tet:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Augment TE node connectivity-matrix.";
  uses te-node-connectivity-matrix-attributes;
}

augment
  "/nw:networks/nw:network/nw:node/te/te-node-attributes/"
+ "connectivity-matrices/connectivity-matrix" {
  when "../..../nw:network-types/tet:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Augment TE node connectivity-matrix.";
  uses te-node-connectivity-matrix-attributes;
}

augment
  "/nw:networks/nw:network/nw:node/te/"
+ "tunnel-termination-point/local-link-connectivities" {
  when "../..../nw:network-types/tet:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
```

```
    "Augment TE node tunnel termination point LLCs
      (Local Link Connectivities).";
    uses te-node-tunnel-termination-point-llc-list;
  }
}
<CODE ENDS>
```

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

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- o /nw:networks/nw:network/nw:network-types/tet:te-topology  
This subtree specifies the TE topology type. Modifying the configurations can make TE topology type invalid. By such modifications, a malicious attacker may disable the TE capabilities on the related networks and cause traffic disrupted or misrouted.
- o /nw:networks/tet:te  
This subtree specifies the TE node templates and TE link templates. Modifying the configurations in this subtree will change the related future TE configurations. By such modifications, a malicious attacker may change the TE capabilities scheduled at a future time, to cause traffic disrupted or misrouted.

- o `/nw:networks/nw:network`  
This subtree specifies the topology-wide configurations, including the TE topology ID and topology-wide policies. Modifying the configurations in this subtree can add, remove, or modify TE topologies. By adding a TE topology, a malicious attacker may create an unauthorized traffic network. By removing or modifying a TE topology, a malicious attacker may cause traffic disabled or misrouted in the specified TE topology. Such traffic changes may also affect the traffic in the connected TE topologies.
- o `/nw:networks/nw:network/nw:node`  
This subtree specifies the configurations for TE nodes. Modifying the configurations in this subtree can add, remove, or modify TE nodes. By adding a TE node, a malicious attacker may create an unauthorized traffic path. By removing or modifying a TE node, a malicious attacker may cause traffic disabled or misrouted in the specified TE node. Such traffic changes may also affect the traffic on the surrounding TE nodes and TE links in this TE topology and the connected TE topologies.
- o `/nw:networks/nw:network/nt:link/tet:te`  
This subtree specifies the configurations for TE links. Modifying the configurations in this subtree can add, remove, or modify TE links. By adding a TE link, a malicious attacker may create an unauthorized traffic path. By removing or modifying a TE link, a malicious attacker may cause traffic disabled or misrouted on the specified TE link. Such traffic changes may also affect the traffic on the surrounding TE nodes and TE links in this TE topology and the connected TE topologies.
- o `/nw:networks/nw:network/nw:node/nt:termination-point`  
This subtree specifies the configurations of TE link termination points. Modifying the configurations in this subtree can add, remove, or modify TE link termination points. By adding a TE link termination point, a malicious attacker may create an unauthorized traffic path. By removing or modifying a TE link termination point, a malicious attacker may cause traffic disabled or misrouted on the specified TE link termination point. Such traffic changes may also affect the traffic on the surrounding TE nodes and TE links in this TE topology and the connected TE topologies.

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:

- o /nw:networks/nw:network/nw:network-types/tet:te-topology  
Unauthorized access to this subtree can disclose the TE topology type.
- o /nw:networks/tet:te  
Unauthorized access to this subtree can disclose the TE node templates and TE link templates.
- o /nw:networks/nw:network  
Unauthorized access to this subtree can disclose the topology-wide configurations, including the TE topology ID, the topology-wide policies, and the topology geolocation.
- o /nw:networks/nw:network/nw:node  
Unauthorized access to this subtree can disclose the operational state information of TE nodes.
- o /nw:networks/nw:network/nt:link/tet:te  
Unauthorized access to this subtree can disclose the operational state information of TE links.
- o /nw:networks/nw:network/nw:node/nt:termination-point  
Unauthorized access to this subtree can disclose the operational state information of TE link termination points.

## 9. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-te-topology  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-topology-state  
Registrant Contact: The IESG.  
XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC7950].

name: ietf-te-topology  
namespace: urn:ietf:params:xml:ns:yang:ietf-te-topology  
prefix: tet  
reference: RFC XXXX

name: ietf-te-topology-state  
namespace: urn:ietf:params:xml:ns:yang:ietf-te-topology-state  
prefix: tet-s  
reference: RFC XXXX

## 10. References

### 10.1. 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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3945] Mannie, E., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, DOI 10.17487/RFC3945, October 2004, <<https://www.rfc-editor.org/info/rfc3945>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7926] Farrel, A., Ed., Drake, J., Bitar, N., Swallow, G., Ceccarelli, D., and X. Zhang, "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", BCP 206, RFC 7926, DOI 10.17487/RFC7926, July 2016, <<https://www.rfc-editor.org/info/rfc7926>>.
- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [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>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [I-D.ietf-teas-yang-te-types]  
Saad, T., Gandhi, R., Liu, X., Beeram, V., and  
I. Bryskin, "Traffic Engineering Common YANG Types",  
draft-ietf-teas-yang-te-types-08 (work in progress),  
April 2019.

## 10.2. Informative References

- [G.709] ITU-T, "Interfaces for the optical transport network", ITU-T Recommendation G.709, June 2016.
- [G.805] ITU-T, "Generic functional architecture of transport networks", ITU-T Recommendation G.805, March 2000.
- [G.872] ITU-T, "Architecture of optical transport networks", ITU-T Recommendation G.872, January 2017.
- [G.8080] ITU-T, "Architecture for the automatically switched optical network", ITU-T Recommendation G.8080, February 2012.

- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <<https://www.rfc-editor.org/info/rfc1195>>.
- [RFC2702] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, DOI 10.17487/RFC2702, September 1999, <<https://www.rfc-editor.org/info/rfc2702>>.
- [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>>.
- [RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, DOI 10.17487/RFC3272, May 2002, <<https://www.rfc-editor.org/info/rfc3272>>.
- [RFC3471] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description", RFC 3471, DOI 10.17487/RFC3471, January 2003, <<https://www.rfc-editor.org/info/rfc3471>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC3785] Le Faucheur, F., Uppili, R., Vedrenne, A., Merckx, P., and T. Telkamp, "Use of Interior Gateway Protocol (IGP) Metric as a second MPLS Traffic Engineering (TE) Metric", BCP 87, RFC 3785, DOI 10.17487/RFC3785, May 2004, <<https://www.rfc-editor.org/info/rfc3785>>.
- [RFC4201] Kompella, K., Rekhter, Y., and L. Berger, "Link Bundling in MPLS Traffic Engineering (TE)", RFC 4201, DOI 10.17487/RFC4201, October 2005, <<https://www.rfc-editor.org/info/rfc4201>>.
- [RFC4202] Kompella, K., Ed. and Y. Rekhter, Ed., "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, DOI 10.17487/RFC4202, October 2005, <<https://www.rfc-editor.org/info/rfc4202>>.
- [RFC4203] Kompella, K., Ed. and Y. Rekhter, Ed., "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching

- (GMPLS)", RFC 4203, DOI 10.17487/RFC4203, October 2005, <<https://www.rfc-editor.org/info/rfc4203>>.
- [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, DOI 10.17487/RFC4206, October 2005, <<https://www.rfc-editor.org/info/rfc4206>>.
- [RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <<https://www.rfc-editor.org/info/rfc4872>>.
- [RFC5152] Vasseur, JP., Ed., Ayyangar, A., Ed., and R. Zhang, "A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs)", RFC 5152, DOI 10.17487/RFC5152, February 2008, <<https://www.rfc-editor.org/info/rfc5152>>.
- [RFC5212] Shiimoto, K., Papadimitriou, D., Le Roux, JL., Vigoureux, M., and D. Brungard, "Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)", RFC 5212, DOI 10.17487/RFC5212, July 2008, <<https://www.rfc-editor.org/info/rfc5212>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5316] Chen, M., Zhang, R., and X. Duan, "ISIS Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5316, DOI 10.17487/RFC5316, December 2008, <<https://www.rfc-editor.org/info/rfc5316>>.
- [RFC5329] Ishiguro, K., Manral, V., Davey, A., and A. Lindem, Ed., "Traffic Engineering Extensions to OSPF Version 3", RFC 5329, DOI 10.17487/RFC5329, September 2008, <<https://www.rfc-editor.org/info/rfc5329>>.
- [RFC5392] Chen, M., Zhang, R., and X. Duan, "OSPF Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5392, DOI 10.17487/RFC5392, January 2009, <<https://www.rfc-editor.org/info/rfc5392>>.
- [RFC6001] Papadimitriou, D., Vigoureux, M., Shiimoto, K., Brungard,

- D., and JL. Le Roux, "Generalized MPLS (GMPLS) Protocol Extensions for Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 6001, DOI 10.17487/RFC6001, October 2010, <<https://www.rfc-editor.org/info/rfc6001>>.
- [RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <<https://www.rfc-editor.org/info/rfc7308>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC7579] Bernstein, G., Ed., Lee, Y., Ed., Li, D., Imajuku, W., and J. Han, "General Network Element Constraint Encoding for GMPLS-Controlled Networks", RFC 7579, DOI 10.17487/RFC7579, June 2015, <<https://www.rfc-editor.org/info/rfc7579>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [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>>.
- [I-D.ietf-netconf-subscribed-notifications]  
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-23 (work in progress), February 2019.
- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.
- [I-D.liu-netmod-yang-schedule]  
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "A YANG Data Model for Configuration Scheduling", draft-liu-netmod-yang-schedule-05 (work in progress),

March 2018.

[I-D.ietf-ccamp-wson-yang]

Lee, Y., Dhody, D., Zhang, X., Guo, A., Lopezalvarez, V., King, D., Yoon, B., and R. Vilata, "A Yang Data Model for WSON Optical Networks", draft-ietf-ccamp-wson-yang-20 (work in progress), March 2019.

[I-D.ietf-ccamp-otn-topo-yang]

zhenghaomian@huawei.com, z., Guo, A., Busi, I., Sharma, A., Liu, X., Belotti, S., Xu, Y., Wang, L., and O. Dios, "A YANG Data Model for Optical Transport Network Topology", draft-ietf-ccamp-otn-topo-yang-06 (work in progress), February 2019.

[I-D.ietf-teas-yang-l3-te-topo]

Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Layer 3 TE Topologies", draft-ietf-teas-yang-l3-te-topo-04 (work in progress), March 2019.

[I-D.ietf-teas-te-topo-and-tunnel-modeling]

Bryskin, I., Beeram, V., Saad, T., and X. Liu, "TE Topology and Tunnel Modeling for Transport Networks", draft-ietf-teas-te-topo-and-tunnel-modeling-03 (work in progress), October 2018.

## 11. Acknowledgments

The authors would like to thank Lou Berger, Sue Hares, Mazen Khaddam, Cyril Margaria and Zafar Ali for participating in design discussions and providing valuable insights.

## Appendix A. Complete Model Tree Structure

```

module: ietf-te-topology
  augment /nw:networks/nw:network/nw:network-types:
    +--rw te-topology!
  augment /nw:networks:
    +--rw te!
      +--rw templates
        +--rw node-template* [name] {template}?
          +--rw name
          |   te-types:te-template-name
          +--rw priority?          uint16
          +--rw reference-change-policy?  enumeration
          +--rw te-node-attributes
            +--rw admin-status?          te-types:te-admin-status
            +--rw domain-id?             uint32
            +--rw is-abstract?           empty
            +--rw name?                  string
            +--rw signaling-address*     inet:ip-address
            +--rw underlay-topology {te-topology-hierarchy}?
              +--rw network-ref?
              |   -> /nw:networks/network/network-id
        +--rw link-template* [name] {template}?
          +--rw name
          |   te-types:te-template-name
          +--rw priority?          uint16
          +--rw reference-change-policy?  enumeration
          +--rw te-link-attributes
            +--rw access-type?
            |   te-types:te-link-access-type
            +--rw external-domain
            |   +--rw network-ref?
            |   |   -> /nw:networks/network/network-id
            |   +--rw remote-te-node-id?    te-types:te-node-id
            |   +--rw remote-te-link-tp-id? te-types:te-tp-id
            +--rw is-abstract?          empty
            +--rw name?                  string
            +--rw underlay {te-topology-hierarchy}?
              +--rw enabled?              boolean
              +--rw primary-path
              |   +--rw network-ref?

```

```

|         -> /nw:networks/network/network-id
+--rw path-element* [path-element-id]
|   +--rw path-element-id          uint32
|   +--rw (type)?
|     +--:(numbered-node-hop)
|       +--rw numbered-node-hop
|         +--rw node-id          te-node-id
|         +--rw hop-type?       te-hop-type
|     +--:(numbered-link-hop)
|       +--rw numbered-link-hop
|         +--rw link-tp-id       te-tp-id
|         +--rw hop-type?       te-hop-type
|         +--rw direction?
|           te-link-direction
|     +--:(unnumbered-link-hop)
|       +--rw unnumbered-link-hop
|         +--rw link-tp-id       te-tp-id
|         +--rw node-id          te-node-id
|         +--rw hop-type?       te-hop-type
|         +--rw direction?
|           te-link-direction
|     +--:(as-number)
|       +--rw as-number-hop
|         +--rw as-number        inet:as-number
|         +--rw hop-type?       te-hop-type
|     +--:(label)
|       +--rw label-hop
|         +--rw te-label
|           +--rw (technology)?
|             +--:(generic)
|               +--rw generic?
|                 rt-
types:generalized-label
|         +--rw direction?
|           te-label-direction
+--rw backup-path* [index]
|   +--rw index          uint32
|   +--rw network-ref?
|     -> /nw:networks/network/network-id
+--rw path-element* [path-element-id]
|   +--rw path-element-id          uint32

```

```

+--rw (type)?
+--:(numbered-node-hop)
|   +--rw numbered-node-hop
|       +--rw node-id      te-node-id
|       +--rw hop-type?    te-hop-type
+--:(numbered-link-hop)
|   +--rw numbered-link-hop
|       +--rw link-tp-id    te-tp-id
|       +--rw hop-type?    te-hop-type
|       +--rw direction?
|           te-link-direction
+--:(unnumbered-link-hop)
|   +--rw unnumbered-link-hop
|       +--rw link-tp-id    te-tp-id
|       +--rw node-id      te-node-id
|       +--rw hop-type?    te-hop-type
|       +--rw direction?
|           te-link-direction
+--:(as-number)
|   +--rw as-number-hop
|       +--rw as-number     inet:as-number
|       +--rw hop-type?    te-hop-type
+--:(label)
|   +--rw label-hop
|       +--rw te-label
|           +--rw (technology)?
|               +--:(generic)
|                   +--rw generic?
|                       rt-
types:generalized-label
|       +--rw direction?
|           te-label-direction
+--rw protection-type?          identityref
+--rw tunnel-termination-points
|   +--rw source?               binary
|   +--rw destination?         binary
+--rw tunnels
|   +--rw sharing?              boolean
|   +--rw tunnel* [tunnel-name]
|       +--rw tunnel-name      string
|       +--rw sharing?         boolean

```



```

+--rw admin-status?
|   te-types:te-admin-status
+--rw link-index?                               uint64
+--rw administrative-group?
|   te-types:admin-groups
+--rw interface-switching-capability*
|   [switching-capability encoding]
|   +--rw switching-capability identityref
|   +--rw encoding identityref
|   +--rw max-lsp-bandwidth* [priority]
|       +--rw priority uint8
|       +--rw te-bandwidth
|           +--rw (technology)?
|               +--:(generic)
|                   +--rw generic? te-bandwidth
+--rw label-restrictions
|   +--rw label-restriction* [index]
|       +--rw restriction? enumeration
|       +--rw index uint32
|       +--rw label-start
|           +--rw te-label
|               +--rw (technology)?
|                   +--:(generic)
|                       +--rw generic?
|                           rt-types:generalized-label
|               +--rw direction? te-label-direction
+--rw label-end
|   +--rw te-label
|       +--rw (technology)?
|           +--:(generic)
|               +--rw generic?
|                   rt-types:generalized-label
|       +--rw direction? te-label-direction
+--rw label-step
|   +--rw (technology)?
|       +--:(generic)
|           +--rw generic? int32
|   +--rw range-bitmap? yang:hex-string
+--rw link-protection-type? identityref
+--rw max-link-bandwidth
|   +--rw te-bandwidth

```

```

    |         +---rw (technology)?
    |         |         +---:(generic)
    |         |         +---rw generic?    te-bandwidth
+---rw max-resv-link-bandwidth
    |         +---rw te-bandwidth
    |         |         +---rw (technology)?
    |         |         +---:(generic)
    |         |         +---rw generic?    te-bandwidth
+---rw unreserved-bandwidth* [priority]
    |         +---rw priority            uint8
    |         +---rw te-bandwidth
    |         |         +---rw (technology)?
    |         |         +---:(generic)
    |         |         +---rw generic?    te-bandwidth
+---rw te-default-metric?                uint32
+---rw te-delay-metric?                  uint32
+---rw te-igp-metric?                    uint32
+---rw te-srlgs
    |   +---rw value*    te-types:srlg
+---rw te-nsrlgs {nsrlg}?
    |   +---rw id*      uint32
augment /nw:networks/nw:network:
+---rw te-topology-identifier
    |   +---rw provider-id?    te-global-id
    |   +---rw client-id?      te-global-id
    |   +---rw topology-id?    te-topology-id
+---rw te!
    |   +---rw name?            string
    |   +---rw preference?      uint8
    |   +---rw optimization-criterion?  identityref
    |   +---rw nsrlg* [id] {nsrlg}?
    |   |   +---rw id          uint32
    |   |   +---rw disjointness? te-types:te-path-disjointness
+---ro geolocation
    |   +---ro altitude?        int64
    |   +---ro latitude?        geographic-coordinate-degree
    |   +---ro longitude?       geographic-coordinate-degree
augment /nw:networks/nw:network/nw:node:
+---rw te-node-id?    te-types:te-node-id
+---rw te!
    |   +---rw te-node-template*

```

```

|         -> ../../../../te/templates/node-template/name
|         {template}?
+--rw te-node-attributes
|   +--rw admin-status?          te-types:te-admin-status
|   +--rw connectivity-matrices
|   |   +--rw number-of-entries?    uint16
|   |   +--rw label-restrictions
|   |   |   +--rw label-restriction* [index]
|   |   |   |   +--rw restriction?    enumeration
|   |   |   |   +--rw index          uint32
|   |   |   |   +--rw label-start
|   |   |   |   |   +--rw te-label
|   |   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   |   |   +--rw generic?
|   |   |   |   |   |   |   |   |   rt-types:generalized-label
|   |   |   |   |   |   |   |   |   +--rw direction?          te-label-direction
|   |   |   |   |   +--rw label-end
|   |   |   |   |   |   +--rw te-label
|   |   |   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   |   |   |   +--rw generic?
|   |   |   |   |   |   |   |   |   |   rt-types:generalized-label
|   |   |   |   |   |   |   |   |   |   +--rw direction?          te-label-direction
|   |   |   |   |   +--rw label-step
|   |   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   |   |   +--rw generic?    int32
|   |   |   |   |   +--rw range-bitmap?    yang:hex-string
|   |   +--rw is-allowed?          boolean
|   +--rw underlay {te-topology-hierarchy}?
|   |   +--rw enabled?              boolean
|   |   +--rw primary-path
|   |   |   +--rw network-ref?
|   |   |   |   -> /nw:networks/network/network-id
|   |   |   +--rw path-element* [path-element-id]
|   |   |   |   +--rw path-element-id          uint32
|   |   |   |   +--rw (type)?
|   |   |   |   |   +--:(numbered-node-hop)
|   |   |   |   |   |   +--rw numbered-node-hop
|   |   |   |   |   |   |   +--rw node-id          te-node-id

```



```
label |   |   |   |         +---rw direction?           te-link-direction
      |   |   |   |       +--:(unnumbered-link-hop)
      |   |   |   |     +---rw unnumbered-link-hop
      |   |   |   |        +---rw link-tp-id          te-tp-id
      |   |   |   |        +---rw node-id             te-node-id
      |   |   |   |        +---rw hop-type?            te-hop-type
      |   |   |   |        +---rw direction?          te-link-direction
      |   |   |   |       +--:(as-number)
      |   |   |   |     +---rw as-number-hop
      |   |   |   |        +---rw as-number            inet:as-number
      |   |   |   |        +---rw hop-type?            te-hop-type
      |   |   |   |       +--:(label)
      |   |   |   |     +---rw label-hop
      |   |   |   |        +---rw te-label
      |   |   |   |              +---rw (technology)?
      |   |   |   |                +--:(generic)
      |   |   |   |                  +---rw generic?
      |   |   |   |                    rt-types:generalized-
label |   |   |   |                                     |
      |   |   |   |               +---rw direction?
      |   |   |   |                                   te-label-direction
+---rw protection-type?                                identityref
+---rw tunnel-termination-points
|    +---rw source?          binary
|    +---rw destination?     binary
+---rw tunnels
    +---rw sharing?          boolean
    +---rw tunnel* [tunnel-name]
        +---rw tunnel-name    string
        +---rw sharing?        boolean
+---rw path-constraints
+---rw te-bandwidth
|    +---rw (technology)?
|    +--:(generic)
|        +---rw generic?      te-bandwidth
+---rw link-protection?            identityref
+---rw setup-priority?             uint8
+---rw hold-priority?              uint8
+---rw signaling-type?             identityref
+---rw path-metric-bounds
|    +---rw path-metric-bound* [metric-type]
```

```

|         +---rw metric-type      identityref
|         +---rw upper-bound?    uint64
+---rw path-affinities-values
|         +---rw path-affinities-value* [usage]
|         +---rw usage            identityref
|         +---rw value?          admin-groups
+---rw path-affinity-names
|         +---rw path-affinity-name* [usage]
|         +---rw usage            identityref
|         +---rw affinity-name* [name]
|         +---rw name             string
+---rw path-srlgs-lists
|         +---rw path-srlgs-list* [usage]
|         +---rw usage            identityref
|         +---rw values*          srlg
+---rw path-srlgs-names
|         +---rw path-srlgs-name* [usage]
|         +---rw usage            identityref
|         +---rw names*           string
+---rw disjointness?              te-path-disjointness
+---rw optimizations
+---rw (algorithm)?
|         +---:(metric) {path-optimization-metric}?
|         |         +---rw optimization-metric* [metric-type]
|         |         |         +---rw metric-type
|         |         |         |         identityref
|         |         +---rw weight?
|         |         |         uint8
|         +---rw explicit-route-exclude-objects
|         |         +---rw route-object-exclude-object*
|         |         |         [index]
|         |         +---rw index
|         |         |         uint32
|         +---rw (type)?
|         |         +---:(numbered-node-hop)
|         |         |         +---rw numbered-node-hop
|         |         |         |         +---rw node-id      te-node-id
|         |         |         |         +---rw hop-type?    te-hop-type
|         |         +---:(numbered-link-hop)
|         |         |         +---rw numbered-link-hop
|         |         |         |         +---rw link-tp-id    te-tp-id

```

```

+---rw hop-type?
|       te-hop-type
+---rw direction?
|       te-link-direction
+---:(unnumbered-link-hop)
+---rw unnumbered-link-hop
+---rw link-tp-id      te-tp-id
+---rw node-id
|       te-node-id
+---rw hop-type?
|       te-hop-type
+---rw direction?
|       te-link-direction
+---:(as-number)
+---rw as-number-hop
+---rw as-number
|       inet:as-number
+---rw hop-type?
|       te-hop-type
+---:(label)
+---rw label-hop
+---rw te-label
|       +---rw (technology)?
|       |       +---:(generic)
|       |       +---rw generic?
|       |       rt-
types:generalized-label
|       +---rw direction?
|       |       te-label-direction
+---:(srlg)
+---rw srlg
|       +---rw srlg?      uint32
+---rw explicit-route-include-objects
+---rw route-object-include-object*
|       [index]
+---rw index
|       uint32
+---rw (type)?
+---:(numbered-node-hop)
+---rw numbered-node-hop
|       +---rw node-id      te-node-id

```

```

+---rw hop-type?      te-hop-type
+---:(numbered-link-hop)
|   +---rw numbered-link-hop
|       +---rw link-tp-id      te-tp-id
|       +---rw hop-type?
|           |
|           te-hop-type
|       +---rw direction?
|           |
|           te-link-direction
+---:(unnumbered-link-hop)
|   +---rw unnumbered-link-hop
|       +---rw link-tp-id      te-tp-id
|       +---rw node-id
|           |
|           te-node-id
|       +---rw hop-type?
|           |
|           te-hop-type
|       +---rw direction?
|           |
|           te-link-direction
+---:(as-number)
|   +---rw as-number-hop
|       +---rw as-number
|           |
|           inet:as-number
|       +---rw hop-type?
|           |
|           te-hop-type
+---:(label)
|   +---rw label-hop
|       +---rw te-label
|           +---rw (technology)?
|               +---:(generic)
|                   +---rw generic?
|                       |
|                       rt-
|
|       +---rw direction?
|                   |
|                   te-label-direction
+---rw tiebreakers
|   +---rw tiebreaker* [tiebreaker-type]
|       +---rw tiebreaker-type      identityref
+---:(objective-function)
|   {path-optimization-objective-function}?
+---rw objective-function
|   +---rw objective-function-type?      identityref
+---ro path-properties

```



```

+--ro path-metric* [metric-type]
|   +--ro metric-type      identityref
|   +--ro accumulative-value?  uint64
+--ro path-affinities-values
|   +--ro path-affinities-value* [usage]
|   |   +--ro usage      identityref
|   |   +--ro value?    admin-groups
+--ro path-affinity-names
|   +--ro path-affinity-name* [usage]
|   |   +--ro usage      identityref
|   |   +--ro affinity-name* [name]
|   |   |   +--ro name    string
+--ro path-srlgs-lists
|   +--ro path-srlgs-list* [usage]
|   |   +--ro usage      identityref
|   |   +--ro values*    srlg
+--ro path-srlgs-names
|   +--ro path-srlgs-name* [usage]
|   |   +--ro usage      identityref
|   |   +--ro names*    string
+--ro path-route-objects
|   +--ro path-route-object* [index]
|   |   +--ro index                                uint32
|   |   +--ro (type)?
|   |   |   +--:(numbered-node-hop)
|   |   |   |   +--ro numbered-node-hop
|   |   |   |   |   +--ro node-id      te-node-id
|   |   |   |   |   +--ro hop-type?    te-hop-type
|   |   |   +--:(numbered-link-hop)
|   |   |   |   +--ro numbered-link-hop
|   |   |   |   |   +--ro link-tp-id    te-tp-id
|   |   |   |   |   +--ro hop-type?    te-hop-type
|   |   |   |   |   +--ro direction?    te-link-direction
|   |   |   +--:(unnumbered-link-hop)
|   |   |   |   +--ro unnumbered-link-hop
|   |   |   |   |   +--ro link-tp-id    te-tp-id
|   |   |   |   |   +--ro node-id      te-node-id
|   |   |   |   |   +--ro hop-type?    te-hop-type
|   |   |   |   |   +--ro direction?    te-link-direction
|   |   |   +--:(as-number)
|   |   |   |   +--ro as-number-hop

```

				+--ro as-number	inet:as-number
				+--ro hop-type?	te-hop-type
			+---:(label)		
			+--ro label-hop		
			+--ro te-label		
			+--ro (technology)?		
			+---:(generic)		
			+--ro generic?		
				rt-types:generalized-	
label					
				+--ro direction?	
				te-label-direction	
		+--rw connectivity-matrix*	[id]		
		+--rw id		uint32	
		+--rw from			
		+--rw tp-ref?		leafref	
		+--rw label-restrictions			
		+--rw label-restriction*	[index]		
		+--rw restriction?		enumeration	
		+--rw index		uint32	
		+--rw label-start			
		+--rw te-label			
		+--rw (technology)?			
		+---:(generic)			
		+--rw generic?			
				rt-types:generalized-	
label					
				+--rw direction?	
				te-label-direction	
		+--rw label-end			
		+--rw te-label			
		+--rw (technology)?			
		+---:(generic)			
		+--rw generic?			
				rt-types:generalized-	
label					
				+--rw direction?	
				te-label-direction	
		+--rw label-step			
		+--rw (technology)?			
		+---:(generic)			



```

+---rw numbered-node-hop
+---rw node-id          te-node-id
+---rw hop-type?       te-hop-type
+---:(numbered-link-hop)
+---rw numbered-link-hop
+---rw link-tp-id       te-tp-id
+---rw hop-type?       te-hop-type
+---rw direction?
+---rw te-link-direction
+---:(unnumbered-link-hop)
+---rw unnumbered-link-hop
+---rw link-tp-id       te-tp-id
+---rw node-id          te-node-id
+---rw hop-type?       te-hop-type
+---rw direction?
+---rw te-link-direction
+---:(as-number)
+---rw as-number-hop
+---rw as-number         inet:as-number
+---rw hop-type?       te-hop-type
+---:(label)
+---rw label-hop
+---rw te-label
+---rw (technology)?
+---:(generic)
+---rw generic?
+---rw rt-
types:generalized-label
+---rw direction?
+---rw te-label-direction
+---rw backup-path* [index]
+---rw index          uint32
+---rw network-ref?
+---rw -> /nw:networks/network/network-id
+---rw path-element* [path-element-id]
+---rw path-element-id          uint32
+---rw (type)?
+---:(numbered-node-hop)
+---rw numbered-node-hop
+---rw node-id          te-node-id
+---rw hop-type?       te-hop-type

```

```

+---:(numbered-link-hop)
+---rw numbered-link-hop
+---rw link-tp-id      te-tp-id
+---rw hop-type?      te-hop-type
+---rw direction?
+---rw te-link-direction
+---:(unnumbered-link-hop)
+---rw unnumbered-link-hop
+---rw link-tp-id      te-tp-id
+---rw node-id         te-node-id
+---rw hop-type?      te-hop-type
+---rw direction?
+---rw te-link-direction
+---:(as-number)
+---rw as-number-hop
+---rw as-number       inet:as-number
+---rw hop-type?      te-hop-type
+---:(label)
+---rw label-hop
+---rw te-label
+---rw (technology)?
+---:(generic)
+---rw generic?
+---rw rt-
types:generalized-label
+---rw direction?
+---rw te-label-direction
+---rw protection-type?      identityref
+---rw tunnel-termination-points
+---rw source?      binary
+---rw destination? binary
+---rw tunnels
+---rw sharing?     boolean
+---rw tunnel* [tunnel-name]
+---rw tunnel-name  string
+---rw sharing?     boolean
+---rw path-constraints
+---rw te-bandwidth
+---rw (technology)?
+---:(generic)
+---rw generic?      te-bandwidth

```

```

+--rw link-protection?          identityref
+--rw setup-priority?           uint8
+--rw hold-priority?            uint8
+--rw signaling-type?           identityref
+--rw path-metric-bounds
|   +--rw path-metric-bound* [metric-type]
|   |   +--rw metric-type      identityref
|   |   +--rw upper-bound?    uint64
+--rw path-affinities-values
|   +--rw path-affinities-value* [usage]
|   |   +--rw usage           identityref
|   |   +--rw value?         admin-groups
+--rw path-affinity-names
|   +--rw path-affinity-name* [usage]
|   |   +--rw usage           identityref
|   |   +--rw affinity-name* [name]
|   |   |   +--rw name        string
+--rw path-srlgs-lists
|   +--rw path-srlgs-list* [usage]
|   |   +--rw usage           identityref
|   |   +--rw values*        srlg
+--rw path-srlgs-names
|   +--rw path-srlgs-name* [usage]
|   |   +--rw usage           identityref
|   |   +--rw names*         string
+--rw disjointness?
|   te-path-disjointness
+--rw optimizations
|   +--rw (algorithm)?
|   |   +--:(metric) {path-optimization-metric}?
|   |   |   +--rw optimization-metric* [metric-type]
|   |   |   |   +--rw metric-type
|   |   |   |   |   identityref
|   |   |   |   +--rw weight?
|   |   |   |   |   uint8
|   |   |   |   +--rw explicit-route-exclude-objects
|   |   |   |   |   +--rw route-object-exclude-object*
|   |   |   |   |   |   [index]
|   |   |   |   |   |   +--rw index
|   |   |   |   |   |   |   uint32
|   |   |   |   +--rw (type)?

```

							+--:(numbered-node-hop)
							+---rw numbered-node-hop
							+---rw node-id
							te-node-id
							+---rw hop-type?
							te-hop-type
							+--:(numbered-link-hop)
							+---rw numbered-link-hop
							+---rw link-tp-id
							te-tp-id
							+---rw hop-type?
							te-hop-type
							+---rw direction?
							te-link-direction
							+--:(unnumbered-link-hop)
							+---rw unnumbered-link-hop
							+---rw link-tp-id
							te-tp-id
							+---rw node-id
							te-node-id
							+---rw hop-type?
							te-hop-type
							+---rw direction?
							te-link-direction
							+--:(as-number)
							+---rw as-number-hop
							+---rw as-number
							inet:as-number
							+---rw hop-type?
							te-hop-type
							+--:(label)
							+---rw label-hop
							+---rw te-label
							+---rw (technology)?
							+--:(generic)
							+---rw generic?
							rt-
types:generalized-label							
							+---rw direction?
direction							te-label-

```

+--:(srlg)
  +--rw srlg
    +--rw srlg? uint32
+--rw explicit-route-include-objects
  +--rw route-object-include-object*
    [index]
  +--rw index
    | uint32
  +--rw (type)?
    +--:(numbered-node-hop)
      +--rw numbered-node-hop
        +--rw node-id
          | te-node-id
        +--rw hop-type?
          | te-hop-type
    +--:(numbered-link-hop)
      +--rw numbered-link-hop
        +--rw link-tp-id
          | te-tp-id
        +--rw hop-type?
          | te-hop-type
        +--rw direction?
          | te-link-direction
    +--:(unnumbered-link-hop)
      +--rw unnumbered-link-hop
        +--rw link-tp-id
          | te-tp-id
        +--rw node-id
          | te-node-id
        +--rw hop-type?
          | te-hop-type
        +--rw direction?
          | te-link-direction
    +--:(as-number)
      +--rw as-number-hop
        +--rw as-number
          | inet:as-number
        +--rw hop-type?
          | te-hop-type
    +--:(label)
      +--rw label-hop

```



```

types:generalized-label
direction
function}?

+--rw te-label
+--rw (technology)?
+--:(generic)
+--rw generic?
rt-

+--rw direction?
te-label-

+--rw tiebreakers
+--rw tiebreaker* [tiebreaker-type]
+--rw tiebreaker-type identityref
+--:(objective-function)
{path-optimization-objective-

+--rw objective-function
+--rw objective-function-type?
identityref

+--ro path-properties
+--ro path-metric* [metric-type]
| +--ro metric-type identityref
| +--ro accumulative-value? uint64
+--ro path-affinities-values
+--ro path-affinities-value* [usage]
+--ro usage identityref
+--ro value? admin-groups
+--ro path-affinity-names
+--ro path-affinity-name* [usage]
+--ro usage identityref
+--ro affinity-name* [name]
+--ro name string
+--ro path-srlgs-lists
+--ro path-srlgs-list* [usage]
+--ro usage identityref
+--ro values* srlg
+--ro path-srlgs-names
+--ro path-srlgs-name* [usage]
+--ro usage identityref
+--ro names* string
+--ro path-route-objects
+--ro path-route-object* [index]

```

```

+--ro index                               uint32
+--ro (type)?
  +--:(numbered-node-hop)
    +--ro numbered-node-hop
      +--ro node-id      te-node-id
      +--ro hop-type?    te-hop-type
  +--:(numbered-link-hop)
    +--ro numbered-link-hop
      +--ro link-tp-id    te-tp-id
      +--ro hop-type?    te-hop-type
      +--ro direction?
        te-link-direction
  +--:(unnumbered-link-hop)
    +--ro unnumbered-link-hop
      +--ro link-tp-id    te-tp-id
      +--ro node-id      te-node-id
      +--ro hop-type?    te-hop-type
      +--ro direction?
        te-link-direction
  +--:(as-number)
    +--ro as-number-hop
      +--ro as-number    inet:as-number
      +--ro hop-type?    te-hop-type
  +--:(label)
    +--ro label-hop
      +--ro te-label
        +--ro (technology)?
          +--:(generic)
            +--ro generic?
              rt-
          +--ro direction?
            te-label-direction
types:generalized-label
+--rw domain-id?                uint32
+--rw is-abstract?              empty
+--rw name?                     string
+--rw signaling-address*        inet:ip-address
+--rw underlay-topology {te-topology-hierarchy}?
  +--rw network-ref?    -> /nw:networks/network/network-id
+--ro oper-status?            te-types:te-oper-status
+--ro geolocation

```

```

    |   +---ro altitude?      int64
    |   +---ro latitude?     geographic-coordinate-degree
    |   +---ro longitude?    geographic-coordinate-degree
+---ro is-multi-access-dr?      empty
+---ro information-source?      te-info-source
+---ro information-source-instance? string
+---ro information-source-state
    |   +---ro credibility-preference? uint16
    |   +---ro logical-network-element? string
    |   +---ro network-instance?      string
    |   +---ro topology
    |       +---ro node-ref?      leafref
    |       +---ro network-ref?   -> /nw:networks/network/network-id
+---ro information-source-entry*
    |   [information-source information-source-instance]
    |   +---ro information-source      te-info-source
    |   +---ro information-source-instance string
    |   +---ro information-source-state
    |       +---ro credibility-preference? uint16
    |       +---ro logical-network-element? string
    |       +---ro network-instance?      string
    |       +---ro topology
    |           +---ro node-ref?      leafref
    |           +---ro network-ref?   -> /nw:networks/network/network-id
+---ro connectivity-matrices
    |   +---ro number-of-entries?      uint16
    |   +---ro label-restrictions
    |       +---ro label-restriction* [index]
    |           +---ro restriction?    enumeration
    |           +---ro index          uint32
    |           +---ro label-start
    |               +---ro te-label
    |                   +---ro (technology)?
    |                       +---:(generic)
    |                           +---ro generic?
    |                               rt-types:generalized-label
    |               +---ro direction?    te-label-direction
    |   +---ro label-end
    |       +---ro te-label
    |           +---ro (technology)?

```

```

+---:(generic)
+---ro generic?
+---ro rt-types:generalized-label
+---ro direction?      te-label-direction
+---ro label-step
+---ro (technology)?
+---:(generic)
+---ro generic?      int32
+---ro range-bitmap?  yang:hex-string
+---ro is-allowed?    boolean
+---ro underlay {te-topology-hierarchy}?
+---ro enabled?      boolean
+---ro primary-path
+---ro network-ref?
+---ro -> /nw:networks/network/network-id
+---ro path-element* [path-element-id]
+---ro path-element-id      uint32
+---ro (type)?
+---:(numbered-node-hop)
+---ro numbered-node-hop
+---ro node-id      te-node-id
+---ro hop-type?    te-hop-type
+---:(numbered-link-hop)
+---ro numbered-link-hop
+---ro link-tp-id    te-tp-id
+---ro hop-type?    te-hop-type
+---ro direction?    te-link-direction
+---:(unnumbered-link-hop)
+---ro unnumbered-link-hop
+---ro link-tp-id    te-tp-id
+---ro node-id      te-node-id
+---ro hop-type?    te-hop-type
+---ro direction?    te-link-direction
+---:(as-number)
+---ro as-number-hop
+---ro as-number      inet:as-number
+---ro hop-type?      te-hop-type
+---:(label)
+---ro label-hop
+---ro te-label
+---ro (technology)?

```

					<pre>       +--:(generic)         +--ro generic?           rt-types:generalized- </pre>
label					<pre>       +--ro direction?         te-label-direction     +--ro backup-path* [index]       +--ro index          uint32       +--ro network-ref?                   -&gt; /nw:networks/network/network-id       +--ro path-element* [path-element-id]         +--ro path-element-id          uint32         +--ro (type)?           +--:(numbered-node-hop)             +--ro numbered-node-hop               +--ro node-id      te-node-id               +--ro hop-type?    te-hop-type           +--:(numbered-link-hop)             +--ro numbered-link-hop               +--ro link-tp-id    te-tp-id               +--ro hop-type?    te-hop-type               +--ro direction?   te-link-direction           +--:(unnumbered-link-hop)             +--ro unnumbered-link-hop               +--ro link-tp-id    te-tp-id               +--ro node-id      te-node-id               +--ro hop-type?    te-hop-type               +--ro direction?   te-link-direction           +--:(as-number)             +--ro as-number-hop               +--ro as-number    inet:as-number               +--ro hop-type?    te-hop-type           +--:(label)             +--ro label-hop               +--ro te-label                 +--ro (technology)?                   +--:(generic)                     +--ro generic?                       rt-types:generalized- </pre>
label					<pre>       +--ro direction? </pre>

```

|                                     te-label-direction
+--ro protection-type?               identityref
+--ro tunnel-termination-points
|   +--ro source?                    binary
|   +--ro destination?               binary
+--ro tunnels
|   +--ro sharing?                   boolean
|   +--ro tunnel* [tunnel-name]
|       +--ro tunnel-name            string
|       +--ro sharing?               boolean
+--ro path-constraints
|   +--ro te-bandwidth
|       +--ro (technology)?
|           +--:(generic)
|               +--ro generic?       te-bandwidth
+--ro link-protection?               identityref
+--ro setup-priority?                uint8
+--ro hold-priority?                 uint8
+--ro signaling-type?                identityref
+--ro path-metric-bounds
|   +--ro path-metric-bound* [metric-type]
|       +--ro metric-type            identityref
|       +--ro upper-bound?           uint64
+--ro path-affinities-values
|   +--ro path-affinities-value* [usage]
|       +--ro usage                  identityref
|       +--ro value?                 admin-groups
+--ro path-affinity-names
|   +--ro path-affinity-name* [usage]
|       +--ro usage                  identityref
|       +--ro affinity-name* [name]
|           +--ro name               string
+--ro path-srlgs-lists
|   +--ro path-srlgs-list* [usage]
|       +--ro usage                  identityref
|       +--ro values*                srlg
+--ro path-srlgs-names
|   +--ro path-srlgs-name* [usage]
|       +--ro usage                  identityref
|       +--ro names*                 string
+--ro disjointness?                  te-path-disjointness

```

```

+--ro optimizations
  +--ro (algorithm)?
    +--:(metric) {path-optimization-metric}?
      +--ro optimization-metric* [metric-type]
        +--ro metric-type
          | identityref
        +--ro weight?
          | uint8
        +--ro explicit-route-exclude-objects
          +--ro route-object-exclude-object*
            [index]
          +--ro index
            | uint32
          +--ro (type)?
            +--:(numbered-node-hop)
              +--ro numbered-node-hop
                +--ro node-id      te-node-id
                +--ro hop-type?   te-hop-type
            +--:(numbered-link-hop)
              +--ro numbered-link-hop
                +--ro link-tp-id   te-tp-id
                +--ro hop-type?
                | te-hop-type
                +--ro direction?
                  te-link-direction
            +--:(unnumbered-link-hop)
              +--ro unnumbered-link-hop
                +--ro link-tp-id   te-tp-id
                +--ro node-id
                | te-node-id
                +--ro hop-type?
                | te-hop-type
                +--ro direction?
                  te-link-direction
            +--:(as-number)
              +--ro as-number-hop
                +--ro as-number
                | inet:as-number
                +--ro hop-type?
                  te-hop-type
            +--:(label)

```

					+--ro label-hop
					+--ro te-label
					+--ro (technology)?
					+--:(generic)
					+--ro generic?
					rt-
types:	generalized-label				
					+--ro direction?
					te-label-direction
					+--:(srlg)
					+--ro srlg
					+--ro srlg?     uint32
					+--ro explicit-route-include-objects
					+--ro route-object-include-object*
					[index]
					+--ro index
					uint32
					+--ro (type)?
					+--:(numbered-node-hop)
					+--ro numbered-node-hop
					+--ro node-id     te-node-id
					+--ro hop-type?   te-hop-type
					+--:(numbered-link-hop)
					+--ro numbered-link-hop
					+--ro link-tp-id     te-tp-id
					+--ro hop-type?
					te-hop-type
					+--ro direction?
					te-link-direction
					+--:(unnumbered-link-hop)
					+--ro unnumbered-link-hop
					+--ro link-tp-id     te-tp-id
					+--ro node-id
					te-node-id
					+--ro hop-type?
					te-hop-type
					+--ro direction?
					te-link-direction
					+--:(as-number)
					+--ro as-number-hop
					+--ro as-number



[illegible]

			<pre> +--ro path-route-objects   +--ro path-route-object* [index]     +--ro index                               uint32     +--ro (type)?       +--:(numbered-node-hop)         +--ro numbered-node-hop           +--ro node-id      te-node-id           +--ro hop-type?    te-hop-type       +--:(numbered-link-hop)         +--ro numbered-link-hop           +--ro link-tp-id    te-tp-id           +--ro hop-type?    te-hop-type           +--ro direction?   te-link-direction       +--:(unnumbered-link-hop)         +--ro unnumbered-link-hop           +--ro link-tp-id    te-tp-id           +--ro node-id      te-node-id           +--ro hop-type?    te-hop-type           +--ro direction?   te-link-direction       +--:(as-number)         +--ro as-number-hop           +--ro as-number    inet:as-number           +--ro hop-type?    te-hop-type       +--:(label)         +--ro label-hop           +--ro te-label             +--ro (technology)?               +--:(generic)                 +--ro generic?                   rt-types:generalized- </pre>
label			<pre>           +--ro direction?             te-label-direction +--ro connectivity-matrix* [id]   +--ro id                               uint32   +--ro from     +--ro tp-ref?                        leafref     +--ro label-restrictions       +--ro label-restriction* [index]         +--ro restriction?              enumeration         +--ro index                     uint32 </pre>

				<pre> +--ro label-start     +--ro te-label         +--ro (technology)?             +--:(generic)                 +--ro generic?                     rt-types:generalized- </pre>
label				<pre>     +--ro direction?         te-label-direction +--ro label-end     +--ro te-label         +--ro (technology)?             +--:(generic)                 +--ro generic?                     rt-types:generalized- </pre>
label				<pre>     +--ro direction?         te-label-direction +--ro label-step     +--ro (technology)?         +--:(generic)             +--ro generic?   int32 +--ro range-bitmap?   yang:hex-string </pre>
			<pre> +--ro to +--ro tp-ref?           leafref +--ro label-restrictions     +--ro label-restriction* [index]         +--ro restriction?   enumeration         +--ro index         uint32 +--ro label-start     +--ro te-label         +--ro (technology)?             +--:(generic)                 +--ro generic?                     rt-types:generalized- </pre>	
label				<pre>     +--ro direction?         te-label-direction +--ro label-end     +--ro te-label         +--ro (technology)? </pre>

					<pre> +--:(generic)   +--ro generic?     rt-types:generalized- </pre>
label					<pre> +--ro direction?   te-label-direction +--ro label-step   +--ro (technology)?     +--:(generic)       +--ro generic?    int32       +--ro range-bitmap? yang:hex-string +--ro is-allowed?      boolean +--ro underlay {te-topology-hierarchy}?   +--ro enabled?      boolean   +--ro primary-path     +--ro network-ref?                 -&gt; /nw:networks/network/network-id +--ro path-element* [path-element-id]   +--ro path-element-id      uint32   +--ro (type)?     +--:(numbered-node-hop)       +--ro numbered-node-hop         +--ro node-id      te-node-id         +--ro hop-type?    te-hop-type     +--:(numbered-link-hop)       +--ro numbered-link-hop         +--ro link-tp-id    te-tp-id         +--ro hop-type?    te-hop-type         +--ro direction?           te-link-direction     +--:(unnumbered-link-hop)       +--ro unnumbered-link-hop         +--ro link-tp-id    te-tp-id         +--ro node-id      te-node-id         +--ro hop-type?    te-hop-type         +--ro direction?           te-link-direction     +--:(as-number)       +--ro as-number-hop         +--ro as-number    inet:as-number         +--ro hop-type?    te-hop-type </pre>

```

+---:(label)
+---ro label-hop
+---ro te-label
+---ro (technology)?
+---:(generic)
+---ro generic?
rt-

types:generalized-label

+---ro direction?
te-label-direction
+---ro backup-path* [index]
+---ro index uint32
+---ro network-ref?
| -> /nw:networks/network/network-id
+---ro path-element* [path-element-id]
+---ro path-element-id uint32
+---ro (type)?
+---:(numbered-node-hop)
+---ro numbered-node-hop
+---ro node-id te-node-id
+---ro hop-type? te-hop-type
+---:(numbered-link-hop)
+---ro numbered-link-hop
+---ro link-tp-id te-tp-id
+---ro hop-type? te-hop-type
+---ro direction?
te-link-direction
+---:(unnumbered-link-hop)
+---ro unnumbered-link-hop
+---ro link-tp-id te-tp-id
+---ro node-id te-node-id
+---ro hop-type? te-hop-type
+---ro direction?
te-link-direction
+---:(as-number)
+---ro as-number-hop
+---ro as-number inet:as-number
+---ro hop-type? te-hop-type
+---:(label)
+---ro label-hop
+---ro te-label

```

```

types:generalized-label
    +--ro (technology)?
    |   +--:(generic)
    |   +--ro generic?
    |       rt-
    +--ro direction?
    |   te-label-direction
    +--ro protection-type?
    |   identityref
    +--ro tunnel-termination-points
    |   +--ro source?
    |       binary
    |   +--ro destination?
    |       binary
    +--ro tunnels
    |   +--ro sharing?
    |       boolean
    |   +--ro tunnel* [tunnel-name]
    |       +--ro tunnel-name
    |           string
    |       +--ro sharing?
    |           boolean
    +--ro path-constraints
    |   +--ro te-bandwidth
    |       +--ro (technology)?
    |       +--:(generic)
    |       +--ro generic?
    |           te-bandwidth
    |   +--ro link-protection?
    |       identityref
    |   +--ro setup-priority?
    |       uint8
    |   +--ro hold-priority?
    |       uint8
    |   +--ro signaling-type?
    |       identityref
    |   +--ro path-metric-bounds
    |       +--ro path-metric-bound* [metric-type]
    |       +--ro metric-type
    |           identityref
    |       +--ro upper-bound?
    |           uint64
    |   +--ro path-affinities-values
    |       +--ro path-affinities-value* [usage]
    |       +--ro usage
    |           identityref
    |       +--ro value?
    |           admin-groups
    |   +--ro path-affinity-names
    |       +--ro path-affinity-name* [usage]
    |       +--ro usage
    |           identityref
    |       +--ro affinity-name* [name]
    |       +--ro name
    |           string
    |   +--ro path-srlgs-lists
    |       +--ro path-srlgs-list* [usage]
    |       +--ro usage
    |           identityref

```

```

|         +---ro values*    srlg
+---ro path-srlgs-names
|         +---ro path-srlgs-name* [usage]
|         +---ro usage      identityref
|         +---ro names*     string
+---ro disjointness?
|         te-path-disjointness
+---ro optimizations
+---ro (algorithm)?
|         +---:(metric) {path-optimization-metric}?
|         |         +---ro optimization-metric* [metric-type]
|         |         |         +---ro metric-type
|         |         |         |         identityref
|         |         |         +---ro weight?
|         |         |         |         uint8
|         |         |         +---ro explicit-route-exclude-objects
|         |         |         |         +---ro route-object-exclude-object*
|         |         |         |         |         [index]
|         |         |         |         +---ro index
|         |         |         |         |         uint32
|         |         |         |         +---ro (type)?
|         |         |         |         |         +---:(numbered-node-hop)
|         |         |         |         |         |         +---ro numbered-node-hop
|         |         |         |         |         |         |         +---ro node-id
|         |         |         |         |         |         |         |         te-node-id
|         |         |         |         |         |         |         +---ro hop-type?
|         |         |         |         |         |         |         |         te-hop-type
|         |         |         |         |         |         +---:(numbered-link-hop)
|         |         |         |         |         |         |         +---ro numbered-link-hop
|         |         |         |         |         |         |         |         +---ro link-tp-id
|         |         |         |         |         |         |         |         |         te-tp-id
|         |         |         |         |         |         |         |         +---ro hop-type?
|         |         |         |         |         |         |         |         |         te-hop-type
|         |         |         |         |         |         |         |         +---ro direction?
|         |         |         |         |         |         |         |         |         te-link-direction
|         |         |         |         |         |         +---:(unnumbered-link-hop)
|         |         |         |         |         |         |         +---ro unnumbered-link-hop
|         |         |         |         |         |         |         |         +---ro link-tp-id
|         |         |         |         |         |         |         |         |         te-tp-id
|         |         |         |         |         |         |         +---ro node-id
|         |         |         |         |         |         |         |         te-node-id

```

							+--ro hop-type?
							te-hop-type
						+--ro direction?	
							te-link-direction
						+---:(as-number)	
						+--ro as-number-hop	
						+--ro as-number	
							inet:as-number
						+--ro hop-type?	
							te-hop-type
						+---:(label)	
						+--ro label-hop	
						+--ro te-label	
						+--ro (technology)?	
							+---:(generic)
							+--ro generic?
							rt-
types:generalized-label							
						+--ro direction?	
							te-label-
direction							
						+---:(srlg)	
						+--ro srlg	
						+--ro srlg?	uint32
						+--ro explicit-route-include-objects	
						+--ro route-object-include-object*	
						[index]	
						+--ro index	
							uint32
						+--ro (type)?	
						+---:(numbered-node-hop)	
						+--ro numbered-node-hop	
						+--ro node-id	
							te-node-id
						+--ro hop-type?	
							te-hop-type
						+---:(numbered-link-hop)	
						+--ro numbered-link-hop	
						+--ro link-tp-id	
							te-tp-id
						+--ro hop-type?	



							te-hop-type
							+--ro direction?
							te-link-direction
							+--:(unnumbered-link-hop)
							+--ro unnumbered-link-hop
							+--ro link-tp-id
							te-tp-id
							+--ro node-id
							te-node-id
							+--ro hop-type?
							te-hop-type
							+--ro direction?
							te-link-direction
							+--:(as-number)
							+--ro as-number-hop
							+--ro as-number
							inet:as-number
							+--ro hop-type?
							te-hop-type
							+--:(label)
							+--ro label-hop
							+--ro te-label
							+--ro (technology)?
							+--:(generic)
							+--ro generic?
							rt-
types:generalized-label							
							+--ro direction?
							te-label-
direction							
							+--ro tiebreakers
							+--ro tiebreaker* [tiebreaker-type]
							+--ro tiebreaker-type identityref
							+--:(objective-function)
							{path-optimization-objective-
function}?							
							+--ro objective-function
							+--ro objective-function-type?
							identityref
							+--ro path-properties
							+--ro path-metric* [metric-type]

```

|   +--ro metric-type          identityref
|   +--ro accumulative-value?  uint64
+--ro path-affinities-values
|   +--ro path-affinities-value* [usage]
|       +--ro usage          identityref
|       +--ro value?         admin-groups
+--ro path-affinity-names
|   +--ro path-affinity-name* [usage]
|       +--ro usage          identityref
|       +--ro affinity-name* [name]
|           +--ro name        string
+--ro path-srlgs-lists
|   +--ro path-srlgs-list* [usage]
|       +--ro usage          identityref
|       +--ro values*        srlg
+--ro path-srlgs-names
|   +--ro path-srlgs-name* [usage]
|       +--ro usage          identityref
|       +--ro names*         string
+--ro path-route-objects
|   +--ro path-route-object* [index]
|       +--ro index          uint32
|       +--ro (type)?
|           +--:(numbered-node-hop)
|               +--ro numbered-node-hop
|                   +--ro node-id      te-node-id
|                   +--ro hop-type?    te-hop-type
|           +--:(numbered-link-hop)
|               +--ro numbered-link-hop
|                   +--ro link-tp-id    te-tp-id
|                   +--ro hop-type?     te-hop-type
|                   +--ro direction?
|                       te-link-direction
|           +--:(unnumbered-link-hop)
|               +--ro unnumbered-link-hop
|                   +--ro link-tp-id    te-tp-id
|                   +--ro node-id      te-node-id
|                   +--ro hop-type?     te-hop-type
|                   +--ro direction?
|                       te-link-direction
|           +--:(as-number)

```

```

+--ro as-number-hop
+--ro as-number      inet:as-number
+--ro hop-type?      te-hop-type
+--:(label)
+--ro label-hop
+--ro te-label
+--ro (technology)?
+--:(generic)
+--ro generic?
rt-
types:generalized-label
+--ro direction?
te-label-direction
+--ro domain-id?      uint32
+--ro is-abstract?    empty
+--ro name?           string
+--ro signaling-address*  inet:ip-address
+--ro underlay-topology {te-topology-hierarchy}?
+--ro network-ref?    -> /nw:networks/network/network-id
+--ro statistics
+--ro discontinuity-time?  yang:date-and-time
+--ro node
+--ro disables?        yang:counter32
+--ro enables?         yang:counter32
+--ro maintenance-sets? yang:counter32
+--ro maintenance-clears? yang:counter32
+--ro modifies?        yang:counter32
+--ro connectivity-matrix-entry
+--ro creates?          yang:counter32
+--ro deletes?          yang:counter32
+--ro disables?         yang:counter32
+--ro enables?          yang:counter32
+--ro modifies?         yang:counter32
+--rw tunnel-termination-point* [tunnel-tp-id]
+--rw tunnel-tp-id      binary
+--rw admin-status?
| te-types:te-admin-status
+--rw name?             string
+--rw switching-capability?  identityref
+--rw encoding?          identityref
+--rw inter-layer-lock-id*  uint32

```

```

+--rw protection-type?                               identityref
+--rw client-layer-adaptation
|   +--rw switching-capability*
|   |   [switching-capability encoding]
|   |   +--rw switching-capability    identityref
|   |   +--rw encoding                identityref
|   |   +--rw te-bandwidth
|   |   |   +--rw (technology)?
|   |   |   |   +--:(generic)
|   |   |   |   +--rw generic?    te-bandwidth
|   |   +--rw generic?    te-bandwidth
+--rw local-link-connectivities
|   +--rw number-of-entries?    uint16
|   +--rw label-restrictions
|   |   +--rw label-restriction* [index]
|   |   |   +--rw restriction?    enumeration
|   |   |   +--rw index          uint32
|   |   |   +--rw label-start
|   |   |   |   +--rw te-label
|   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   +--rw generic?
|   |   |   |   |   |   |   rt-types:generalized-label
|   |   |   |   |   +--rw direction?    te-label-direction
|   |   |   +--rw label-end
|   |   |   |   +--rw te-label
|   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   +--rw generic?
|   |   |   |   |   |   |   rt-types:generalized-label
|   |   |   |   |   +--rw direction?    te-label-direction
|   |   |   +--rw label-step
|   |   |   |   +--rw (technology)?
|   |   |   |   |   +--:(generic)
|   |   |   |   |   +--rw generic?    int32
|   |   |   +--rw range-bitmap?    yang:hex-string
|   +--rw is-allowed?    boolean
+--rw underlay {te-topology-hierarchy}?
|   +--rw enabled?    boolean
|   +--rw primary-path
|   |   +--rw network-ref?
|   |   |   -> /nw:networks/network/network-id

```

			<pre> +--rw path-element* [path-element-id] +--rw path-element-id          uint32 +--rw (type)? +--: (numbered-node-hop)     +--rw numbered-node-hop         +--rw node-id          te-node-id         +--rw hop-type?        te-hop-type +--: (numbered-link-hop)     +--rw numbered-link-hop         +--rw link-tp-id        te-tp-id         +--rw hop-type?        te-hop-type         +--rw direction?       te-link-direction +--: (unnumbered-link-hop)     +--rw unnumbered-link-hop         +--rw link-tp-id        te-tp-id         +--rw node-id          te-node-id         +--rw hop-type?        te-hop-type         +--rw direction?       te-link-direction +--: (as-number)     +--rw as-number-hop         +--rw as-number        inet:as-number         +--rw hop-type?        te-hop-type +--: (label)     +--rw label-hop         +--rw te-label             +--rw (technology)?                 +--: (generic)                     +--rw generic?                         rt-types:generalized- </pre>
label			<pre> +--rw direction? +--rw te-label-direction +--rw backup-path* [index] +--rw index          uint32 +--rw network-ref?     -&gt; /nw:networks/network/network-id +--rw path-element* [path-element-id] +--rw path-element-id          uint32 +--rw (type)? +--: (numbered-node-hop)     +--rw numbered-node-hop </pre>

				<pre>       +--rw node-id      te-node-id       +--rw hop-type?    te-hop-type +--:(numbered-link-hop)   +--rw numbered-link-hop     +--rw link-tp-id     te-tp-id     +--rw hop-type?      te-hop-type     +--rw direction?     te-link-direction +--:(unnumbered-link-hop)   +--rw unnumbered-link-hop     +--rw link-tp-id     te-tp-id     +--rw node-id        te-node-id     +--rw hop-type?      te-hop-type     +--rw direction?     te-link-direction +--:(as-number)   +--rw as-number-hop     +--rw as-number      inet:as-number     +--rw hop-type?      te-hop-type +--:(label)   +--rw label-hop     +--rw te-label       +--rw (technology)?         +--:(generic)           +--rw generic?             rt-types:generalized- </pre>
label				<pre>       +--rw direction?         te-label-direction +--rw protection-type?      identityref +--rw tunnel-termination-points   +--rw source?             binary   +--rw destination?        binary +--rw tunnels   +--rw sharing?            boolean   +--rw tunnel* [tunnel-name]     +--rw tunnel-name       string     +--rw sharing?          boolean +--rw path-constraints   +--rw te-bandwidth     +--rw (technology)?       +--:(generic)         +--rw generic?     te-bandwidth </pre>

```

+--rw link-protection?          identityref
+--rw setup-priority?           uint8
+--rw hold-priority?            uint8
+--rw signaling-type?           identityref
+--rw path-metric-bounds
|   +--rw path-metric-bound* [metric-type]
|   |   +--rw metric-type      identityref
|   |   +--rw upper-bound?    uint64
+--rw path-affinities-values
|   +--rw path-affinities-value* [usage]
|   |   +--rw usage            identityref
|   |   +--rw value?          admin-groups
+--rw path-affinity-names
|   +--rw path-affinity-name* [usage]
|   |   +--rw usage            identityref
|   |   +--rw affinity-name* [name]
|   |   |   +--rw name        string
+--rw path-srlgs-lists
|   +--rw path-srlgs-list* [usage]
|   |   +--rw usage            identityref
|   |   +--rw values*         srlg
+--rw path-srlgs-names
|   +--rw path-srlgs-name* [usage]
|   |   +--rw usage            identityref
|   |   +--rw names*          string
+--rw disjointness?             te-path-disjointness
+--rw optimizations
|   +--rw (algorithm)?
|   |   +--:(metric) {path-optimization-metric}?
|   |   |   +--rw optimization-metric* [metric-type]
|   |   |   |   +--rw metric-type
|   |   |   |   |   identityref
|   |   |   |   +--rw weight?
|   |   |   |   |   uint8
|   |   |   +--rw explicit-route-exclude-objects
|   |   |   |   +--rw route-object-exclude-object*
|   |   |   |   |   [index]
|   |   |   |   |   +--rw index
|   |   |   |   |   |   uint32
|   |   |   +--rw (type)?
|   |   |   |   +--:(numbered-node-hop)

```

					+--rw numbered-node-hop
					+--rw node-id te-node-id
					+--rw hop-type? te-hop-type
				+---:(numbered-link-hop)	
				+--rw numbered-link-hop	
				+--rw link-tp-id te-tp-id	
				+--rw hop-type?	
				te-hop-type	
				+--rw direction?	
				te-link-direction	
				+---:(unnumbered-link-hop)	
				+--rw unnumbered-link-hop	
				+--rw link-tp-id te-tp-id	
				+--rw node-id	
				te-node-id	
				+--rw hop-type?	
				te-hop-type	
				+--rw direction?	
				te-link-direction	
				+---:(as-number)	
				+--rw as-number-hop	
				+--rw as-number	
				inet:as-number	
				+--rw hop-type?	
				te-hop-type	
				+---:(label)	
				+--rw label-hop	
				+--rw te-label	
				+--rw (technology)?	
				+---:(generic)	
				+--rw generic?	
				rt-	
				+--rw direction?	
				te-label-direction	
				+---:(srlg)	
				+--rw srlg	
				+--rw srlg? uint32	
				+--rw explicit-route-include-objects	
				+--rw route-object-include-object*	
				[index]	



					+--rw index
					uint32
					+--rw (type)?
					+--:(numbered-node-hop)
					+--rw numbered-node-hop
					+--rw node-id te-node-id
					+--rw hop-type? te-hop-type
					+--:(numbered-link-hop)
					+--rw numbered-link-hop
					+--rw link-tp-id te-tp-id
					+--rw hop-type?
					te-hop-type
					+--rw direction?
					te-link-direction
					+--:(unnumbered-link-hop)
					+--rw unnumbered-link-hop
					+--rw link-tp-id te-tp-id
					+--rw node-id
					te-node-id
					+--rw hop-type?
					te-hop-type
					+--rw direction?
					te-link-direction
					+--:(as-number)
					+--rw as-number-hop
					+--rw as-number
					inet:as-number
					+--rw hop-type?
					te-hop-type
					+--:(label)
					+--rw label-hop
					+--rw te-label
					+--rw (technology)?
					+--:(generic)
					+--rw generic?
					rt-
					+--rw direction?
					te-label-direction
					+--rw tiebreakers
					+--rw tiebreaker* [tiebreaker-type]

types:generalized-label

```

|         +--rw tiebreaker-type      identityref
+--:(objective-function)
|         {path-optimization-objective-function}?
|         +--rw objective-function
|         +--rw objective-function-type?  identityref
+--ro path-properties
+--ro path-metric* [metric-type]
|   +--ro metric-type      identityref
|   +--ro accumulative-value?  uint64
+--ro path-affinities-values
|   +--ro path-affinities-value* [usage]
|   |   +--ro usage      identityref
|   |   +--ro value?    admin-groups
+--ro path-affinity-names
|   +--ro path-affinity-name* [usage]
|   |   +--ro usage      identityref
|   |   +--ro affinity-name* [name]
|   |   |   +--ro name      string
+--ro path-srlgs-lists
|   +--ro path-srlgs-list* [usage]
|   |   +--ro usage      identityref
|   |   +--ro values*    srlg
+--ro path-srlgs-names
|   +--ro path-srlgs-name* [usage]
|   |   +--ro usage      identityref
|   |   +--ro names*    string
+--ro path-route-objects
|   +--ro path-route-object* [index]
|   |   +--ro index      uint32
|   |   +--ro (type)?
|   |   |   +--:(numbered-node-hop)
|   |   |   |   +--ro numbered-node-hop
|   |   |   |   |   +--ro node-id      te-node-id
|   |   |   |   |   +--ro hop-type?    te-hop-type
|   |   |   +--:(numbered-link-hop)
|   |   |   |   +--ro numbered-link-hop
|   |   |   |   |   +--ro link-tp-id    te-tp-id
|   |   |   |   |   +--ro hop-type?    te-hop-type
|   |   |   |   |   +--ro direction?    te-link-direction
|   |   |   +--:(unnumbered-link-hop)
|   |   |   |   +--ro unnumbered-link-hop

```

```

|         +--ro link-tp-id          te-tp-id
|         +--ro node-id             te-node-id
|         +--ro hop-type?           te-hop-type
|         +--ro direction?          te-link-direction
|     +---:(as-number)
|         +--ro as-number-hop
|         +--ro as-number           inet:as-number
|         +--ro hop-type?           te-hop-type
|     +---:(label)
|         +--ro label-hop
|         +--ro te-label
|             +--ro (technology)?
|                 +---:(generic)
|                     +--ro generic?
|                         rt-types:generalized-
label
|
|         +--ro direction?
|             te-label-direction
+--rw local-link-connectivity* [link-tp-ref]
    +--rw link-tp-ref
        -> ../../../../nt:termination-point/tp-id
    +--rw label-restrictions
        +--rw label-restriction* [index]
            +--rw restriction?       enumeration
            +--rw index               uint32
            +--rw label-start
                +--rw te-label
                    +--rw (technology)?
                        +---:(generic)
                            +--rw generic?
                                rt-types:generalized-label
                    +--rw direction?   te-label-direction
            +--rw label-end
                +--rw te-label
                    +--rw (technology)?
                        +---:(generic)
                            +--rw generic?
                                rt-types:generalized-label
                    +--rw direction?   te-label-direction
            +--rw label-step
                +--rw (technology)?

```

```

+---:(generic)
|   +---rw generic?    int32
|   +---rw range-bitmap? yang:hex-string
+---rw is-allowed?      boolean
+---rw underlay {te-topology-hierarchy}?
|   +---rw enabled?      boolean
|   +---rw primary-path
|   |   +---rw network-ref?
|   |   |   -> /nw:networks/network/network-id
|   +---rw path-element* [path-element-id]
|   |   +---rw path-element-id      uint32
|   +---rw (type)?
|   |   +---:(numbered-node-hop)
|   |   |   +---rw numbered-node-hop
|   |   |   |   +---rw node-id      te-node-id
|   |   |   |   +---rw hop-type?    te-hop-type
|   |   +---:(numbered-link-hop)
|   |   |   +---rw numbered-link-hop
|   |   |   |   +---rw link-tp-id    te-tp-id
|   |   |   |   +---rw hop-type?    te-hop-type
|   |   |   |   +---rw direction?
|   |   |   |   |   te-link-direction
|   |   +---:(unnumbered-link-hop)
|   |   |   +---rw unnumbered-link-hop
|   |   |   |   +---rw link-tp-id    te-tp-id
|   |   |   |   +---rw node-id      te-node-id
|   |   |   |   +---rw hop-type?    te-hop-type
|   |   |   |   +---rw direction?
|   |   |   |   |   te-link-direction
|   |   +---:(as-number)
|   |   |   +---rw as-number-hop
|   |   |   |   +---rw as-number      inet:as-number
|   |   |   |   +---rw hop-type?      te-hop-type
|   |   +---:(label)
|   |   |   +---rw label-hop
|   |   |   |   +---rw te-label
|   |   |   |   |   +---rw (technology)?
|   |   |   |   |   |   +---:(generic)
|   |   |   |   |   |   |   +---rw generic?
|   |   |   |   |   |   |   |   rt-
types:generalized-label

```

```

|                                     +--rw direction?
|                                     te-label-direction
+--rw backup-path* [index]
|   +--rw index                      uint32
|   +--rw network-ref?
|   |   -> /nw:networks/network/network-id
+--rw path-element* [path-element-id]
|   +--rw path-element-id            uint32
|   +--rw (type)?
|   |   +--:(numbered-node-hop)
|   |   |   +--rw numbered-node-hop
|   |   |   |   +--rw node-id        te-node-id
|   |   |   |   +--rw hop-type?     te-hop-type
|   |   +--:(numbered-link-hop)
|   |   |   +--rw numbered-link-hop
|   |   |   |   +--rw link-tp-id     te-tp-id
|   |   |   |   +--rw hop-type?     te-hop-type
|   |   |   |   +--rw direction?
|   |   |   |   te-link-direction
|   |   +--:(unnumbered-link-hop)
|   |   |   +--rw unnumbered-link-hop
|   |   |   |   +--rw link-tp-id     te-tp-id
|   |   |   |   +--rw node-id        te-node-id
|   |   |   |   +--rw hop-type?     te-hop-type
|   |   |   |   +--rw direction?
|   |   |   |   te-link-direction
|   |   +--:(as-number)
|   |   |   +--rw as-number-hop
|   |   |   |   +--rw as-number      inet:as-number
|   |   |   |   +--rw hop-type?     te-hop-type
|   |   +--:(label)
|   |   |   +--rw label-hop
|   |   |   |   +--rw te-label
|   |   |   |   |   +--rw (technology)?
|   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   |   +--rw generic?
|   |   |   |   |   |   |   rt-
|   |   |   |   |   |   |   +--rw direction?
|   |   |   |   |   |   |   te-label-direction
+--rw protection-type?              identityref

```

types:generalized-label

```

+--rw tunnel-termination-points
|   +--rw source?      binary
|   +--rw destination? binary
+--rw tunnels
|   +--rw sharing?     boolean
|   +--rw tunnel* [tunnel-name]
|       +--rw tunnel-name    string
|       +--rw sharing?      boolean
+--rw path-constraints
|   +--rw te-bandwidth
|       +--rw (technology)?
|           +--:(generic)
|               +--rw generic?    te-bandwidth
+--rw link-protection?      identityref
+--rw setup-priority?       uint8
+--rw hold-priority?        uint8
+--rw signaling-type?       identityref
+--rw path-metric-bounds
|   +--rw path-metric-bound* [metric-type]
|       +--rw metric-type    identityref
|       +--rw upper-bound?   uint64
+--rw path-affinities-values
|   +--rw path-affinities-value* [usage]
|       +--rw usage          identityref
|       +--rw value?         admin-groups
+--rw path-affinity-names
|   +--rw path-affinity-name* [usage]
|       +--rw usage          identityref
|       +--rw affinity-name* [name]
|           +--rw name        string
+--rw path-srlgs-lists
|   +--rw path-srlgs-list* [usage]
|       +--rw usage          identityref
|       +--rw values*        srlg
+--rw path-srlgs-names
|   +--rw path-srlgs-name* [usage]
|       +--rw usage          identityref
|       +--rw names*         string
+--rw disjointness?
|       te-path-disjointness
+--rw optimizations

```

```

+--rw (algorithm)?
  +--:(metric) {path-optimization-metric}?
    +--rw optimization-metric* [metric-type]
      +--rw metric-type
        | identityref
      +--rw weight?
        | uint8
      +--rw explicit-route-exclude-objects
        +--rw route-object-exclude-object*
          [index]
        +--rw index
          | uint32
        +--rw (type)?
          +--:(numbered-node-hop)
            +--rw numbered-node-hop
              +--rw node-id
                | te-node-id
              +--rw hop-type?
                | te-hop-type
          +--:(numbered-link-hop)
            +--rw numbered-link-hop
              +--rw link-tp-id
                | te-tp-id
              +--rw hop-type?
                | te-hop-type
              +--rw direction?
                | te-link-direction
          +--:(unnumbered-link-hop)
            +--rw unnumbered-link-hop
              +--rw link-tp-id
                | te-tp-id
              +--rw node-id
                | te-node-id
              +--rw hop-type?
                | te-hop-type
              +--rw direction?
                | te-link-direction
          +--:(as-number)
            +--rw as-number-hop
              +--rw as-number
                | inet:as-number

```

						+--rw hop-type?
						te-hop-type
					+--:(label)	
					+--rw label-hop	
					+--rw te-label	
					+--rw (technology)?	
					+--:(generic)	
					+--rw generic?	
					rt-	
types:generalized-label						
						+--rw direction?
						te-label-
direction						
					+--:(srlg)	
					+--rw srlg	
					+--rw srlg? uint32	
				+--rw explicit-route-include-objects		
				+--rw route-object-include-object*		
				[index]		
				+--rw index		
				uint32		
				+--rw (type)?		
				+--:(numbered-node-hop)		
				+--rw numbered-node-hop		
				+--rw node-id		
				te-node-id		
				+--rw hop-type?		
				te-hop-type		
				+--:(numbered-link-hop)		
				+--rw numbered-link-hop		
				+--rw link-tp-id		
				te-tp-id		
				+--rw hop-type?		
				te-hop-type		
				+--rw direction?		
				te-link-direction		
				+--:(unnumbered-link-hop)		
				+--rw unnumbered-link-hop		
				+--rw link-tp-id		
				te-tp-id		
				+--rw node-id		



					te-node-id	
					+--rw hop-type?	
					te-hop-type	
					+--rw direction?	
					te-link-direction	
				+---:(as-number)		
				+--rw as-number-hop		
				+--rw as-number		
				inet:as-number		
				+--rw hop-type?		
				te-hop-type		
				+---:(label)		
				+--rw label-hop		
				+--rw te-label		
				+--rw (technology)?		
				+---:(generic)		
				+--rw generic?		
				rt-		
types:generalized-label						
					+--rw direction?	
					te-label-	
direction						
				+--rw tiebreakers		
				+--rw tiebreaker* [tiebreaker-type]		
				+--rw tiebreaker-type identityref		
				+---:(objective-function)		
				{path-optimization-objective-		
function}?						
				+--rw objective-function		
				+--rw objective-function-type?		
				identityref		
				+--ro path-properties		
				+--ro path-metric* [metric-type]		
				+--ro metric-type identityref		
				+--ro accumulative-value? uint64		
				+--ro path-affinities-values		
				+--ro path-affinities-value* [usage]		
				+--ro usage identityref		
				+--ro value? admin-groups		
				+--ro path-affinity-names		
				+--ro path-affinity-name* [usage]		

```

    +--ro usage          identityref
    +--ro affinity-name* [name]
      +--ro name          string
+--ro path-srlgs-lists
  +--ro path-srlgs-list* [usage]
    +--ro usage          identityref
    +--ro values*        srlg
+--ro path-srlgs-names
  +--ro path-srlgs-name* [usage]
    +--ro usage          identityref
    +--ro names*         string
+--ro path-route-objects
  +--ro path-route-object* [index]
    +--ro index          uint32
    +--ro (type)?
      +--:(numbered-node-hop)
        +--ro numbered-node-hop
          +--ro node-id    te-node-id
          +--ro hop-type?  te-hop-type
      +--:(numbered-link-hop)
        +--ro numbered-link-hop
          +--ro link-tp-id  te-tp-id
          +--ro hop-type?  te-hop-type
          +--ro direction?
            te-link-direction
      +--:(unnumbered-link-hop)
        +--ro unnumbered-link-hop
          +--ro link-tp-id  te-tp-id
          +--ro node-id    te-node-id
          +--ro hop-type?  te-hop-type
          +--ro direction?
            te-link-direction
      +--:(as-number)
        +--ro as-number-hop
          +--ro as-number  inet:as-number
          +--ro hop-type?  te-hop-type
      +--:(label)
        +--ro label-hop
          +--ro te-label
            +--ro (technology)?
              +--:(generic)

```

```

|                                     | +--ro generic?
|                                     | rt-
types:generalized-label
|                                     | +--ro direction?
|                                     | te-label-direction
| +--ro oper-status?
| | te-types:te-oper-status
| +--ro geolocation
| | +--ro altitude?      int64
| | +--ro latitude?     geographic-coordinate-degree
| | +--ro longitude?    geographic-coordinate-degree
| +--ro statistics
| | +--ro discontinuity-time?      yang:date-and-time
| | +--ro tunnel-termination-point
| | | +--ro disables?      yang:counter32
| | | +--ro enables?      yang:counter32
| | | +--ro maintenance-clears? yang:counter32
| | | +--ro maintenance-sets? yang:counter32
| | | +--ro modifies?      yang:counter32
| | | +--ro downs?        yang:counter32
| | | +--ro ups?          yang:counter32
| | | +--ro in-service-clears? yang:counter32
| | | +--ro in-service-sets? yang:counter32
| | +--ro local-link-connectivity
| | | +--ro creates?      yang:counter32
| | | +--ro deletes?     yang:counter32
| | | +--ro disables?    yang:counter32
| | | +--ro enables?     yang:counter32
| | | +--ro modifies?    yang:counter32
| +--rw supporting-tunnel-termination-point*
| | [node-ref tunnel-tp-ref]
| | +--rw node-ref      inet:uri
| | +--rw tunnel-tp-ref binary
augment /nw:networks/nw:network/nt:link:
+--rw te!
| +--rw (bundle-stack-level)?
| | +--:(bundle)
| | | +--rw bundled-links
| | | | +--rw bundled-link* [sequence]
| | | | | +--rw sequence      uint32
| | | | | +--rw src-tp-ref?   leafref

```

```

|         +---rw des-tp-ref?   leafref
+---:(component)
|         +---rw component-links
|         |         +---rw component-link* [sequence]
|         |         |         +---rw sequence           uint32
|         |         |         +---rw src-interface-ref?  string
|         |         |         +---rw des-interface-ref?  string
+---rw te-link-template*
|         -> ../../../../te/templates/link-template/name
|         {template}?
+---rw te-link-attributes
|         +---rw access-type?
|         |         te-types:te-link-access-type
+---rw external-domain
|         +---rw network-ref?
|         |         -> /nw:networks/network/network-id
+---rw remote-te-node-id?      te-types:te-node-id
|         +---rw remote-te-link-tp-id?  te-types:te-tp-id
+---rw is-abstract?           empty
+---rw name?                  string
+---rw underlay {te-topology-hierarchy}?
|         +---rw enabled?          boolean
+---rw primary-path
|         +---rw network-ref?
|         |         -> /nw:networks/network/network-id
+---rw path-element* [path-element-id]
|         +---rw path-element-id      uint32
|         +---rw (type)?
|         |         +---:(numbered-node-hop)
|         |         |         +---rw numbered-node-hop
|         |         |         |         +---rw node-id      te-node-id
|         |         |         |         +---rw hop-type?    te-hop-type
|         |         |         +---:(numbered-link-hop)
|         |         |         |         +---rw numbered-link-hop
|         |         |         |         |         +---rw link-tp-id    te-tp-id
|         |         |         |         |         +---rw hop-type?    te-hop-type
|         |         |         |         |         +---rw direction?  te-link-direction
|         |         |         +---:(unnumbered-link-hop)
|         |         |         |         +---rw unnumbered-link-hop
|         |         |         |         |         +---rw link-tp-id    te-tp-id
|         |         |         |         |         +---rw node-id      te-node-id

```

				<pre>       +--rw hop-type?      te-hop-type       +--rw direction?    te-link-direction +--:(as-number)       +--rw as-number-hop       +--rw as-number      inet:as-number       +--rw hop-type?      te-hop-type +--:(label)       +--rw label-hop       +--rw te-label           +--rw (technology)?               +--:(generic)                   +--rw generic?                       rt-types:generalized- </pre>
label				<pre>           +--rw direction?               te-label-direction +--rw backup-path* [index] +--rw index          uint32 +--rw network-ref?     -&gt; /nw:networks/network/network-id +--rw path-element* [path-element-id] +--rw path-element-id          uint32 +--rw (type)? +--:(numbered-node-hop)     +--rw numbered-node-hop         +--rw node-id      te-node-id         +--rw hop-type?    te-hop-type +--:(numbered-link-hop)     +--rw numbered-link-hop         +--rw link-tp-id    te-tp-id         +--rw hop-type?    te-hop-type         +--rw direction?   te-link-direction +--:(unnumbered-link-hop)     +--rw unnumbered-link-hop         +--rw link-tp-id    te-tp-id         +--rw node-id      te-node-id         +--rw hop-type?    te-hop-type         +--rw direction?   te-link-direction +--:(as-number)     +--rw as-number-hop         +--rw as-number      inet:as-number </pre>

```

|         +--rw hop-type?      te-hop-type
+---:(label)
|         +--rw label-hop
|         +--rw te-label
|         +--rw (technology)?
|         |         +---:(generic)
|         |         +--rw generic?
|         |         rt-types:generalized-
label
|         +--rw direction?
|         |         te-label-direction
+---rw protection-type?      identityref
+---rw tunnel-termination-points
|   +--rw source?            binary
|   +--rw destination?      binary
+---rw tunnels
|   +--rw sharing?          boolean
|   +--rw tunnel* [tunnel-name]
|   |   +--rw tunnel-name    string
|   |   +--rw sharing?      boolean
+---rw admin-status?
|   te-types:te-admin-status
+---rw link-index?          uint64
+---rw administrative-group?
|   te-types:admin-groups
+---rw interface-switching-capability*
|   [switching-capability encoding]
|   +--rw switching-capability identityref
|   +--rw encoding            identityref
|   +--rw max-lsp-bandwidth* [priority]
|   |   +--rw priority        uint8
|   |   +--rw te-bandwidth
|   |   |   +--rw (technology)?
|   |   |   +---:(generic)
|   |   |   +--rw generic?    te-bandwidth
+---rw label-restrictions
|   +--rw label-restriction* [index]
|   |   +--rw restriction?    enumeration
|   |   +--rw index          uint32
|   +--rw label-start
|   |   +--rw te-label

```

```

      +---rw (technology)?
      |   +---:(generic)
      |       +---rw generic?
      |           rt-types:generalized-label
      +---rw direction?          te-label-direction
+---rw label-end
|   +---rw te-label
|   +---rw (technology)?
|   |   +---:(generic)
|   |       +---rw generic?
|   |           rt-types:generalized-label
|   +---rw direction?          te-label-direction
+---rw label-step
|   +---rw (technology)?
|   |   +---:(generic)
|   |       +---rw generic?    int32
+---rw range-bitmap?    yang:hex-string
+---rw link-protection-type?    identityref
+---rw max-link-bandwidth
|   +---rw te-bandwidth
|   +---rw (technology)?
|   |   +---:(generic)
|   |       +---rw generic?    te-bandwidth
+---rw max-resv-link-bandwidth
|   +---rw te-bandwidth
|   +---rw (technology)?
|   |   +---:(generic)
|   |       +---rw generic?    te-bandwidth
+---rw unreserved-bandwidth* [priority]
|   +---rw priority          uint8
|   +---rw te-bandwidth
|   +---rw (technology)?
|   |   +---:(generic)
|   |       +---rw generic?    te-bandwidth
+---rw te-default-metric?          uint32
+---rw te-delay-metric?            uint32
+---rw te-igp-metric?             uint32
+---rw te-srlgs
|   +---rw value*    te-types:srlg
+---rw te-nsrlgs {nsrlg}?
|   +---rw id*    uint32

```

```

+--ro oper-status?                te-types:te-oper-status
+--ro is-transitional?            empty
+--ro information-source?         te-info-source
+--ro information-source-instance? string
+--ro information-source-state
|   +--ro credibility-preference? uint16
|   +--ro logical-network-element? string
|   +--ro network-instance?       string
|   +--ro topology
|       +--ro link-ref?           leafref
|       +--ro network-ref?       -> /nw:networks/network/network-id
+--ro information-source-entry*
|   [information-source information-source-instance]
|   +--ro information-source         te-info-source
|   +--ro information-source-instance string
|   +--ro information-source-state
|       +--ro credibility-preference? uint16
|       +--ro logical-network-element? string
|       +--ro network-instance?       string
|       +--ro topology
|           +--ro link-ref?           leafref
|           +--ro network-ref?       -> /nw:networks/network/network-id
+--ro link-index?                 uint64
+--ro administrative-group?
|   te-types:admin-groups
+--ro interface-switching-capability*
|   [switching-capability encoding]
|   +--ro switching-capability      identityref
|   +--ro encoding                  identityref
|   +--ro max-lsp-bandwidth* [priority]
|       +--ro priority              uint8
|       +--ro te-bandwidth
|           +--ro (technology)?
|               +--:(generic)
|                   +--ro generic?   te-bandwidth
+--ro label-restrictions
|   +--ro label-restriction* [index]
|       +--ro restriction?      enumeration
|       +--ro index              uint32
|       +--ro label-start

```



```

    +--ro te-label
      +--ro (technology)?
        +--:(generic)
          +--ro generic?
            rt-types:generalized-label
      +--ro direction?          te-label-direction
    +--ro label-end
      +--ro te-label
        +--ro (technology)?
          +--:(generic)
            +--ro generic?
              rt-types:generalized-label
        +--ro direction?          te-label-direction
    +--ro label-step
      +--ro (technology)?
        +--:(generic)
          +--ro generic?    int32
    +--ro range-bitmap?    yang:hex-string
+--ro link-protection-type?    identityref
+--ro max-link-bandwidth
  +--ro te-bandwidth
    +--ro (technology)?
      +--:(generic)
        +--ro generic?    te-bandwidth
+--ro max-resv-link-bandwidth
  +--ro te-bandwidth
    +--ro (technology)?
      +--:(generic)
        +--ro generic?    te-bandwidth
+--ro unreserved-bandwidth* [priority]
  +--ro priority          uint8
  +--ro te-bandwidth
    +--ro (technology)?
      +--:(generic)
        +--ro generic?    te-bandwidth
+--ro te-default-metric?          uint32
+--ro te-delay-metric?            uint32
+--ro te-igp-metric?              uint32
+--ro te-srlgs
  +--ro value*    te-types:srlg
+--ro te-nsrlgs {nsrlg}?

```

```

    |         +---ro id*      uint32
+---ro recovery
    |         +---ro restoration-status?   te-types:te-recovery-status
    |         +---ro protection-status?    te-types:te-recovery-status
+---ro underlay {te-topology-hierarchy}?
    |         +---ro dynamic?      boolean
    |         +---ro committed?    boolean
+---ro statistics
    |         +---ro discontinuity-time?    yang:date-and-time
    |         +---ro disables?             yang:counter32
    |         +---ro enables?             yang:counter32
    |         +---ro maintenance-clears?   yang:counter32
    |         +---ro maintenance-sets?    yang:counter32
    |         +---ro modifies?            yang:counter32
    |         +---ro downs?               yang:counter32
    |         +---ro ups?                 yang:counter32
    |         +---ro fault-clears?        yang:counter32
    |         +---ro fault-detects?       yang:counter32
    |         +---ro protection-switches?  yang:counter32
    |         +---ro protection-reverts?   yang:counter32
    |         +---ro restoration-failures? yang:counter32
    |         +---ro restoration-starts?   yang:counter32
    |         +---ro restoration-successes? yang:counter32
    |         +---ro restoration-reversion-failures? yang:counter32
    |         +---ro restoration-reversion-starts? yang:counter32
    |         +---ro restoration-reversion-successes? yang:counter32
augment /nw:networks/nw:network/nw:node/nt:termination-point:
+---rw te-tp-id?   te-types:te-tp-id
+---rw te!
    +---rw admin-status?
    |         te-types:te-admin-status
    +---rw name?                                string
    +---rw interface-switching-capability*
    |         [switching-capability encoding]
    |         +---rw switching-capability      identityref
    |         +---rw encoding                  identityref
    |         +---rw max-lsp-bandwidth* [priority]
    |         |         +---rw priority          uint8
    |         |         +---rw te-bandwidth
    |         |         |         +---rw (technology)?
    |         |         |         |         +---:(generic)

```

```
|          +--rw generic?    te-bandwidth
+--rw inter-domain-plug-id?          binary
+--rw inter-layer-lock-id*           uint32
+--ro oper-status?
|   te-types:te-oper-status
+--ro geolocation
    +--ro altitude?    int64
    +--ro latitude?    geographic-coordinate-degree
    +--ro longitude?   geographic-coordinate-degree
```

## Appendix B. Companion YANG Model for Non-NMDA Compliant Implementations

The YANG module `ietf-te-topology` defined in this document is designed to be used in conjunction with implementations that support the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. In order to allow implementations to use the model even in cases when NMDA is not supported, the following companion module `ietf-te-topology-state` is defined as a state model, which mirrors the module `ietf-te-topology` defined earlier in this document. However, all data nodes in the companion module are non-configurable, to represent the applied configuration or the derived operational states.

The companion module, `ietf-te-topology-state`, is redundant and SHOULD NOT be supported by implementations that support NMDA.

As the structure of the module `ietf-te-topology-state` mirrors that of the module `ietf-te-topology`. The YANG tree of the module `ietf-te-topology-state` is not depicted separately.

## B.1. TE Topology State YANG Module

This module references [RFC6001], [RFC8345], and [I-D.ietf-teas-yang-te-types].

```
<CODE BEGINS> file "ietf-te-topology-state@2019-02-07.yang"
module ietf-te-topology-state {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology-state";

  prefix "tet-s";

  import ietf-te-types {
    prefix "te-types";
    reference
      "I-D.ietf-teas-yang-te-types: Traffic Engineering Common YANG
      Types";
  }

  import ietf-te-topology {
    prefix "tet";
  }

  import ietf-network-state {
```

```
    prefix "nw-s";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
}

import ietf-network-topology-state {
    prefix "nt-s";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
}

organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
    Working Group";

contact
    "WG Web:    <http://tools.ietf.org/wg/teas/>
    WG List:    <mailto:teas@ietf.org>

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

    Editor:     Igor Bryskin
                <mailto:Igor.Bryskin@huawei.com>

    Editor:     Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

    Editor:     Tarek Saad
                <mailto:tsaad@juniper.net>

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

    Editor:     Oscar Gonzalez De Dios
                <mailto:oscar.gonzalezdedios@telefonica.com>";

description
    "TE topology state model.

    Copyright (c) 2019 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."

```
revision "2019-02-07" {
  description "Initial revision";
  reference "RFC XXXX: YANG Data Model for TE Topologies";
  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note
}

/*
 * Groupings
 */
grouping te-node-connectivity-matrix-attributes {
  description
    "Termination point references of a connectivity matrix entry.";
  container from {
    description
      "Reference to source link termination point.";
    leaf tp-ref {
      type leafref {
        path "../..../nt-s:termination-point/nt-s:tp-id";
      }
      description
        "Relative reference to a termination point.";
    }
    uses te-types:label-set-info;
  }
  container to {
    description
      "Reference to destination link termination point.";
    leaf tp-ref {
      type leafref {
        path "../..../nt-s:termination-point/nt-s:tp-id";
      }
    }
  }
}
```

```
    }
    description
      "Relative reference to a termination point.";
  }
  uses te-types:label-set-info;
}
uses tet:connectivity-matrix-entry-path-attributes;
} // te-node-connectivity-matrix-attributes

grouping te-node-tunnel-termination-point-llc-list {
  description
    "Local link connectivity list of a tunnel termination
     point on a TE node.";
  list local-link-connectivity {
    key "link-tp-ref";
    description
      "The termination capabilities between
       tunnel-termination-point and link termination-point.
       The capability information can be used to compute
       the tunnel path.
       The Interface Adjustment Capability Descriptors (IACD)
       (defined in RFC 6001) on each link-tp can be derived from
       this local-link-connectivity list.";
    reference
      "RFC 6001: Generalized MPLS (GMPLS) Protocol Extensions
       for Multi-Layer and Multi-Region Networks (MLN/MRN).";

    leaf link-tp-ref {
      type leafref {
        path "../.../.../.../nt-s:termination-point/nt-s:tp-id";
      }
      description
        "Link termination point.";
    }
    uses te-types:label-set-info;
    uses tet:connectivity-matrix-entry-path-attributes;
  } // local-link-connectivity
} // te-node-tunnel-termination-point-config

/*
 * Data nodes
```

```
*/
augment "/nw-s:networks/nw-s:network/nw-s:network-types" {
  description
    "Introduce new network type for TE topology.";
  container te-topology {
    presence "Indicates TE topology.";
    description
      "Its presence identifies the TE topology type.";
  }
}

augment "/nw-s:networks" {
  description
    "Augmentation parameters for TE topologies.";
  uses tet:te-topologies-augment;
}

augment "/nw-s:networks/nw-s:network" {
  when "nw-s:network-types/tet-s:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Configuration parameters for TE topology.";
  uses tet:te-topology-augment;
}

augment "/nw-s:networks/nw-s:network/nw-s:node" {
  when "../nw-s:network-types/tet-s:te-topology" {
    description
      "Augmentation parameters apply only for networks with
      TE topology type.";
  }
  description
    "Configuration parameters for TE at node level.";
  leaf te-node-id {
    type te-types:te-node-id;
    description
      "The identifier of a node in the TE topology.
      A node is specific to a topology to which it belongs.";
  }
}
```



```
    }
    container te {
      must "../te-node-id" {
        description
          "te-node-id is mandatory.";
      }
      must "count(..nw-s:supporting-node)<=1" {
        description
          "For a node in a TE topology, there cannot be more
           than 1 supporting node. If multiple nodes are abstracted,
           the underlay-topology is used.";
      }
      presence "TE support.";
      description
        "Indicates TE support.";
      uses tet:te-node-augment;
    } // te
  }

  augment "/nw-s:networks/nw-s:network/nt-s:link" {
    when "../nw-s:network-types/tet-s:te-topology" {
      description
        "Augmentation parameters apply only for networks with
         TE topology type.";
    }
    description
      "Configuration parameters for TE at link level.";
    container te {
      must "count(..nt-s:supporting-link)<=1" {
        description
          "For a link in a TE topology, there cannot be more
           than 1 supporting link. If one or more link paths are
           abstracted, the underlay is used.";
      }
      presence "TE support.";
      description
        "Indicates TE support.";
      uses tet:te-link-augment;
    } // te
  }
}
```

```
augment "/nw-s:networks/nw-s:network/nw-s:node/"
  + "nt-s:termination-point" {
    when "../..nw-s:network-types/tet-s:te-topology" {
      description
        "Augmentation parameters apply only for networks with
        TE topology type.";
    }
    description
      "Configuration parameters for TE at termination point level.";
    uses tet:te-termination-point-augment;
  }

augment
  "/nw-s:networks/nw-s:network/nt-s:link/te/bundle-stack-level/"
  + "bundle/bundled-links/bundled-link" {
    when "../..nw-s:network-types/tet-s:te-topology" {
      description
        "Augmentation parameters apply only for networks with
        TE topology type.";
    }
    description
      "Augment TE link bundled link.";
    leaf src-tp-ref {
      type leafref {
        path "../..nw-s:node[nw-s:node-id = "
          + "current()../..nt-s:source/"
          + "nt-s:source-node]/"
          + "nt-s:termination-point/nt-s:tp-id";
        require-instance true;
      }
      description
        "Reference to another TE termination point on the
        same source node.";
    }
    leaf des-tp-ref {
      type leafref {
        path "../..nw-s:node[nw-s:node-id = "
          + "current()../..nt-s:destination/"
          + "nt-s:dest-node]/"
          + "nt-s:termination-point/nt-s:tp-id";
        require-instance true;
      }
    }
  }
```

```
    }
    description
      "Reference to another TE termination point on the
       same destination node.";
  }
}

augment
  "/nw-s:networks/nw-s:network/nw-s:node/te/"
+ "information-source-entry/connectivity-matrices/"
+ "connectivity-matrix" {
  when "../..../nw-s:network-types/tet-s:te-topology" {
    description
      "Augmentation parameters apply only for networks with
       TE topology type.";
  }
  description
    "Augment TE node connectivity-matrix.";
  uses te-node-connectivity-matrix-attributes;
}

augment
  "/nw-s:networks/nw-s:network/nw-s:node/te/te-node-attributes/"
+ "connectivity-matrices/connectivity-matrix" {
  when "../..../nw-s:network-types/tet-s:te-topology" {
    description
      "Augmentation parameters apply only for networks with
       TE topology type.";
  }
  description
    "Augment TE node connectivity-matrix.";
  uses te-node-connectivity-matrix-attributes;
}

augment
  "/nw-s:networks/nw-s:network/nw-s:node/te/"
+ "tunnel-termination-point/local-link-connectivities" {
  when "../..../nw-s:network-types/tet-s:te-topology" {
    description
      "Augmentation parameters apply only for networks with
       TE topology type.";
```

```
    }  
    description  
        "Augment TE node tunnel termination point LLCs  
        (Local Link Connectivities).";  
    uses te-node-tunnel-termination-point-llc-list;  
    }  
}  
<CODE ENDS>
```

## Appendix C. Example: YANG Model for Technology Specific Augmentations

This section provides an example YANG module to define a technology specific TE topology model for the example-topology described in Section 6.

```
module example-topology {
  yang-version 1.1;

  namespace "http://example.com/example-topology";
  prefix "ex-topo";

  import ietf-network {
    prefix "nw";
  }

  import ietf-network-topology {
    prefix "nt";
  }

  import ietf-te-topology {
    prefix "tet";
  }

  organization
    "Example Organization";
  contact
    "Editor: Example Author";

  description
    "This module defines a topology data model for the example
    technology.";

  revision 2018-06-15 {
    description
      "Initial revision.";
    reference
      "Example reference.";
  }

  /*
   * Data nodes
```

```
*/
augment "/nw:networks/nw:network/nw:network-types/"
+ "tet:te-topology" {
  description
    "Augment network types to define example topology type.";
  container example-topology {
    presence
      "Introduce new network type for example topology.";
    description
      "Its presence identifies the example topology type.";
  }
}

augment "/nw:networks/nw:network/tet:te" {
  when "../nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  description "Augment network topology.";
  container attributes {
    description "Attributes for example technology.";
    leaf attribute-1 {
      type uint8;
      description "Attribute 1 for example technology.";
    }
  }
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes" {
  when "../..nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  description "Augment node attributes.";
  container attributes {
    description "Attributes for example technology.";
  }
}
```

```
    leaf attribute-2 {
      type uint8;
      description "Attribute 2 for example technology.";
    }
  }
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices" {
  when "../..../nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  description "Augment node connectivity matrices.";
  container attributes {
    description "Attributes for example technology.";
    leaf attribute-3 {
      type uint8;
      description "Attribute 3 for example technology.";
    }
  }
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix" {
  when "../..../nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  description "Augment node connectivity matrix.";
  container attributes {
    description "Attributes for example technology.";
    leaf attribute-3 {
      type uint8;
      description "Attribute 3 for example technology.";
    }
  }
}
```

```
    }
  }

  augment "/nw:networks/nw:network/nw:node/tet:te/"
    + "tet:tunnel-termination-point" {
    when "../.../nw:network-types/tet:te-topology/"
      + "ex-topo:example-topology" {
      description
        "Augmentation parameters apply only for networks with
        example topology type.";
    }
    description "Augment tunnel termination point.";
    container attributes {
      description "Attributes for example technology.";
      leaf attribute-4 {
        type uint8;
        description "Attribute 4 for example technology.";
      }
    }
  }
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point/"
  + "tet:te" {
  when "../.../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  description "Augment link termination point.";
  container attributes {
    description "Attributes for example technology.";
    leaf attribute-5 {
      type uint8;
      description "Attribute 5 for example technology.";
    }
  }
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
  + "tet:te-link-attributes" {
```



```
when "../../../nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
description "Augment link attributes.";
container attributes {
  description "Attributes for example technology.";
  leaf attribute-6 {
    type uint8;
    description "Attribute 6 for example technology.";
  }
}
}

/*
 * Augment TE bandwidth.
 */

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:interface-switching-capability/tet:max-lsp-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:max-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  case "example" {
```

```
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:max-resv-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:unreserved-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
```

```

}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
when "../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
when "../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
}
}

```

```
    }
    description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
when "../..../..../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
when "../..../..../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
}
```

```

    }
  }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:client-layer-adaptation/"
+ "tet:switching-capability/tet:te-bandwidth/tet:technology" {
  when "../../../../../../../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
  when "../../../../../../../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;

```

```

        description "Bandwidth 1 for example technology.";
    }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:path-constraints/tet:te-bandwidth/tet:technology" {
when "../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:interface-switching-capability/tet:max-lsp-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
when "../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example{

```

```
        description "Attributes for example technology.";
        leaf bandwidth-1 {
            type uint32;
            description "Bandwidth 1 for example technology.";
        }
    }
    description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:max-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
    when "../..../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf bandwidth-1 {
                type uint32;
                description "Bandwidth 1 for example technology.";
            }
        }
    }
    description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:max-resv-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
    when "../..../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
}
```

```

    }
    case "example" {
      container example {
        description "Attributes for example technology.";
        leaf bandwidth-1 {
          type uint32;
          description "Bandwidth 1 for example technology.";
        }
      }
    }
    description "Augment TE bandwidth.";
  }

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:interface-switching-capability/tet:max-lsp-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  when "../..../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:max-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  when "../..../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {

```



```
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:max-resv-link-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
  when "../..../nw:network-types/tet:te-topology/"
  + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf bandwidth-1 {
        type uint32;
        description "Bandwidth 1 for example technology.";
      }
    }
  }
  description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:unreserved-bandwidth/"
```

```
+ "tet:te-bandwidth/tet:technology" {
when "../..../..../..../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf bandwidth-1 {
      type uint32;
      description "Bandwidth 1 for example technology.";
    }
  }
}
description "Augment TE bandwidth.";
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point/"
+ "tet:te/"
+ "tet:interface-switching-capability/tet:max-lsp-bandwidth/"
+ "tet:te-bandwidth/tet:technology" {
when "../..../..../..../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf bandwidth-1 {
      type uint32;
      description "Bandwidth 1 for example technology.";
    }
  }
}
description "Augment TE bandwidth.";
}
```

```
/*
 * Augment TE label.
 */

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
  case "example" {
```

```

        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

augment "/nw:networks/tet:te/tet:templates/"
+ "tet:link-template/tet:te-link-attributes/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
}
description "Augment TE label.";
}

/* Under te-node-attributes/connectivity-matrices */

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
    when "../.../nw:network-types/tet:te-topology/"
        + "ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
    case "example" {
        container example {

```

```

        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../..../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
}

```

```

    }
    case "example" {
      container example {
        description "Attributes for example technology.";
        leaf label-1 {
          type uint32;
          description "Label 1 for example technology.";
        }
      }
    }
    description "Augment TE label.";
  }

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:path-properties/tet:path-route-objects/"
+ "tet:path-route-object/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../..../nw:network-types/"

```

```
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
         example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

/* Under te-node-attributes/.../connectivity-matrix */

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:from/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
         example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}
```

```

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:from/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
container example {
description "Attributes for example technology.";
leaf label-1 {
type uint32;
description "Label 1 for example technology.";
}
}
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:to/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
container example {
description "Attributes for example technology.";
leaf label-1 {
type uint32;
description "Label 1 for example technology.";
}
}
}

```



```

    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:to/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {

```

```

    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../.../../.../../.../../.../../.../../.../../.../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:te-node-attributes/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:path-properties/tet:path-route-objects/"
+ "tet:path-route-object/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../.../../.../../.../../.../../.../../.../../.../../nw:network-types/"

```

```
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

/* Under information-source-entry/connectivity-matrices */

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
  when "../../../nw:network-types/tet:te-topology/"
    + "ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
}
description "Augment TE label.";
}
```

```

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
}

```

```

    description "Augment TE label.";
  }

  augment "/nw:networks/nw:network/nw:node/tet:te/"
    + "tet:information-source-entry/tet:connectivity-matrices/"
    + "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
    + "tet:label/tet:label-hop/tet:te-label/tet:technology" {
    when "../..../..../..../..../..../..../..../..../nw:network-types/"
      + "tet:te-topology/ex-topo:example-topology" {
      description
        "Augmentation parameters apply only for networks with
        example topology type.";
    }
    case "example" {
      container example {
        description "Attributes for example technology.";
        leaf label-1 {
          type uint32;
          description "Label 1 for example technology.";
        }
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
  + "tet:information-source-entry/tet:connectivity-matrices/"
  + "tet:path-properties/tet:path-route-objects/"
  + "tet:path-route-object/tet:type/"
  + "tet:label/tet:label-hop/tet:te-label/tet:technology" {
  when "../..../..../..../..../..../..../..../..../nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
      example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;

```

```

        description "Label 1 for example technology.";
    }
}
description "Augment TE label.";
}

/* Under information-source-entry/.../connectivity-matrix */

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:from/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
container example {
description "Attributes for example technology.";
leaf label-1 {
type uint32;
description "Label 1 for example technology.";
}
}
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:from/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with

```

```
        example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:to/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
    when "../..//../..//../..//../..//../..//nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/tet:to/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
```

```

+ "tet:te-label/tet:technology" {
when "../..../..../..../..../..../..../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../..../..../..../..../..../..../..../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

```



```
augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:information-source-entry/tet:connectivity-matrices/"
+ "tet:connectivity-matrix/"
+ "tet:path-properties/tet:path-route-objects/"
+ "tet:path-route-object/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
    }
  }
}
```

```
        description "Label 1 for example technology.";
    }
}
description "Augment TE label.";
}

/* Under tunnel-termination-point/local-link-connectivities */

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
container example {
description "Attributes for example technology.";
leaf label-1 {
type uint32;
description "Label 1 for example technology.";
}
}
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../../../../../../../../../../../nw:network-types/tet:te-topology/"
+ "ex-topo:example-topology" {
description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
}
```

```

case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../.../../.../../.../../.../../.../../.../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../.../../.../../.../../.../../.../../.../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description

```

```
        "Augmentation parameters apply only for networks with
        example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:path-properties/tet:path-route-objects/"
+ "tet:path-route-object/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
    when "../..//../..//../..//../..//../..//../..//nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

/* Under tunnel-termination-point/.../local-link-connectivity */

augment "/nw:networks/nw:network/nw:node/tet:te/"
```

```
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
when "../.../../...../nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
             example topology type.";
    }
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
when "../.../../...../nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
             example topology type.";
    }
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
```

```

    }
    description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
    description
        "Augmentation parameters apply only for networks with
        example topology type.";
}
case "example" {
    container example {
        description "Attributes for example technology.";
    }
}
}

```

```

        leaf label-1 {
            type uint32;
            description "Label 1 for example technology.";
        }
    }
    description "Augment TE label.";
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
+ "tet:tunnel-termination-point/tet:local-link-connectivities/"
+ "tet:local-link-connectivity/"
+ "tet:path-properties/tet:path-route-objects/"
+ "tet:path-route-object/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
    when "../..../nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {
        description
            "Augmentation parameters apply only for networks with
            example topology type.";
    }
    case "example" {
        container example {
            description "Attributes for example technology.";
            leaf label-1 {
                type uint32;
                description "Label 1 for example technology.";
            }
        }
    }
    description "Augment TE label.";
}

/* Under te-link-attributes */

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
    when "../..../nw:network-types/"
    + "tet:te-topology/ex-topo:example-topology" {

```

```
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
  when "../..//../..//../..//../nw:network-types/"
  + "tet:te-topology/ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:underlay/tet:primary-path/tet:path-element/tet:type/"
```



```
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:te-link-attributes/"
+ "tet:underlay/tet:backup-path/tet:path-element/tet:type/"
+ "tet:label/tet:label-hop/tet:te-label/tet:technology" {
when "../../../../../../../../../../../../../../../nw:network-types/"
+ "tet:te-topology/ex-topo:example-topology" {
  description
    "Augmentation parameters apply only for networks with
    example topology type.";
}
case "example" {
  container example {
    description "Attributes for example technology.";
    leaf label-1 {
      type uint32;
      description "Label 1 for example technology.";
    }
  }
}
description "Augment TE label.";
}
```

```
/* Under te-link information-source-entry */

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-start/"
+ "tet:te-label/tet:technology" {
  when "../..//../..//../..//../nw:network-types/"
  + "tet:te-topology/ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
  description "Augment TE label.";
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
+ "tet:information-source-entry/"
+ "tet:label-restrictions/tet:label-restriction/tet:label-end/"
+ "tet:te-label/tet:technology" {
  when "../..//../..//../..//../nw:network-types/"
  + "tet:te-topology/ex-topo:example-topology" {
    description
      "Augmentation parameters apply only for networks with
       example topology type.";
  }
  case "example" {
    container example {
      description "Attributes for example technology.";
      leaf label-1 {
        type uint32;
        description "Label 1 for example technology.";
      }
    }
  }
}
```

```
    }  
  }  
  description "Augment TE label.";  
}  
}
```

#### Contributors

Sergio Belotti  
Nokia  
Email: sergio.belotti@nokia.com

Dieter Beller  
Nokia  
Email: Dieter.Beller@nokia.com

Carlo Perocchio  
Ericsson  
Email: carlo.perocchio@ericsson.com

Italo Busi  
Huawei Technologies  
Email: Italo.Busi@huawei.com

#### Authors' Addresses

Xufeng Liu  
Volta Networks  
Email: xufeng.liu.ietf@gmail.com

Igor Bryskin  
Huawei Technologies  
Email: Igor.Bryskin@huawei.com

Vishnu Pavan Beeram  
Juniper Networks  
Email: vbeeram@juniper.net

Tarek Saad  
Juniper Networks  
Email: tsaad@juniper.net

Himanshu Shah  
Ciena  
Email: hshah@ciena.com

Oscar Gonzalez De Dios  
Telefonica  
Email: oscar.gonzalezdedios@telefonica.com



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: October 28, 2017

X. Chen  
L. Andersson  
Huawei Technologies  
N. Leymann  
Deutsche Telekom  
I. Minei  
Google  
K. Raza  
Cisco Systems, Inc.  
April 26, 2017

LDP Specification  
draft-ijln-mpls-rfc5036bis-04.txt

Abstract

The architecture for Multiprotocol Label Switching (MPLS) is described in RFC 3031. A fundamental concept in MPLS is that two Label Switching Routers (LSRs) must agree on the meaning of the labels used to forward traffic between and through them. This common understanding is achieved by using a set of procedures, called a label distribution protocol, by which one LSR informs another of label bindings it has made. This document defines a set of such procedures called LDP (for Label Distribution Protocol) by which LSRs distribute labels to support MPLS forwarding along normally routed paths.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 28, 2017.

## 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 (<http://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.

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

## Table of Contents

1. Editors notes - this section will be removed before publication . . . . .	5
1.1. Scope of RFC5036bis work . . . . .	5
1.2. ToDo . . . . .	6
2. LDP Overview . . . . .	7
2.1. LDP Peers . . . . .	8
2.2. LDP Message Exchange . . . . .	8
2.3. LDP Message Structure . . . . .	9
2.4. LDP Error Handling . . . . .	9
2.5. LDP Extensibility and Future Compatibility . . . . .	9
2.6. Specification Language . . . . .	9
3. LDP Operation . . . . .	9
3.1. FECs . . . . .	9
3.2. Label Spaces, Identifiers, Sessions, and Transport . . .	11
3.2.1. Label Spaces . . . . .	11
3.2.2. LDP Identifiers . . . . .	11
3.2.3. LDP Sessions . . . . .	12
3.2.4. LDP Transport . . . . .	12
3.3. LDP Sessions between Non-Directly Connected LSRs . . . .	12

3.4.	LDP Discovery . . . . .	13
3.4.1.	Basic Discovery Mechanism . . . . .	13
3.4.2.	Extended Discovery Mechanism . . . . .	13
3.5.	Establishing and Maintaining LDP Sessions . . . . .	14
3.5.1.	LDP Session Establishment . . . . .	14
3.5.2.	Transport Connection Establishment . . . . .	14
3.5.3.	Session Initialization . . . . .	16
3.5.4.	Initialization State Machine . . . . .	18
3.5.5.	Maintaining Hello Adjacencies . . . . .	20
3.5.6.	Maintaining LDP Sessions . . . . .	21
3.6.	Label Distribution and Management . . . . .	21
3.6.1.	Label Distribution Control Mode . . . . .	22
3.6.1.1.	Independent Label Distribution Control . . . . .	22
3.6.1.2.	Ordered Label Distribution Control . . . . .	22
3.6.2.	Label Retention Mode . . . . .	23
3.6.2.1.	Conservative Label Retention Mode . . . . .	23
3.6.2.2.	Liberal Label Retention Mode . . . . .	23
3.6.3.	Label Advertisement Mode . . . . .	24
3.7.	LDP Identifiers and Next Hop Addresses . . . . .	24
3.8.	Loop Detection . . . . .	24
3.8.1.	Label Request Message . . . . .	25
3.8.2.	Label Mapping Message . . . . .	26
3.8.3.	Discussion . . . . .	28
3.9.	Authenticity and Integrity of LDP Messages . . . . .	29
3.9.1.	TCP MD5 Signature Option . . . . .	29
3.9.2.	LDP Use of TCP MD5 Signature Option . . . . .	31
4.	Protocol Specification . . . . .	31
4.1.	LDP PDUs . . . . .	31
4.2.	LDP Procedures . . . . .	32
4.3.	Type-Length-Value Encoding . . . . .	33
4.4.	TLV Encodings for Commonly Used Parameters . . . . .	35
4.4.1.	FEC TLV . . . . .	35
4.4.1.1.	FEC Procedures . . . . .	37
4.4.2.	Label TLVs . . . . .	37
4.4.2.1.	Generic Label TLV . . . . .	37
4.4.2.2.	ATM Label TLV . . . . .	38
4.4.2.3.	Frame Relay Label TLV . . . . .	39
4.4.3.	Address List TLV . . . . .	40
4.4.4.	Hop Count TLV . . . . .	41
4.4.4.1.	Hop Count Procedures . . . . .	42
4.4.5.	Path Vector TLV . . . . .	43
4.4.5.1.	Path Vector Procedures . . . . .	44
4.4.6.	Status TLV . . . . .	45
4.5.	LDP Messages . . . . .	47
4.5.1.	Notification Message . . . . .	49
4.5.1.1.	Notification Message Procedures . . . . .	50
4.5.1.2.	Events Signaled by Notification Messages . . . . .	51
4.5.2.	Hello Message . . . . .	53



4.5.2.1. Hello Message Procedures . . . . .	56
4.5.3. Initialization Message . . . . .	57
4.5.3.1. Initialization Message Procedures . . . . .	66
4.5.4. KeepAlive Message . . . . .	67
4.5.4.1. KeepAlive Message Procedures . . . . .	67
4.5.5. Address Message . . . . .	67
4.5.5.1. Address Message Procedures . . . . .	68
4.5.6. Address Withdraw Message . . . . .	69
4.5.6.1. Address Withdraw Message Procedures . . . . .	70
4.5.7. Label Mapping Message . . . . .	70
4.5.7.1. Label Mapping Message Procedures . . . . .	71
4.5.8. Label Request Message . . . . .	74
4.5.8.1. Label Request Message Procedures . . . . .	75
4.5.9. Label Abort Request Message . . . . .	77
4.5.9.1. Label Abort Request Message Procedures . . . . .	77
4.5.10. Label Withdraw Message . . . . .	79
4.5.10.1. Label Withdraw Message Procedures . . . . .	80
4.5.11. Label Release Message . . . . .	81
4.5.11.1. Label Release Message Procedures . . . . .	82
4.6. Messages and TLVs for Extensibility . . . . .	82
4.6.1. LDP Vendor-Private Extensions . . . . .	83
4.6.1.1. LDP Vendor-Private TLVs . . . . .	83
4.6.1.2. LDP Vendor-Private Messages . . . . .	84
4.6.2. LDP Experimental Extensions . . . . .	86
4.7. Message Summary . . . . .	86
4.8. TLV Summary . . . . .	87
4.9. Status Code Summary . . . . .	89
4.10. Well-Known Numbers . . . . .	90
4.10.1. UDP and TCP Ports . . . . .	90
4.10.2. Implicit NULL Label . . . . .	90
5. RFC 5036 IANA Considerations . . . . .	90
5.1. Message Type Name Space . . . . .	91
5.2. TLV Type Name Space . . . . .	92
5.3. FEC Type Name Space . . . . .	92
5.4. Status Code Name Space . . . . .	92
5.5. Experiment ID Name Space . . . . .	93
6. Security Considerations . . . . .	93
6.1. Spoofing . . . . .	93
6.2. Privacy . . . . .	94
6.3. Denial of Service . . . . .	95
7. Areas for Future Study . . . . .	96
8. Changes from RFC 5036 . . . . .	97
9. IANA Considerations . . . . .	97
10. References . . . . .	98
10.1. Normative References . . . . .	98
10.2. Informative References . . . . .	100
Appendix A. LDP Label Distribution Procedures . . . . .	101
A.1. Handling Label Distribution Events . . . . .	104

A.1.1.	Receive Label Request . . . . .	104
A.1.2.	Receive Label Mapping . . . . .	108
A.1.3.	Receive Label Abort Request . . . . .	114
A.1.4.	Receive Label Release . . . . .	115
A.1.5.	Receive Label Withdraw . . . . .	117
A.1.6.	Recognize New FEC . . . . .	119
A.1.7.	Detect Change in FEC Next Hop . . . . .	122
A.1.8.	Receive Notification / Label Request Aborted . . . . .	124
A.1.9.	Receive Notification / No Label Resources . . . . .	125
A.1.10.	Receive Notification / No Route . . . . .	126
A.1.11.	Receive Notification / Loop Detected . . . . .	127
A.1.12.	Receive Notification / Label Resources Available . . . . .	127
A.1.13.	Detect Local Label Resources Have Become Available . . . . .	128
A.1.14.	LSR Decides to No Longer Label Switch a FEC . . . . .	129
A.1.15.	Timeout of Deferred Label Request . . . . .	130
A.2.	Common Label Distribution Procedures . . . . .	130
A.2.1.	Send_Label . . . . .	130
A.2.2.	Send_Label_Request . . . . .	132
A.2.3.	Send_Label_Withdraw . . . . .	133
A.2.4.	Send_Notification . . . . .	133
A.2.5.	Send_Message . . . . .	134
A.2.6.	Check_Received_Attributes . . . . .	134
A.2.7.	Prepare_Label_Request_Attributes . . . . .	136
A.2.8.	Prepare_Label_Mapping_Attributes . . . . .	137
	Acknowledgments . . . . .	140
	Authors' Addresses . . . . .	141

## 1. Editors notes - this section will be removed before publication

This entire section will be removed before publication.

### 1.1. Scope of RFC5036bis work

The goal of this document is to take the LDP specification to Internet Standard.

Currently RFC 5036 - the LDP Specification - is a Draft Standard; this step on the Standards Track has been removed. It is therefore the plan to move the document to Internet Standard.

This document includes updates to the LDP Specification defined since the document was published, including:

#### 1. Updates all references and citations.

RFC 5036 obsoleted RFC 3036, and will in turn be obsoleted by the RFC5036-bis-to-be.

2. RFC 5036 is updated by RFC 6720, RFC 6790, RFC 7358, RFC 7552.
3. Incorporate all outstanding Errata.

These include the following approved Erratum with IDs: 3156, 2133, 3155, 3415, 3425.

[Ed Note: Done in rev -01]

4. Close loops with Stephen on the security section.
5. Have IANA review the IANA section.

#### 1.2. ToDo

1. Evaluation of which of the RFCs that updated RFC 5036 need to be incorporated into the rfc5036bis document. Specifically, these RFCs updated RFC 5036: RFC 6720, RFC 6790, RFC 7358, RFC 7552. RFCs that updated RFC 5036 and will be incorporated into this rfc5036bis, will be Obsoleted by rfc5036bis.
2. Review IANA Allocations. Review the IANA sections (there are currently two) and merge them into one.
3. Evaluate if there are things, based on e.g. non-deployment, that should be removed from the rfc5036bis-to-be.
4. Evaluate what to do about RFC 7349, it is not listed as an update of LDP, but it is an extension of the LDP security mechanisms (Hello crypto).
5. RFC 2385 (The TCP Authentication Option) has been obsoleted by RFC 5925, the changes are such that the text we refer to is no longer there. We probably need a minor re-write of section 2.9.1.
6. This document now carries a pre RFC 5378 copyright statement, since there clearly are material included in the document prior to RFC 5378 (10 Nov 2008), Verify that this is the right approach.
7. Revisit section 4.5.3 "Initialization Message" if we decide to remove ATM and FR.
8. Check if how to forward messages relating path-vector and hop-count from multiple downstreams needs to be specified/clarified.

## 2. LDP Overview

The MPLS architecture RFC 3031 [RFC3031] defines a label distribution protocol as a set of procedures by which one Label Switched Router (LSR) informs another of the meaning of labels used to forward traffic between and through them.

The MPLS architecture does not assume a single label distribution protocol. In fact, a number of different label distribution protocols are being standardized. Existing protocols have been extended so that label distribution can be piggybacked on them. New protocols have also been defined for the explicit purpose of distributing labels. The MPLS architecture discusses some of the considerations when choosing a label distribution protocol for use in particular MPLS applications such as Traffic Engineering RFC 2702 [RFC2702].

The Label Distribution Protocol (LDP) is a protocol defined for distributing labels. It was originally published as RFC 3036 in January 2001. It was produced by the MPLS Working Group of the IETF and was jointly authored by Loa Andersson, Paul Doolan, Nancy Feldman, Andre Fredette, and Bob Thomas.

LDP is a protocol defined for distributing labels. It is the set of procedures and messages by which Label Switched Routers (LSRs) establish Label Switched Paths (LSPs) through a network by mapping network-layer routing information directly to data-link layer switched paths. These LSPs may have an endpoint at a directly attached neighbor (comparable to IP hop-by-hop forwarding), or may have an endpoint at a network egress node, enabling switching via all intermediary nodes.

LDP associates a Forwarding Equivalence Class (FEC) RFC 3031 [RFC3031] with each LSP it creates. The FEC associated with an LSP specifies which packets are "mapped" to that LSP. LSPs are extended through a network as each LSR "splices" incoming labels for a FEC to the outgoing label assigned to the next hop for the given FEC.

More information about the applicability of LDP can be found in RFC 3037 [RFC3037].

This document assumes (but does not require) familiarity with the MPLS architecture RFC 3031 [RFC3031]. Note that RFC 3031 [RFC3031] includes a glossary of MPLS terminology, such as ingress, label switched path, etc.

## 2.1. LDP Peers

Two LSRs that use LDP to exchange label/FEC mapping information are known as "LDP Peers" with respect to that information, and we speak of there being an "LDP Session" between them. A single LDP session allows each peer to learn the other's label mappings; i.e., the protocol is bidirectional.

## 2.2. LDP Message Exchange

There are four categories of LDP messages:

1. Discovery messages, used to announce and maintain the presence of an LSR in a network.
2. Session messages, used to establish, maintain, and terminate sessions between LDP peers.
3. Advertisement messages, used to create, change, and delete label mappings for FECs.
4. Notification messages, used to provide advisory information and to signal error information.

Discovery messages provide a mechanism whereby LSRs indicate their presence in a network by sending a Hello message periodically. This is transmitted as a UDP packet to the LDP port at the 'all routers on this subnet' group multicast address. When an LSR chooses to establish a session with another LSR learned via the Hello message, it uses the LDP initialization procedure over TCP transport. Upon successful completion of the initialization procedure, the two LSRs are LDP peers, and may exchange advertisement messages.

When to request a label or advertise a label mapping to a peer is largely a local decision made by an LSR. In general, the LSR requests a label mapping from a neighboring LSR when it needs one, and advertises a label mapping to a neighboring LSR when it wishes the neighbor to use a label.

Correct operation of LDP requires reliable and in-order delivery of messages. To satisfy these requirements, LDP uses the TCP transport for Session, Advertisement, and Notification messages, i.e., for everything but the UDP-based discovery mechanism.

### 2.3. LDP Message Structure

All LDP messages have a common structure that uses a Type-Length-Value (TLV) encoding scheme; see Section "Type-Length-Value Encoding". The Value part of a TLV-encoded object, or TLV for short, may itself contain one or more TLVs.

### 2.4. LDP Error Handling

LDP errors and other events of interest are signaled to an LDP peer by Notification messages.

There are two kinds of LDP Notification messages:

1. Error Notifications, used to signal fatal errors. If an LSR receives an Error Notification from a peer for an LDP session, it terminates the LDP session by closing the TCP transport connection for the session and discarding all label mappings learned via the session.
2. Advisory Notifications, used to pass on LSR information about the LDP session or the status of some previous message received from the peer.

### 2.5. LDP Extensibility and Future Compatibility

Functionality may be added to LDP in the future. It is likely that future functionality will utilize new messages and object types (TLVs). It may be desirable to employ such new messages and TLVs within a network using older implementations that do not recognize them. While it is not possible to make every future enhancement backwards compatible, some prior planning can ease the introduction of new capabilities. This specification defines rules for handling unknown message types and unknown TLVs for this purpose.

### 2.6. Specification Language

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

## 3. LDP Operation

### 3.1. FECs

It is necessary to precisely specify which packets may be mapped to each LSP. This is done by providing a FEC specification for each

LSP. The FEC identifies the set of IP packets that may be mapped to that LSP.

Each FEC is specified as a set of one or more FEC elements. Each FEC element identifies a set of packets that may be mapped to the corresponding LSP. When an LSP is shared by multiple FEC elements, that LSP is terminated at (or before) the node where the FEC elements can no longer share the same path.

This specification defines a single type of FEC element, the "Address Prefix FEC element". This element is an address prefix of any length from 0 to a full address, inclusive.

Additional FEC elements may be defined, as needed, by other specifications.

In the remainder of this section, we give the rules to be used for mapping packets to LSPs that have been set up using an Address Prefix FEC element.

We say that a particular address "matches" a particular address prefix if and only if that address begins with that prefix. We also say that a particular packet matches a particular LSP if and only if that LSP has an Address Prefix FEC element that matches the packet's destination address. With respect to a particular packet and a particular LSP, we refer to any Address Prefix FEC element that matches the packet as the "matching prefix".

The procedure for mapping a particular packet to a particular LSP uses the following rules. Each rule is applied in turn until the packet can be mapped to an LSP.

- If a packet matches exactly one LSP, the packet is mapped to that LSP.
- If a packet matches multiple LSPs, it is mapped to the LSP whose matching prefix is the longest. If there is no one LSP whose matching prefix is longest, the packet is mapped to one from the set of LSPs whose matching prefix is longer than the others. The procedure for selecting one of those LSPs is beyond the scope of this document.
- If it is known that a packet must traverse a particular egress router, and there is an LSP that has an Address Prefix FEC element that is a /32 address of that router, then the packet is mapped to that LSP. The procedure for obtaining this knowledge is beyond the scope of this document.

The procedure for determining that a packet must traverse a particular egress router is beyond the scope of this document. (As an example, if one is running a link state routing algorithm, it may be possible to obtain this information from the link state data base. As another example, if one is running BGP, it may be possible to obtain this information from the BGP next hop attribute of the packet's route.)

### 3.2. Label Spaces, Identifiers, Sessions, and Transport

#### 3.2.1. Label Spaces

The notion of "label space" is useful for discussing the assignment and distribution of labels. There are two types of label spaces:

- Per interface label space. Interface-specific incoming labels are used for interfaces that use interface resources for labels. An example of such an interface is a label-controlled ATM interface that uses VCIs (Virtual Channel Identifiers) as labels, or a Frame Relay interface that uses DLCIs (Data Link Connection Identifiers) as labels.

Note that the use of a per interface label space only makes sense when the LDP peers are "directly connected" over an interface, and the label is only going to be used for traffic sent over that interface.

- Per platform label space. Platform-wide incoming labels are used for interfaces that can share the same labels.

#### 3.2.2. LDP Identifiers

An LDP Identifier is a six octet quantity used to identify an LSR label space. The first four octets identify the LSR and must be a globally unique value, such as a 32-bit router Id assigned to the LSR. The last two octets identify a specific label space within the LSR. The last two octets of LDP Identifiers for platform-wide label spaces are always both zero. This document uses the following print representation for LDP Identifiers:

<LSR Id> : <label space id>

e.g., lsr171:0, lsr19:2.

Note that an LSR that manages and advertises multiple label spaces uses a different LDP Identifier for each such label space.



A situation where an LSR would need to advertise more than one label space to a peer and hence use more than one LDP Identifier occurs when the LSR has two links to the peer and both are ATM (and use per interface labels). Another situation would be where the LSR had two links to the peer, one of which is ethernet (and uses per platform labels) and the other of which is ATM.

### 3.2.3. LDP Sessions

LDP sessions exist between LSRs to support label exchange between them.

When an LSR uses LDP to advertise more than one label space to another LSR, it uses a separate LDP session for each label space.

### 3.2.4. LDP Transport

LDP uses TCP as a reliable transport for sessions.

When multiple LDP sessions are required between two LSRs, there is one TCP session for each LDP session.

## 3.3. LDP Sessions between Non-Directly Connected LSRs

LDP sessions between LSRs that are not directly connected at the link level may be desirable in some situations.

For example, consider a "traffic engineering" application where LSRa sends traffic matching some criteria via an LSP to non-directly connected LSRb rather than forwarding the traffic along its normally routed path.

The path between LSRa and LSRb would include one or more intermediate LSRs (LSR1,...LSRn). An LDP session between LSRa and LSRb would enable LSRb to label switch traffic arriving on the LSP from LSRa by providing LSRb means to advertise labels for this purpose to LSRa.

In this situation, LSRa would apply two labels to traffic it forwards on the LSP to LSRb: a label learned from LSR1 to forward traffic along the LSP path from LSRa to LSRb; and a label learned from LSRb to enable LSRb to label switch traffic arriving on the LSP.

LSRa first adds the label learned via its LDP session with LSRb to the packet label stack (either by replacing the label on top of the packet label stack with it if the packet arrives labeled or by pushing it if the packet arrives unlabeled). Next, it pushes the label for the LSP learned from LSR1 onto the label stack.

### 3.4. LDP Discovery

LDP discovery is a mechanism that enables an LSR to discover potential LDP peers. Discovery makes it unnecessary to explicitly configure an LSR's label switching peers.

There are two variants of the discovery mechanism:

- A Basic Discovery mechanism used to discover LSR neighbors that are directly connected at the link level.
- An Extended Discovery mechanism used to locate LSRs that are not directly connected at the link level.

#### 3.4.1. Basic Discovery Mechanism

To engage in LDP Basic Discovery on an interface, an LSR periodically sends LDP Link Hellos out the interface. LDP Link Hellos are sent as UDP packets addressed to the well-known LDP discovery port for the "all routers on this subnet" group multicast address.

An LDP Link Hello sent by an LSR carries the LDP Identifier for the label space the LSR intends to use for the interface and possibly additional information.

Receipt of an LDP Link Hello on an interface identifies a "Hello adjacency" with a potential LDP peer reachable at the link level on the interface as well as the label space the peer intends to use for the interface.

#### 3.4.2. Extended Discovery Mechanism

LDP sessions between non-directly connected LSRs are supported by LDP Extended Discovery.

To engage in LDP Extended Discovery, an LSR periodically sends LDP Targeted Hellos to a specific address. LDP Targeted Hellos are sent as UDP packets addressed to the well-known LDP discovery port at the specific address.

An LDP Targeted Hello sent by an LSR carries the LDP Identifier for the label space the LSR intends to use and possibly additional optional information.

Extended Discovery differs from Basic Discovery in the following ways:

- A Targeted Hello is sent to a specific address rather than to the "all routers" group multicast address for the outgoing interface.
- Unlike Basic Discovery, which is symmetric, Extended Discovery is asymmetric.

One LSR initiates Extended Discovery with another targeted LSR, and the targeted LSR decides whether to respond to or ignore the Targeted Hello. A targeted LSR that chooses to respond does so by periodically sending Targeted Hellos to the initiating LSR.

Receipt of an LDP Targeted Hello identifies a "Hello adjacency" with a potential LDP peer reachable at the network level and the label space the peer intends to use.

### 3.5. Establishing and Maintaining LDP Sessions

#### 3.5.1. LDP Session Establishment

The exchange of LDP Discovery Hellos between two LSRs triggers LDP session establishment. Session establishment is a two step process:

- Transport connection establishment
- Session initialization

The following describes establishment of an LDP session between LSRs LSR1 and LSR2 from LSR1's point of view. It assumes the exchange of Hellos specifying label space LSR1:a for LSR1 and label space LSR2:b for LSR2.

#### 3.5.2. Transport Connection Establishment

The exchange of Hellos results in the creation of a Hello adjacency at LSR1 that serves to bind the link (L) and the label spaces LSR1:a and LSR2:b.

1. If LSR1 does not already have an LDP session for the exchange of label spaces LSR1:a and LSR2:b, it attempts to open a TCP connection for a new LDP session with LSR2.

LSR1 determines the transport addresses to be used at its end (A1) and LSR2's end (A2) of the LDP TCP connection. Address A1 is determined as follows:

- a. If LSR1 uses the Transport Address optional object (TLV) in Hellos it sends to LSR2 to advertise an address, A1 is the address LSR1 advertises via the optional object;

- b. If LSR1 does not use the Transport Address optional object, A1 is the source address used in Hellos it sends to LSR2.

Similarly, address A2 is determined as follows:

- a. If LSR2 uses the Transport Address optional object, A2 is the address LSR2 advertises via the optional object;
  - b. If LSR2 does not use the Transport Address optional object, A2 is the source address in Hellos received from LSR2.
2. LSR1 determines whether it will play the active or passive role in session establishment by comparing addresses A1 and A2 as unsigned integers. If  $A1 > A2$ , LSR1 plays the active role; otherwise, it is passive.

The procedure for comparing A1 and A2 as unsigned integers is:

- If A1 and A2 are not in the same address family, they are incomparable, and no session can be established.
  - Let U1 be the abstract unsigned integer obtained by treating A1 as a sequence of bytes, where the byte that appears earliest in the message is the most significant byte of the integer and the byte that appears latest in the message is the least significant byte of the integer.
  - Let U2 be the abstract unsigned integer obtained from A2 in a similar manner.
  - Compare U1 with U2. If  $U1 > U2$ , then  $A1 > A2$ ; if  $U1 < U2$ , then  $A1 < A2$ .
3. If LSR1 is active, it attempts to establish the LDP TCP connection by connecting to the well-known LDP port at address A2. If LSR1 is passive, it waits for LSR2 to establish the LDP TCP connection to its well-known LDP port.

Note that when an LSR sends a Hello, it selects the transport address for its end of the session connection and uses the Hello to advertise the address, either explicitly by including it in an optional Transport Address TLV or implicitly by omitting the TLV and using it as the Hello source address.

An LSR MUST advertise the same transport address in all Hellos that advertise the same label space. This requirement ensures that two

LSRs linked by multiple Hello adjacencies using the same label spaces play the same connection establishment role for each adjacency.

### 3.5.3. Session Initialization

After LSR1 and LSR2 establish a transport connection, they negotiate session parameters by exchanging LDP Initialization messages. The parameters negotiated include LDP protocol version, label distribution method, timer values, VPI/VCI (Virtual Path Identifier / Virtual Channel Identifier) ranges for label controlled ATM, DLCI (Data Link Connection Identifier) ranges for label controlled Frame Relay, etc.

Successful negotiation completes establishment of an LDP session between LSR1 and LSR2 for the advertisement of label spaces LSR1:a and LSR2:b.

The following describes the session initialization from LSR1's point of view.

After the connection is established, if LSR1 is playing the active role, it initiates negotiation of session parameters by sending an Initialization message to LSR2. If LSR1 is passive, it waits for LSR2 to initiate the parameter negotiation.

In general when there are multiple links between LSR1 and LSR2 and multiple label spaces to be advertised by each, the passive LSR cannot know which label space to advertise over a newly established TCP connection until it receives the LDP Initialization message on the connection. The Initialization message carries both the LDP Identifier for the sender's (active LSR's) label space and the LDP Identifier for the receiver's (passive LSR's) label space.

By waiting for the Initialization message from its peer, the passive LSR can match the label space to be advertised by the peer (as determined from the LDP Identifier in the PDU header for the Initialization message) with a Hello adjacency previously created when Hellos were exchanged.

1. When LSR1 plays the passive role:
  - a. If LSR1 receives an Initialization message, it attempts to match the LDP Identifier carried by the message PDU with a Hello adjacency.
  - b. If there is a matching Hello adjacency, the adjacency specifies the local label space for the session.

Next LSR1 checks whether the session parameters proposed in the message are acceptable. If they are, LSR1 replies with an Initialization message of its own to propose the parameters it wishes to use and a KeepAlive message to signal acceptance of LSR2's parameters. If the parameters are not acceptable, LSR1 responds by sending a Session Rejected/Parameters Error Notification message and closing the TCP connection.

- c. If LSR1 cannot find a matching Hello adjacency, it sends a Session Rejected/No Hello Error Notification message and closes the TCP connection.
  - d. If LSR1 receives a KeepAlive in response to its Initialization message, the session is operational from LSR1's point of view.
  - e. If LSR1 receives an Error Notification message, LSR2 has rejected its proposed session and LSR1 closes the TCP connection.
2. When LSR1 plays the active role:
- a. If LSR1 receives an Error Notification message, LSR2 has rejected its proposed session and LSR1 closes the TCP connection.
  - b. If LSR1 receives an Initialization message, it checks whether the session parameters are acceptable. If so, it replies with a KeepAlive message. If the session parameters are unacceptable, LSR1 sends a Session Rejected/Parameters Error Notification message and closes the connection.
  - c. If LSR1 receives a KeepAlive message, LSR2 has accepted its proposed session parameters.
  - d. When LSR1 has received both an acceptable Initialization message and a KeepAlive message, the session is operational from LSR1's point of view.

Until the LDP session is established, no other messages except those listed in the procedures above may be exchanged, and the rules for processing the U-bit in LDP messages are overridden. If a message other than those listed in the procedures above is received, a Shutdown msg MUST be transmitted and the transport connection MUST be closed.

It is possible for a pair of incompatibly configured LSRs that disagree on session parameters to engage in an endless sequence of messages as each NAKs the other's Initialization messages with Error Notification messages.

An LSR MUST throttle its session setup retry attempts with an exponential backoff in situations where Initialization messages are being NAK'd. It is also recommended that an LSR detecting such a situation take action to notify an operator.

The session establishment setup attempt following a NAK'd Initialization message MUST be delayed no less than 15 seconds, and subsequent delays MUST grow to a maximum delay of no less than 2 minutes. The specific session establishment action that must be delayed is the attempt to open the session transport connection by the LSR playing the active role.

The throttled sequence of Initialization NAKs is unlikely to cease until operator intervention reconfigures one of the LSRs. After such a configuration action, there is no further need to throttle subsequent session establishment attempts (until their Initialization messages are NAK'd).

Due to the asymmetric nature of session establishment, reconfiguration of the passive LSR will go unnoticed by the active LSR without some further action. Section "Hello Message" describes an optional mechanism an LSR can use to signal potential LDP peers that it has been reconfigured.

#### 3.5.4. Initialization State Machine

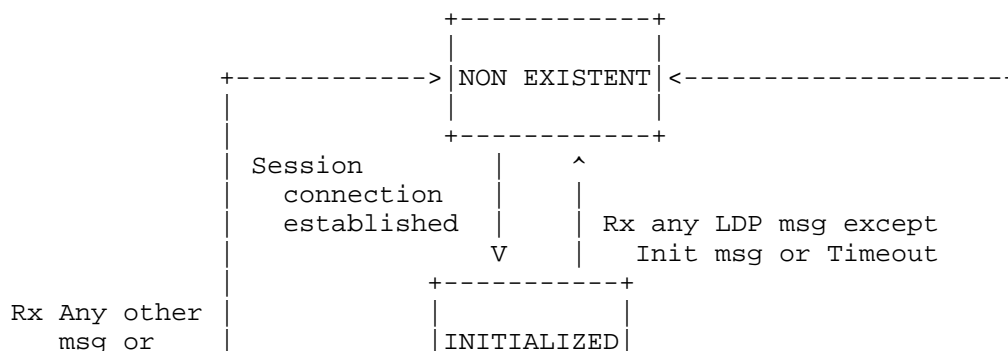
It is convenient to describe LDP session negotiation behavior in terms of a state machine. We define the LDP state machine to have five possible states and present the behavior as a state transition table and as a state transition diagram. Note that a Shutdown message is implemented as a Notification message with a Status TLV indicating a fatal error.

Session Initialization State Transition Table

STATE	EVENT	NEW STATE
NON EXISTENT	Session TCP connection established	INITIALIZED
INITIALIZED	Transmit Initialization msg (Active Role)	OPENSENT

	Receive acceptable Initialization msg (Passive Role) Action: Transmit Initialization msg and KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) and close transport connection	NON EXISTENT
OPENREC	Receive KeepAlive msg	OPERATIONAL
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) and close transport connection	NON EXISTENT
OPENSENT	Receive acceptable Initialization msg Action: Transmit KeepAlive msg	OPENREC
	Receive Any other LDP msg Action: Transmit Error Notification msg (NAK) and close transport connection	NON EXISTENT
OPERATIONAL	Receive Shutdown msg Action: Transmit Shutdown msg and close transport connection	NON EXISTENT
	Receive other LDP msgs Timeout Action: Transmit Shutdown msg and close transport connection	OPERATIONAL NON EXISTENT

Session Initialization State Transition Diagram





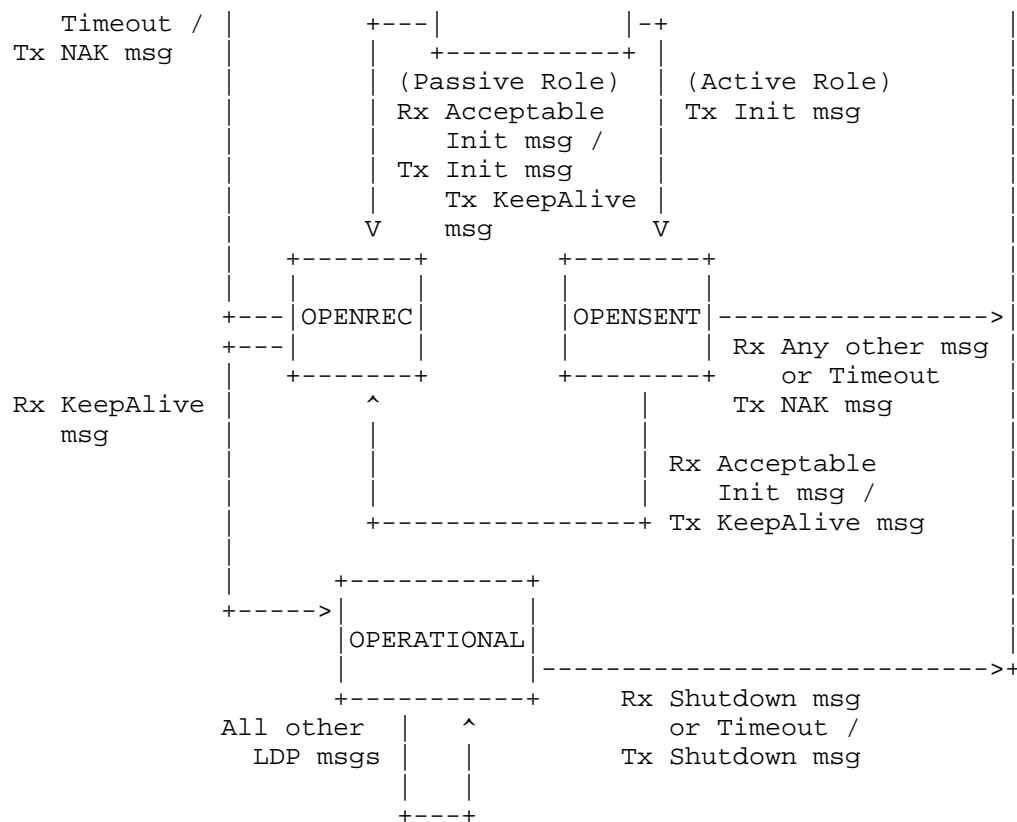


Figure 1: LDP State Machine

### 3.5.5. Maintaining Hello Adjacencies

An LDP session with a peer has one or more Hello adjacencies.

An LDP session has multiple Hello adjacencies when a pair of LSRs is connected by multiple links that share the same label space; for example, multiple PPP links between a pair of routers. In this situation, the Hellos an LSR sends on each such link carry the same LDP Identifier.

LDP includes mechanisms to monitor the necessity of an LDP session and its Hello adjacencies.

LDP uses the regular receipt of LDP Discovery Hellos to indicate a peer's intent to use the label space identified by the Hello. An LSR maintains a hold timer with each Hello adjacency that it restarts

when it receives a Hello that matches the adjacency. If the timer expires without receipt of a matching Hello from the peer, LDP concludes that the peer no longer wishes to label switch using that label space for that link (or target, in the case of Targeted Hellos) or that the peer has failed. The LSR then deletes the Hello adjacency. When the last Hello adjacency for an LDP session is deleted, the LSR terminates the LDP session by sending a Notification message and closing the transport connection.

### 3.5.6. Maintaining LDP Sessions

LDP includes mechanisms to monitor the integrity of the LDP session.

LDP uses the regular receipt of LDP PDUs on the session transport connection to monitor the integrity of the session. An LSR maintains a KeepAlive Timer for each peer session that it resets whenever it receives an LDP PDU from the session peer. If the KeepAlive Timer expires without receipt of an LDP PDU from the peer, the LSR concludes that the transport connection is bad or that the peer has failed, and it terminates the LDP session by closing the transport connection.

After an LDP session has been established, an LSR must arrange that its peer receive an LDP PDU from it at least every KeepAlive time period to ensure the peer restarts the session KeepAlive Timer. The LSR may send any protocol message to meet this requirement. In circumstances where an LSR has no other information to communicate to its peer, it sends a KeepAlive message.

An LSR may choose to terminate an LDP session with a peer at any time. Should it choose to do so, it informs the peer with a Shutdown message.

### 3.6. Label Distribution and Management

The MPLS architecture RFC 3031 [RFC3031] allows an LSR to distribute a FEC label binding in response to an explicit request from another LSR. This is known as Downstream On Demand label distribution. It also allows an LSR to distribute label bindings to LSRs that have not explicitly requested them. RFC 3031 [RFC3031] calls this method of label distribution Unsolicited Downstream; this document uses the term Downstream Unsolicited.

Both of these label distribution techniques may be used in the same network at the same time. However, for any given LDP session, each LSR must be aware of the label distribution method used by its peer in order to avoid situations where one peer using Downstream

Unsolicited label distribution assumes its peer is also. See Section "Downstream on Demand Label Advertisement".

### 3.6.1. Label Distribution Control Mode

The behavior of the initial setup of LSPs is determined by whether the LSR is operating with independent or Ordered LSP Control. An LSR may support both types of control as a configurable option.

#### 3.6.1.1. Independent Label Distribution Control

When using independent LSP control, each LSR may advertise label mappings to its neighbors at any time it desires. For example, when operating in independent Downstream on Demand mode, an LSR may answer requests for label mappings immediately, without waiting for a label mapping from the next hop. When operating in independent Downstream Unsolicited mode, an LSR may advertise a label mapping for a FEC to its neighbors whenever it is prepared to label-switch that FEC.

A consequence of using independent mode is that an upstream label can be advertised before a downstream label is received.

#### 3.6.1.2. Ordered Label Distribution Control

When using LSP Ordered Control, an LSR may initiate the transmission of a label mapping only for a FEC for which it has a label mapping for the FEC next hop, or for which the LSR is the egress. For each FEC for which the LSR is not the egress and no mapping exists, the LSR MUST wait until a label from a downstream LSR is received before mapping the FEC and passing corresponding labels to upstream LSRs. An LSR may be an egress for some FECs and a non-egress for others.

An LSR may act as an egress LSR, with respect to a particular FEC, under any of the following conditions:

1. The FEC refers to the LSR itself (including one of its directly attached interfaces).
2. The next hop router for the FEC is outside of the Label Switching Network.
3. FEC elements are reachable by crossing a routing domain boundary, such as another area for OSPF summary networks, or another autonomous system for OSPF AS externals and BGP routes RFC 2328 [RFC2328] and RFC 4271 [RFC4271].

Note that whether an LSR is an egress for a given FEC may change over time, depending on the state of the network and LSR configuration settings.

### 3.6.2. Label Retention Mode

The MPLS architecture RFC 3031 [RFC3031] introduces the notion of label retention mode which specifies whether an LSR maintains a label binding for a FEC learned from a neighbor that is not its next hop for the FEC.

#### 3.6.2.1. Conservative Label Retention Mode

In Downstream Unsolicited advertisement mode, label mapping advertisements for all routes may be received from all peer LSRs. When using Conservative Label retention, advertised label mappings are retained only if they will be used to forward packets (i.e., if they are received from a valid next hop according to routing). If operating in Downstream on Demand mode, an LSR will request label mappings only from the next hop LSR according to routing. Since Downstream on Demand mode is primarily used when label conservation is desired (e.g., an ATM switch with limited cross connect space), it is typically used with the Conservative Label retention mode.

The main advantage of the conservative mode is that only the labels that are required for the forwarding of data are allocated and maintained. This is particularly important in LSRs where the label space is inherently limited, such as in an ATM switch. A disadvantage of the conservative mode is that if routing changes the next hop for a given destination, a new label must be obtained from the new next hop before labeled packets can be forwarded.

#### 3.6.2.2. Liberal Label Retention Mode

In Downstream Unsolicited advertisement mode, label mapping advertisements for all routes may be received from all LDP peers. When using Liberal Label retention, every label mappings received from a peer LSR is retained regardless of whether the LSR is the next hop for the advertised mapping. When operating in Downstream on Demand mode with Liberal Label retention, an LSR might choose to request label mappings for all known prefixes from all peer LSRs. Note, however, that Downstream on Demand mode is typically used by devices such as ATM switch-based LSRs for which the conservative approach is recommended.

The main advantage of the Liberal Label retention mode is that reaction to routing changes can be quick because labels already

exist. The main disadvantage of the liberal mode is that unneeded label mappings are distributed and maintained.

### 3.6.3. Label Advertisement Mode

Each interface on an LSR is configured to operate in either Downstream Unsolicited or Downstream on Demand advertisement mode. LSRs exchange advertisement modes during initialization. The major difference between Downstream Unsolicited and Downstream on Demand modes is in which LSR takes responsibility for initiating mapping requests and mapping advertisements.

### 3.7. LDP Identifiers and Next Hop Addresses

An LSR maintains learned labels in a Label Information Base (LIB). When operating in Downstream Unsolicited mode, the LIB entry for an address prefix associates a collection of (LDP Identifier, label) pairs with the prefix, one such pair for each peer advertising a label for the prefix.

When the next hop for a prefix changes, the LSR must retrieve the label advertised by the new next hop from the LIB for use in forwarding. To retrieve the label, the LSR must be able to map the next hop address for the prefix to an LDP Identifier.

Similarly, when the LSR learns a label for a prefix from an LDP peer, it must be able to determine whether that peer is currently a next hop for the prefix to determine whether it needs to start using the newly learned label when forwarding packets that match the prefix. To make that decision, the LSR must be able to map an LDP Identifier to the peer's addresses to check whether any are a next hop for the prefix.

To enable LSRs to map between a peer LDP Identifier and the peer's addresses, LSRs advertise their addresses using LDP Address and Withdraw Address messages.

An LSR sends an Address message to advertise its addresses to a peer. An LSR sends a Withdraw Address message to withdraw previously advertised addresses from a peer.

### 3.8. Loop Detection

Loop Detection is a configurable option that provides a mechanism for finding looping LSPs and for preventing Label Request messages from looping in the presence of non-merge capable LSRs.

The mechanism makes use of Path Vector and Hop Count TLVs carried by Label Request and Label Mapping messages. It builds on the following basic properties of these TLVs:

- A Path Vector TLV contains a list of the LSRs that its containing message has traversed. An LSR is identified in a Path Vector list by its unique LSR Identifier (Id), which is the first four octets of its LDP Identifier. When an LSR propagates a message containing a Path Vector TLV, it adds its LSR Id to the Path Vector list. An LSR that receives a message with a Path Vector that contains its LSR Id detects that the message has traversed a loop. LDP supports the notion of a maximum allowable Path Vector length; an LSR that detects a Path Vector has reached the maximum length behaves as if the containing message has traversed a loop.
- A Hop Count TLV contains a count of the LSRS that the containing message has traversed. When an LSR propagates a message containing a Hop Count TLV, it increments the count. An LSR that detects a Hop Count has reached a configured maximum value behaves as if the containing message has traversed a loop. By convention, a count of 0 is interpreted to mean the hop count is unknown. Incrementing an unknown hop count value results in an unknown hop count value (0).

The following paragraphs describe LDP Loop Detection procedures. For these paragraphs, and only these paragraphs, "MUST" is redefined to mean "MUST if configured for Loop Detection". The paragraphs specify messages that MUST carry Path Vector and Hop Count TLVs. Note that the Hop Count TLV and its procedures are used without the Path Vector TLV in situations when Loop Detection is not configured (see RFC 3035 [RFC3035] and RFC 3034 [RFC3034]).

#### 3.8.1. Label Request Message

The use of the Path Vector TLV and Hop Count TLV prevent Label Request messages from looping in environments that include non-merge capable LSRs.

The rules that govern use of the Hop Count TLV in Label Request messages by LSR R when Loop Detection is enabled are the following:

- The Label Request message MUST include a Hop Count TLV.
- If R is sending the Label Request because it is a FEC ingress, it MUST include a Hop Count TLV with hop count value 1.
- If R is sending the Label Request as a result of having received a Label Request from an upstream LSR, and if the received Label

Request contains a Hop Count TLV, R MUST increment the received hop count value by 1 and MUST pass the resulting value in a Hop Count TLV to its next hop along with the Label Request message.

The rules that govern use of the Path Vector TLV in Label Request messages by LSR R when Loop Detection is enabled are the following:

- If R is sending the Label Request because it is a FEC ingress, then if R is non-merge capable, it MUST include a Path Vector TLV of length 1 containing its own LSR Id.
- If R is sending the Label Request as a result of having received a Label Request from an upstream LSR, then if the received Label Request contains a Path Vector TLV or if R is non-merge capable:

R MUST add its own LSR Id to the Path Vector, and MUST pass the resulting Path Vector to its next hop along with the Label Request message. If the Label Request contains no Path Vector TLV, R MUST include a Path Vector TLV of length 1 containing its own LSR Id.

Note that if R receives a Label Request message for a particular FEC, and R has previously sent a Label Request message for that FEC to its next hop and has not yet received a reply, and if R intends to merge the newly received Label Request with the existing outstanding Label Request, then R does not propagate the Label Request to the next hop.

If R receives a Label Request message from its next hop with a Hop Count TLV that exceeds the configured maximum value, or with a Path Vector TLV containing its own LSR Id or which exceeds the maximum allowable length, then R detects that the Label Request message has traveled in a loop.

When R detects a loop, it MUST send a Loop Detected Notification message to the source of the Label Request message and drop the Label Request message.

### 3.8.2. Label Mapping Message

The use of the Path Vector TLV and Hop Count TLV in the Label Mapping message provide a mechanism to find and terminate looping LSPs. When an LSR receives a Label Mapping message from a next hop, the message is propagated upstream as specified below until an ingress LSR is reached or a loop is found.

The rules that govern the use of the Hop Count TLV in Label Mapping messages sent by an LSR R when Loop Detection is enabled are the following:

- R MUST include a Hop Count TLV.
- If R is the egress, the hop count value MUST be 1.
- If the Label Mapping message is being sent to propagate a Label Mapping message received from the next hop to an upstream peer, the hop count value MUST be determined as follows:
  - o If R is a member of the edge set of an LSR domain whose LSRs do not perform 'TTL-decrement' (e.g., an ATM LSR domain or a Frame Relay LSR domain) and the upstream peer is within that domain, R MUST reset the hop count to 1 before propagating the message.
  - o Otherwise, R MUST increment the hop count received from the next hop before propagating the message.
- If the Label Mapping message is not being sent to propagate a Label Mapping message, the hop count value MUST be the result of incrementing R's current knowledge of the hop count learned from previous Label Mapping messages. Note that this hop count value will be unknown if R has not received a Label Mapping message from the next hop.

Any Label Mapping message MAY contain a Path Vector TLV. The rules that govern the mandatory use of the Path Vector TLV in Label Mapping messages sent by LSR R when Loop Detection is enabled are the following:

- If R is the egress, the Label Mapping message need not include a Path Vector TLV.
- If R is sending the Label Mapping message to propagate a Label Mapping message received from the next hop to an upstream peer, then:
  - o If R is merge capable and if R has not previously sent a Label Mapping message to the upstream peer, then it MUST include a Path Vector TLV.
  - o If the received message contains an unknown hop count, then R MUST include a Path Vector TLV.
  - o If R has previously sent a Label Mapping message to the upstream peer, then it MUST include a Path Vector TLV if the received message reports an LSP hop count increase, a change in hop count from unknown to known, or a change from known to unknown.



If the above rules require R include a Path Vector TLV in the Label Mapping message, R computes it as follows:

- o If the received Label Mapping message included a Path Vector, the Path Vector sent upstream MUST be the result of adding R's LSR Id to the received Path Vector.
- o If the received message had no Path Vector, the Path Vector sent upstream MUST be a Path Vector of length 1 containing R's LSR Id.
- If the Label Mapping message is not being sent to propagate a received message upstream, the Label Mapping message MUST include a Path Vector of length 1 containing R's LSR Id.

If R receives a Label Mapping message from its next hop with a Hop Count TLV that exceeds the configured maximum value, or with a Path Vector TLV containing its own LSR Id or that exceeds the maximum allowable length, then R detects that the corresponding LSP contains a loop.

When R detects a loop, it MUST stop using the label for forwarding, drop the Label Mapping message, and signal Loop Detected status to the source of the Label Mapping message.

### 3.8.3. Discussion

If Loop Detection is desired in an MPLS domain, then it should be turned on in ALL LSRs within that MPLS domain, else Loop Detection will not operate properly and may result in undetected loops or in falsely detected loops.

LSRs that are configured for Loop Detection are NOT expected to store the Path Vectors as part of the LSP state.

Note that in a network where only non-merge capable LSRs are present, Path Vectors are passed downstream from ingress to egress, and are not passed upstream. Even when merge is supported, Path Vectors need not be passed upstream along an LSP that is known to reach the egress. When an LSR experiences a change of next hop, it need pass Path Vectors upstream only when it cannot tell from the hop count that the change of next hop does not result in a loop.

In the case of ordered label distribution, Label Mapping messages are propagated from egress toward ingress, naturally creating the Path Vector along the way. In the case of independent label distribution,

an LSR may originate a Label Mapping message for a FEC before receiving a Label Mapping message from its downstream peer for that FEC. In this case, the subsequent Label Mapping message for the FEC received from the downstream peer is treated as an update to LSP attributes, and the Label Mapping message must be propagated upstream. Thus, it is recommended that Loop Detection be configured in conjunction with ordered label distribution, to minimize the number of Label Mapping update messages.

### 3.9. Authenticity and Integrity of LDP Messages

This section specifies a mechanism to protect against the introduction of spoofed TCP segments into LDP session connection streams. The use of this mechanism **MUST** be supported as a configurable option.

The mechanism is based on use of the TCP MD5 Signature Option specified in RFC 2385 [RFC2385] for use by BGP [RFC4271]. See RFC 1321 [RFC1321] for a specification of the MD5 hash function. From a standards maturity point of view, the current document relates to RFC 2385 the same way as RFC 4271 relates to RFC 2385. This is explained in RFC 4278 [RFC4278].

#### 3.9.1. TCP MD5 Signature Option

The following quotes from RFC 2385 [RFC2385] outline the security properties achieved by using the TCP MD5 Signature Option and summarize its operation:

"IESG Note

This document describes current existing practice for securing BGP against certain simple attacks. It is understood to have security weaknesses against concerted attacks."

"Abstract

This memo describes a TCP extension to enhance security for BGP. It defines a new TCP option for carrying an MD5 RFC 1321 [RFC1321] digest in a TCP segment. This digest acts like a signature for that segment, incorporating information known only to the connection end points. Since BGP uses TCP as its transport, using this option in the way described in this paper significantly reduces the danger from certain security attacks on BGP."

"Introduction

The primary motivation for this option is to allow BGP to protect itself against the introduction of spoofed TCP segments into the connection stream. Of particular concern are TCP resets.

To spoof a connection using the scheme described in this paper, an attacker would not only have to guess TCP sequence numbers, but would also have had to obtain the password included in the MD5 digest. This password never appears in the connection stream, and the actual form of the password is up to the application. It could even change during the lifetime of a particular connection so long as this change was synchronized on both ends (although retransmission can become problematical in some TCP implementations with changing passwords).

Finally, there is no negotiation for the use of this option in a connection, rather it is purely a matter of site policy whether or not its connections use the option."

#### "MD5 as a Hashing Algorithm

Since this memo was first issued (under a different title), the MD5 algorithm has been found to be vulnerable to collision search attacks [Dobb], and is considered by some to be insufficiently strong for this type of application.

This memo still specifies the MD5 algorithm, however, since the option has already been deployed operationally, and there was no "algorithm type" field defined to allow an upgrade using the same option number. The original document did not specify a type field since this would require at least one more byte, and it was felt at the time that taking 19 bytes for the complete option (which would probably be padded to 20 bytes in TCP implementations) would be too much of a waste of the already limited option space.

This does not prevent the deployment of another similar option which uses another hashing algorithm (like SHA-1). Also, if most implementations pad the 18 byte option as defined to 20 bytes anyway, it would be just as well to define a new option which contains an algorithm type field.

This would need to be addressed in another document, however."

End of quotes from RFC 2385 [RFC2385].

### 3.9.2. LDP Use of TCP MD5 Signature Option

LDP uses the TCP MD5 Signature Option as follows:

- Use of the MD5 Signature Option for LDP TCP connections is a configurable LSR option.
- An LSR that uses the MD5 Signature Option is configured with a password (shared secret) for each potential LDP peer.
- The LSR applies the MD5 algorithm as specified in RFC 2385 [RFC2385] to compute the MD5 digest for a TCP segment to be sent to a peer. This computation makes use of the peer password as well as the TCP segment.
- When the LSR receives a TCP segment with an MD5 digest, it validates the segment by calculating the MD5 digest (using its own record of the password) and compares the computed digest with the received digest. If the comparison fails, the segment is dropped without any response to the sender.
- The LSR ignores LDP Hellos from any LSR for which a password has not been configured. This ensures that the LSR establishes LDP TCP connections only with LSRs for which a password has been configured.

## 4. Protocol Specification

Previous sections that describe LDP operation have discussed scenarios that involve the exchange of messages among LDP peers. This section specifies the message encodings and procedures for processing the messages.

LDP message exchanges are accomplished by sending LDP protocol data units (PDUs) over LDP session TCP connections.

Each LDP PDU can carry one or more LDP messages. Note that the messages in an LDP PDU need not be related to one another. For example, a single PDU could carry a message advertising FEC-label bindings for several FECs, another message requesting label bindings for several other FECs, and a third Notification message signaling some event.

### 4.1. LDP PDUs

Each LDP PDU is an LDP header followed by one or more LDP messages. The LDP header is:

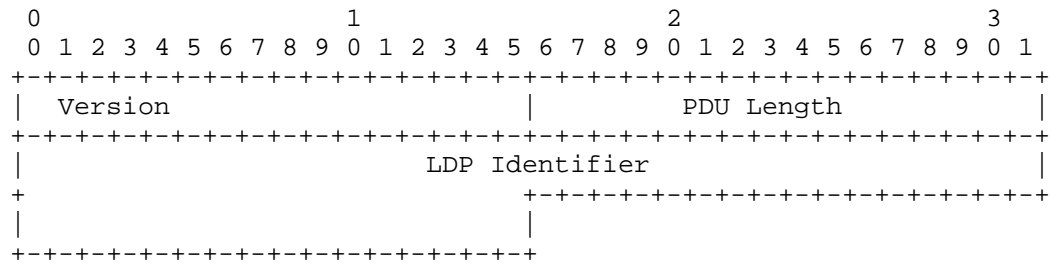


Figure 2: LDP PDU

**Version**

Two octet unsigned integer containing the version number of the protocol. This version of the specification specifies LDP protocol version 1.

**PDU Length**

Two octet integer specifying the total length of this PDU in octets, excluding the Version and PDU Length fields.

The maximum allowable PDU Length is negotiable when an LDP session is initialized. Prior to completion of the negotiation, the maximum allowable length is 4096 octets.

**LDP Identifier**

Six octet field that uniquely identifies the label space of the sending LSR for which this PDU applies. The first four octets identify the LSR and MUST be a globally unique value. It SHOULD be a 32-bit router Id assigned to the LSR and also used to identify it in Loop Detection Path Vectors. The last two octets identify a label space within the LSR. For a platform-wide label space, these SHOULD both be zero.

Note that there is no alignment requirement for the first octet of an LDP PDU.

**4.2. LDP Procedures**

LDP defines messages, TLVs, and procedures in the following areas:

- Peer discovery
- Session management
- Label distribution
- Notification of errors and advisory information

The sections that follow describe the message and TLV encodings for these areas and the procedures that apply to them.

The label distribution procedures are complex and are difficult to describe fully, coherently, and unambiguously as a collection of separate message and TLV specifications.

Appendix A, "LDP Label Distribution Procedures", describes the label distribution procedures in terms of label distribution events that may occur at an LSR and how the LSR must respond. Appendix A is the specification of LDP label distribution procedures. If a procedure described elsewhere in this document conflicts with Appendix A, Appendix A specifies LDP behavior.

### 4.3. Type-Length-Value Encoding

LDP uses a Type-Length-Value (TLV) encoding scheme to encode much of the information carried in LDP messages.

An LDP TLV is encoded as a 2 octet field that uses 14 bits to specify a Type and 2 bits to specify behavior when an LSR doesn't recognize the Type, followed by a 2 octet Length field, followed by a variable length Value field.

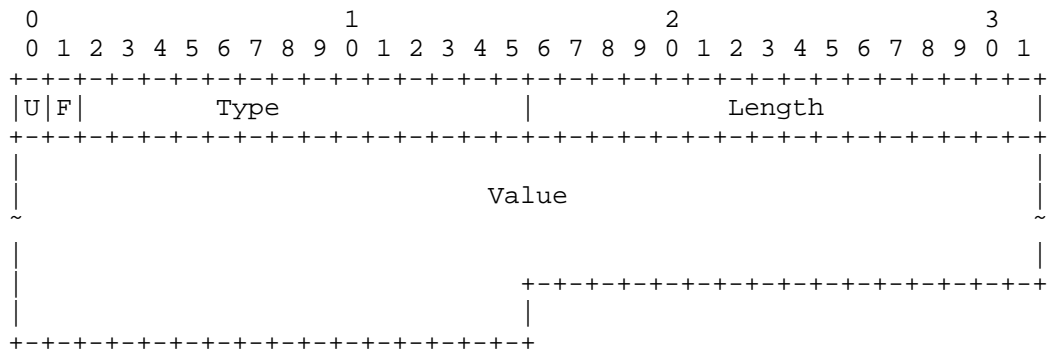


Figure 3: TLV Encoding

## U-bit

Unknown TLV bit. Upon receipt of an unknown TLV, if U is clear (=0), a notification MUST be returned to the message originator and the entire message MUST be ignored; if U is set (=1), the unknown TLV MUST be silently ignored and the rest of the message processed as if the unknown TLV did not exist. The sections following that define TLVs specify a value for the U-bit.

## F-bit

Forward unknown TLV bit. This bit applies only when the U-bit is set and the LDP message containing the unknown TLV is to be forwarded. If F is clear (=0), the unknown TLV is not forwarded with the containing message; if F is set (=1), the unknown TLV is forwarded with the containing message. The sections following that define TLVs specify a value for the F-bit. By setting both the U- and F-bits, a TLV can be propagated as opaque data through nodes that do not recognize the TLV.

#### Type

Encodes how the Value field is to be interpreted.

#### Length

Specifies the length of the Value field in octets.

#### Value

Octet string of Length octets that encodes information to be interpreted as specified by the Type field.

Note that there is no alignment requirement for the first octet of a TLV.

Note that the Value field itself may contain TLV encodings. That is, TLVs may be nested.

The TLV encoding scheme is very general. In principle, everything appearing in an LDP PDU could be encoded as a TLV. This specification does not use the TLV scheme to its full generality. It is not used where its generality is unnecessary and its use would waste space unnecessarily. These are usually places where the type of a value to be encoded is known, for example by its position in a message or an enclosing TLV, and the length of the value is fixed or readily derivable from the value encoding itself.

Some of the TLVs defined for LDP are similar to one another. For example, there is a Generic Label TLV, an ATM Label TLV, and a Frame Relay TLV; see Sections "Generic Label TLV", "ATM Label TLV", and "Frame Relay TLV".

While it is possible to think about TLVs related in this way in terms of a TLV type that specifies a TLV class and a TLV subtype that specifies a particular kind of TLV within that class, this specification does not formalize the notion of a TLV subtype.

The specification assigns type values for related TLVs, such as the label TLVs, from a contiguous block in the 16-bit TLV type number space.

Section 4.8 "TLV Summary" lists the TLVs defined in this version of the protocol and the section in this document that describes each.

#### 4.4. TLV Encodings for Commonly Used Parameters

There are several parameters used by more than one LDP message. The TLV encodings for these commonly used parameters are specified in this section.

##### 4.4.1. FEC TLV

Labels are bound to Forwarding Equivalence Classes (FECs). A FEC is a list of one or more FEC elements. The FEC TLV encodes FEC items.

Its encoding is:

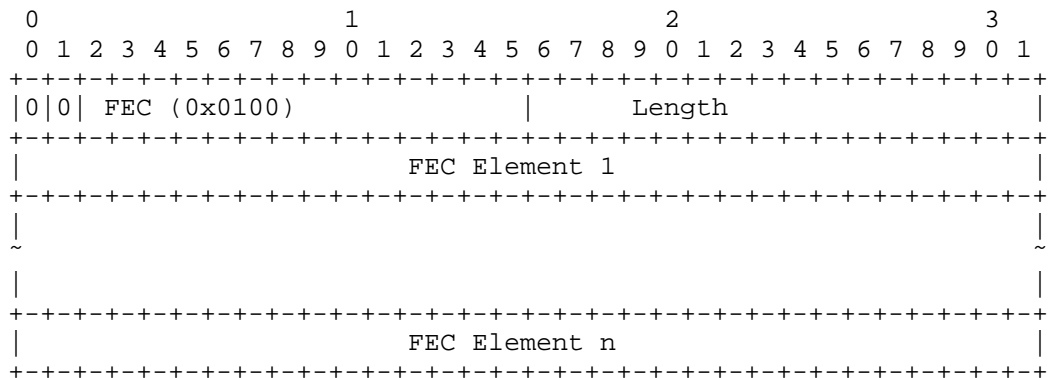


Figure 4: FEC TLV Encoding

##### FEC Element 1 to FEC Element n

There are several types of FEC elements; see Section "FECs". The FEC element encoding depends on the type of FEC element.

A FEC Element value is encoded as a 1 octet field that specifies the element type, and a variable length field that is the type-dependent element value. Note that while the representation of the FEC element value is type-dependent, the FEC element encoding itself is one where standard LDP TLV encoding is not used.

The FEC Element value encoding is:



FEC Element type name	Type	Value
Wildcard	0x01	No value; i.e., 0 value octets; see below.
Prefix	0x02	See below.

Table 1: FEC Element Types

Note that this version of LDP supports the use of multiple FEC Elements per FEC for the Label Mapping message only. The use of multiple FEC Elements in other messages is not permitted in this version, and is a subject for future study.

#### Wildcard FEC Element

To be used only in the Label Withdraw and Label Release messages. Indicates the withdraw/release is to be applied to all FECs associated with the label within the following label TLV. Must be the only FEC Element in the FEC TLV.

#### Prefix FEC Element value encoding:

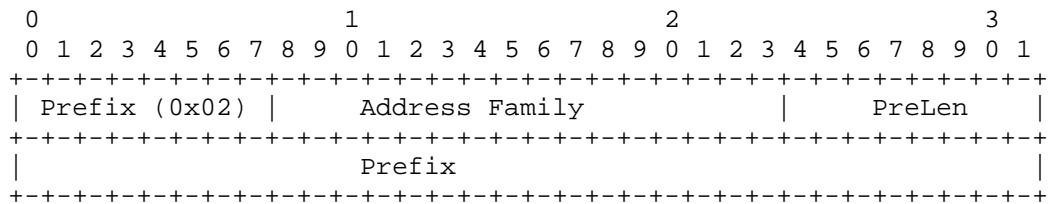


Figure 5: Prefix FEC Element Value Encoding

#### Address Family

Two octet quantity containing a value from ADDRESS FAMILY NUMBERS in [ASSIGNED\_AF] that encodes the address family for the address prefix in the Prefix field.

#### PreLen

One octet unsigned integer containing the length in bits of the address prefix that follows. A length of zero indicates a prefix that matches all addresses (the default destination); in this case, the Prefix itself is zero octets).

#### Prefix

An address prefix encoded according to the Address Family field, whose length, in bits, was specified in the PreLen field, padded to a byte boundary.

#### 4.4.1.1. FEC Procedures

If in decoding a FEC TLV an LSR encounters a FEC Element with an Address Family it does not support, it SHOULD stop decoding the FEC TLV, abort processing the message containing the TLV, and send an "Unsupported Address Family" Notification message to its LDP peer signaling an error.

If it encounters a FEC Element type it cannot decode, it SHOULD stop decoding the FEC TLV, abort processing the message containing the TLV, and send an "Unknown FEC" Notification message to its LDP peer signaling an error.

#### 4.4.2. Label TLVs

Label TLVs encode labels. Label TLVs are carried by the messages used to advertise, request, release, and withdraw label mappings.

There are several different kinds of Label TLVs that can appear in situations that require a Label TLV.

#### 4.4.2.1. Generic Label TLV

An LSR uses Generic Label TLVs to encode labels for use on links for which label values are independent of the underlying link technology. Examples of such links are PPP and Ethernet.

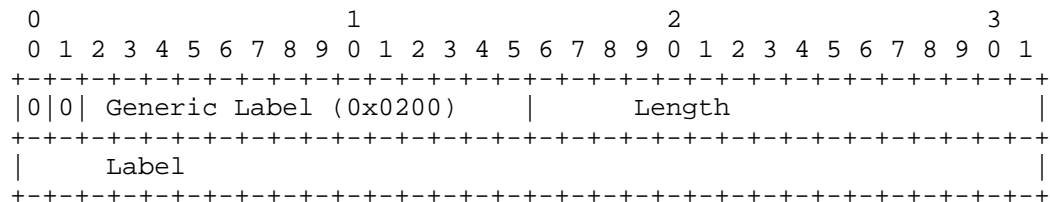


Figure 6: Generic Label

For further information, see RFC 3032 [RFC3032].

Label

This is a 20-bit label value represented as a 20-bit number in a 4 octet field as follows:

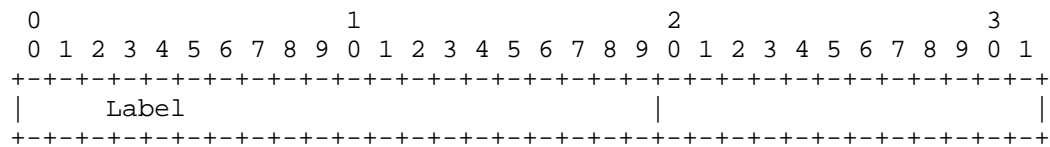


Figure 7: Label

For further information, see RFC 3032 [RFC3032].

#### 4.4.2.2. ATM Label TLV

An LSR uses ATM Label TLVs to encode labels for use on ATM links.

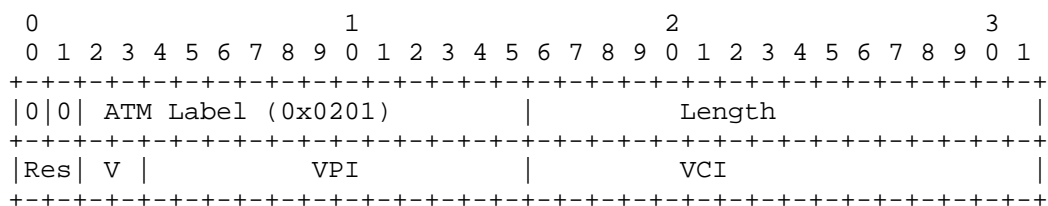


Figure 8: ATM Label TLV

Res

This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

V-bits

Two-bit switching indicator. If V-bits is 00, both the VPI and VCI are significant. If V-bits is 01, only the VPI field is significant. If V-bit is 10, only the VCI is significant.

VPI

Virtual Path Identifier. If VPI is less than 12-bits it SHOULD be right justified in this field and preceding bits SHOULD be set to 0.

VCI

Virtual Channel Identifier. If the VCI is less than 16-bits, it SHOULD be right justified in the field and the preceding bits MUST be set to 0. If Virtual Path switching is indicated in the V-bits field, then this field MUST be ignored by the receiver and set to 0 by the sender.

Editors Note: We have a discussion on whether we need ATM in the Internet Standard version of the LDP specification.

#### 4.4.2.3. Frame Relay Label TLV

An LSR uses Frame Relay Label TLVs to encode labels for use on Frame Relay links.

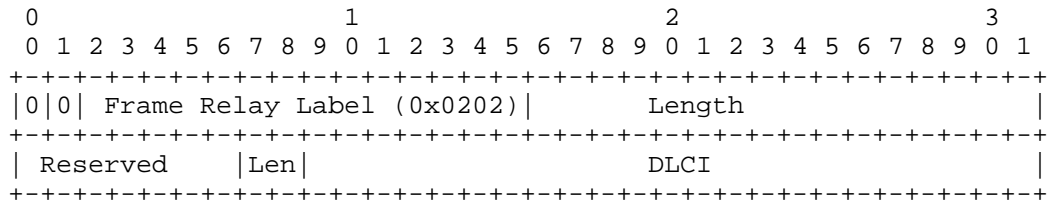


Figure 9: Frame Relay Label TLV

##### Reserved

This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt.

##### Len

This field specifies the number of bits of the DLCI. The following values are supported:

+-----+	
Code	bits of DLCI
+-----+	
0	10
1	reserved
2	32
3	reserved
+-----+	

Table 2: Frame Relay Label Length codes

##### DLCI

The Data Link Connection Identifier

For a 10-bit DLCI, the encoding is:

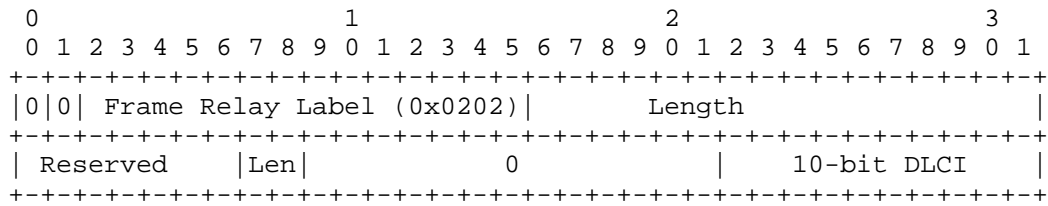


Figure 10: Frame Relay Label TLV for a 10 bit DLCI

For a 23-bit DLCI, the encoding is:

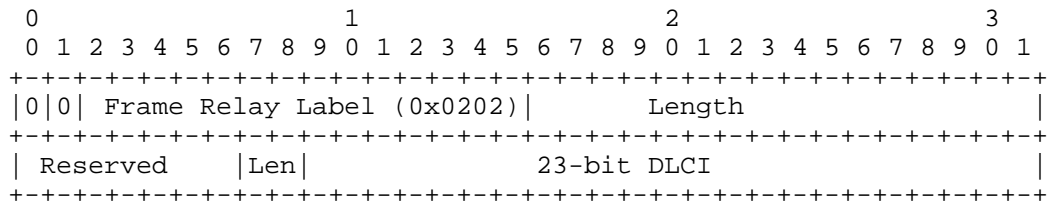


Figure 11: Frame Relay Label TLV for a 10 bit DLCI

For further information, see RFC 3034 [RFC3034].

#### 4.4.3. Address List TLV

The Address List TLV appears in Address and Address Withdraw messages.

Its encoding is:

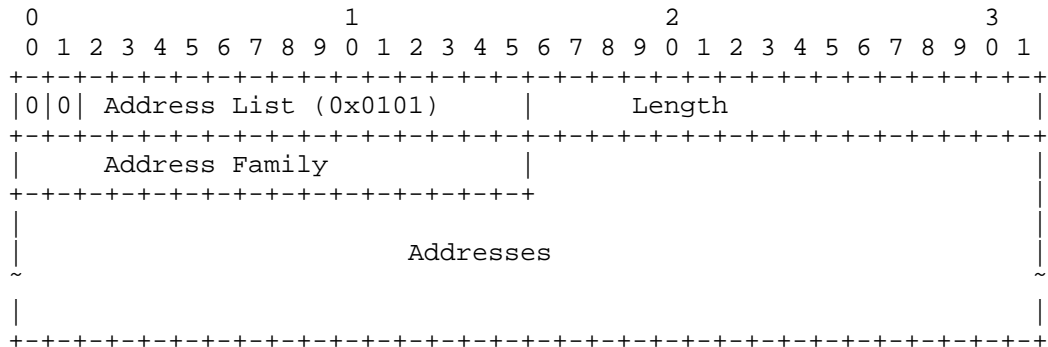


Figure 12: Address List TLV

Address Family

Two octet quantity containing a value from ADDRESS FAMILY NUMBERS in [ASSIGNED\_AF] that encodes the addresses contained in the Addresses field.

#### Addresses

A list of addresses from the specified Address Family. The encoding of the individual addresses depends on the Address Family.  
The following address encodings are defined by this version of the protocol:

Address Family	Address Encoding
IPv4	4 octet full IPv4vaddress
IPv6	16 octet full IPv6 address

Table 3: Address Families

#### 4.4.4. Hop Count TLV

The Hop Count TLV appears as an optional field in messages that set up LSPs. It calculates the number of LSR hops along an LSP as the LSP is being set up.

Note that setup procedures for LSPs that traverse ATM and Frame Relay links require use of the Hop Count TLV (see RFC 3035 [RFC3035] and RFC 3034 [RFC3034]).

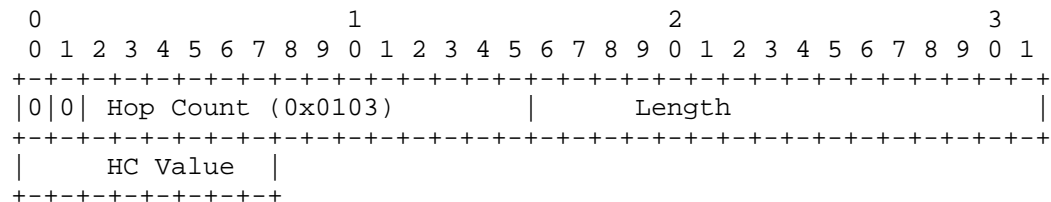


Figure 13: Hop Count TLV

#### HC Value

1 octet unsigned integer hop count value.

#### 4.4.4.1. Hop Count Procedures

During setup of an LSP, an LSR R may receive a Label Mapping or Label Request message for the LSP that contains the Hop Count TLV. If it does, it SHOULD record the hop count value.

If LSR R then propagates the Label Mapping message for the LSP to an upstream peer or the Label Request message to a downstream peer to continue the LSP setup, it must determine a hop count to include in the propagated message as follows:

- If the message is a Label Request message, R MUST increment the received hop count;
- If the message is a Label Mapping message, R determines the hop count as follows:
  - o If R is a member of the edge set of an LSR domain whose LSRs do not perform 'TTL-decrement' and the upstream peer is within that domain, R MUST reset the hop count to 1 before propagating the message.
  - o Otherwise, R MUST increment the received hop count.

The first LSR in the LSP (ingress for a Label Request message, egress for a Label Mapping message) SHOULD set the hop count value to 1.

By convention, a value of 0 indicates an unknown hop count. The result of incrementing an unknown hop count is itself an unknown hop count (0).

Use of the unknown hop count value greatly reduces the signaling overhead when independent control is used. When a new LSP is established, each LSR starts with an unknown hop count. Addition of a new LSR whose hop count is also unknown does not cause a hop count update to be propagated upstream since the hop count remains unknown. When the egress is finally added to the LSP, then the LSRs propagate hop count updates upstream via Label Mapping messages.

Without use of the unknown hop count, each time a new LSR is added to the LSP a hop count update would need to be propagated upstream if the new LSR is closer to the egress than any of the other LSRs. These updates are useless overhead since they don't reflect the hop count to the egress.

From the perspective of the ingress node, the fact that the hop count is unknown implies nothing about whether a packet sent on the LSP

will actually make it to the egress. All it implies is that the hop count update from the egress has not yet reached the ingress.

If an LSR receives a message containing a Hop Count TLV, it MUST check the hop count value to determine whether the hop count has exceeded its configured maximum allowable value. If so, it MUST behave as if the containing message has traversed a loop by sending a Notification message signaling Loop Detected in reply to the sender of the message.

If Loop Detection is configured, the LSR MUST follow the procedures specified in Section 3.8 "Loop Detection".

#### 4.4.5. Path Vector TLV

The Path Vector TLV is used with the Hop Count TLV in Label Request and Label Mapping messages to implement the optional LDP Loop Detection mechanism. See Section "Loop Detection". Its use in the Label Request message records the path of LSRs the request has traversed. Its use in the Label Mapping message records the path of LSRs a label advertisement has traversed to set up an LSP. Its encoding is:

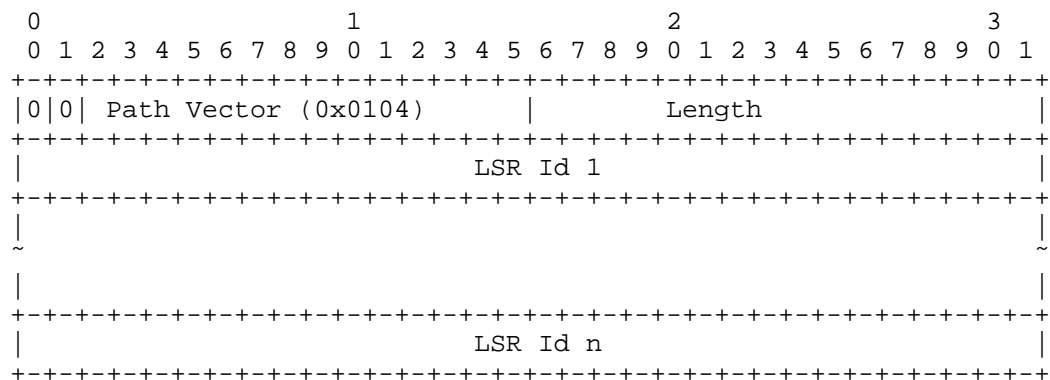


Figure 14: Path Vector TLV

#### One or more LSR Ids

A list of router-ids indicating the path of LSRs the message has traversed. Each LSR Id is the first four octets (router-id) of the LDP Identifier for the corresponding LSR. This ensures it is unique within the LSR network.



#### 4.4.5.1. Path Vector Procedures

The Path Vector TLV is carried in Label Mapping and Label Request messages when Loop Detection is configured.

##### 4.4.5.1.1. Label Request Path Vector

Section 3.8 "Loop Detection" specifies situations when an LSR must include a Path Vector TLV in a Label Request message.

An LSR that receives a Path Vector in a Label Request message MUST perform the procedures described in Section "Loop Detection".

If the LSR detects a loop, it MUST reject the Label Request message.

The LSR MUST:

1. Transmit a Notification message to the sending LSR signaling "Loop Detected".
2. Not propagate the Label Request message further.

Note that a Label Request message with a Path Vector TLV is forwarded until:

1. A loop is found,
2. The LSP egress is reached, or
3. The maximum Path Vector limit or maximum Hop Count limit is reached. This is treated as if a loop had been detected.

##### 4.4.5.1.2. Label Mapping Path Vector

Section 3.8 "Loop Detection" specifies the situations when an LSR must include a Path Vector TLV in a Label Mapping message.

An LSR that receives a Path Vector in a Label Mapping message MUST perform the procedures described in Section "Loop Detection".

If the LSR detects a loop, it MUST reject the Label Mapping message in order to prevent a forwarding loop. The LSR MUST:

1. Transmit a Label Release message carrying a Status TLV to the sending LSR to signal "Loop Detected".
2. Not propagate the message further.



32-bit unsigned integer encoding the event being signaled. The structure of a Status Code is:

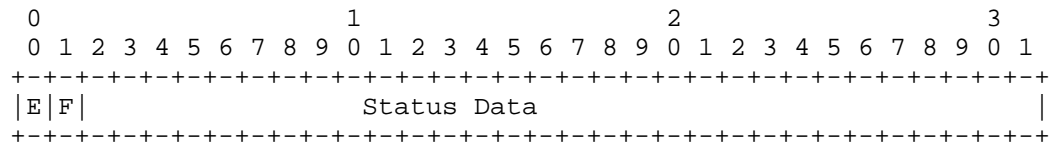


Figure 16: Path Vector TLV

#### E-bit

Fatal error bit. If set (=1), this is a fatal Error Notification. If clear (=0), this is an Advisory Notification.

#### F-bit

Forward bit. If set (=1), the notification SHOULD be forwarded to the LSR for the next-hop or previous-hop for the LSP, if any, associated with the event being signaled. If clear (=0), the notification SHOULD NOT be forwarded.

#### Status Data

30-bit unsigned integer that specifies the status information.

This specification defines Status Codes (32-bit unsigned integers with the above encoding).

A Status Code of 0 signals success.

#### Message ID

If non-zero, 32-bit value that identifies the peer message to which the Status TLV refers. If zero, no specific peer message is being identified.

#### Message Type

If non-zero, the type of the peer message to which the Status TLV refers. If zero, the Status TLV does not refer to any specific message type.

Note that use of the Status TLV is not limited to Notification messages. A message other than a Notification message may carry a Status TLV as an Optional Parameter. When a message other than a Notification carries a Status TLV, the U-bit of the Status TLV SHOULD

be set to 1 to indicate that the receiver SHOULD silently discard the TLV if unprepared to handle it.

#### 4.5. LDP Messages

All LDP messages have the following format:

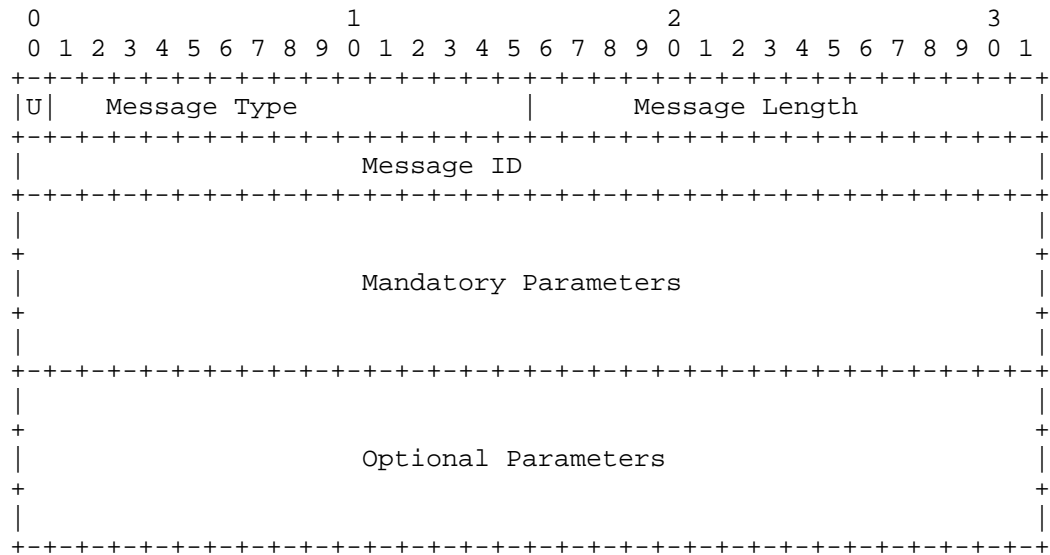


Figure 17: LDP message format

##### U-bit

Unknown message bit. Upon receipt of an unknown message, if U is clear (=0), a notification is returned to the message originator; if U is set (=1), the unknown message is silently ignored. The sections following that define messages specify a value for the U-bit.

##### Message Type

Identifies the type of message.

##### Message Length

Specifies the cumulative length in octets of the Message ID, Mandatory Parameters, and Optional Parameters.

##### Message ID

32-bit value used to identify this message. Used by the sending LSR to facilitate identifying Notification messages that may apply to this message. An LSR sending a Notification message in response to this message SHOULD include this Message ID in the Status TLV carried by the Notification message; see Section 4.5.1 "Notification Message".

#### Mandatory Parameters

Variable length set of required message parameters. Some messages have no required parameters.

For messages that have required parameters, the required parameters MUST appear in the order specified by the individual message specifications in the sections that follow.

#### Optional Parameters

Variable length set of optional message parameters. Many messages have no optional parameters.

For messages that have optional parameters, the optional parameters may appear in any order.

Note that there is no alignment requirement for the first octet of an LDP message and that there is no padding at the end of a message; that is, parameters can end at odd-byte boundaries.

The following message types are defined in this version of LDP:

Message Name	Section Number and Title
Notification	Section 4.5.1 "Notification Message"
Hello	Section 4.5.2 "Hello Message"
Initialization	Section 4.5.3 "Initialization Message"
KeepAlive	Section 4.5.4 "KeepAlive Message"
Address	Section 4.5.5 "Address Message"
Address Withdraw	Section 4.5.6 "Address Withdraw Message"
Label Mapping	Section 4.5.7 "Label Mapping Message"
Label Request	Section 4.5.8 "Label Request Message"
Label Abort Request	Section 4.5.5 "Label Abort Request Message"
Label Withdraw	Section 4.5.10 "Label Withdraw Message"
Label Release	Section 4.5.11 "Label Release Message"

Table 4: LDP Messages

The sections that follow specify the encodings and procedures for these messages.

Some of the above messages are related to one another, for example the Label Mapping, Label Request, Label Withdraw, and Label Release messages.

While it is possible to think about messages related in this way in terms of a message type that specifies a message class and a message subtype that specifies a particular kind of message within that class, this specification does not formalize the notion of a message subtype.

The specification assigns type values for related messages, such as the Label messages, from a contiguous block in the 16-bit message type number space.

#### 4.5.1. Notification Message

An LSR sends a Notification message to inform an LDP peer of a significant event. A Notification message signals a fatal error or provides advisory information such as the outcome of processing an LDP message or the state of the LDP session.

The encoding for the Notification message is:

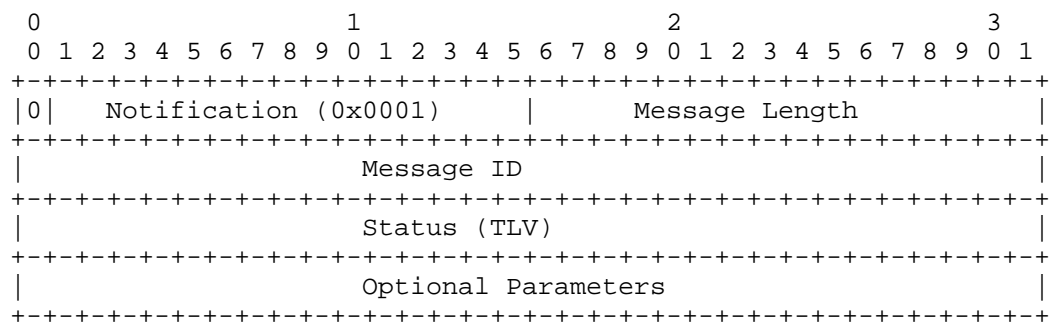


Figure 18: Notification Message

Message ID

32-bit value used to identify this message.

Status TLV

Indicates the event being signaled. The encoding for the Status TLV is specified in Section 4.4.6 "Status TLV".

### Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. The following Optional Parameters are generic and may appear in any Notification message:

Optional Parameter	Type	Length	Value
Extended Status	0x0301	4	See below
Returned PDU	0x0302	var	See below
Returned Message	0x0303	var	See below

Table 5: LDP Messages

Other Optional Parameters, specific to the particular event being signaled by the Notification messages, may appear. These are described elsewhere.

#### Extended Status

The 4 octet value is an Extended Status Code that encodes additional information that supplements the status information contained in the Notification Status Code.

#### Returned PDU

An LSR uses this parameter to return part of an LDP PDU to the LSR that sent it. The value of this TLV is the PDU header and as much PDU data following the header as appropriate for the condition being signaled by the Notification message.

#### Returned Message

An LSR uses this parameter to return part of an LDP message to the LSR that sent it. The value of this TLV is the message type and length fields and as much message data following the type and length fields as appropriate for the condition being signaled by the Notification message.

#### 4.5.1.1. Notification Message Procedures

If an LSR encounters a condition requiring it to notify its peer with advisory or error information, it sends the peer a Notification message containing a Status TLV that encodes the information and optionally additional TLVs that provide more information about the condition.

If the condition is one that is a fatal error, the Status Code carried in the Notification will indicate that. In this case, after sending the Notification message the LSR SHOULD terminate the LDP session by closing the session TCP connection and discard all state associated with the session, including all label-FEC bindings learned via the session.

When an LSR receives a Notification message that carries a Status Code that indicates a fatal error, it SHOULD terminate the LDP session immediately by closing the session TCP connection and discard all state associated with the session, including all label-FEC bindings learned via the session.

The above statement does not apply to the processing of the Shutdown message in the session initialization procedure. When an LSR receives a Shutdown message during session initialization, it SHOULD transmit a Shutdown message and then close the transport connection.

#### 4.5.1.2. Events Signaled by Notification Messages

It is useful for descriptive purpose to classify events signaled by Notification messages into the following categories.

##### 4.5.1.2.1. Malformed PDU or Message

Malformed LDP PDUs or messages that are part of the LDP Discovery mechanism are handled by silently discarding them.

An LDP PDU received on a TCP connection for an LDP session is malformed if:

- The LDP Identifier in the PDU header is unknown to the receiver, or it is known but is not the LDP Identifier associated by the receiver with the LDP peer for this LDP session. This is a fatal error signaled by the Bad LDP Identifier Status Code.
- The LDP protocol version is not supported by the receiver, or it is supported but is not the version negotiated for the session during session establishment. This is a fatal error signaled by the Bad Protocol Version Status Code.
- The PDU Length field is too small (< 14) or too large (> maximum PDU length). This is a fatal error signaled by the Bad PDU Length Status Code. Section "Initialization Message" describes how the maximum PDU length for a session is determined.

An LDP message is malformed if:



- The Message Type is unknown.

If the Message Type is  $< 0x8000$  (high order bit = 0), it is an error signaled by the Unknown Message Type Status Code.

If the Message Type is  $\geq 0x8000$  (high order bit = 1), it is silently discarded.

- The Message Length is too large, that is, indicates that the message extends beyond the end of the containing LDP PDU. This is a fatal error signaled by the Bad Message Length Status Code.
- The Message Length is too small, that is, smaller than the smallest possible value component. This is a fatal error signaled by the Bad Message Length Status Code.
- The message is missing one or more Mandatory Parameters. This is a non-fatal error signaled by the Missing Message Parameters Status Code.

#### 4.5.1.2.2. Unknown or Malformed TLV

Malformed TLVs contained in LDP messages that are part of the LDP Discovery mechanism are handled by silently discarding the containing message.

A TLV contained in an LDP message received on a TCP connection of an LDP is malformed if:

- The TLV Length is too large, that is, indicates that the TLV extends beyond the end of the containing message. This is a fatal error signaled by the Bad TLV Length Status Code.
- The TLV type is unknown.

If the TLV type is  $< 0x8000$  (high order bit = 0), it is an error signaled by the Unknown TLV Status Code.

If the TLV type is  $\geq 0x8000$  (high order bit = 1), the TLV is silently dropped.

- The TLV Value is malformed. This occurs when the receiver handles the TLV but cannot decode the TLV Value. This is interpreted as indicative of a bug in either the sending or receiving LSR. It is a fatal error signaled by the Malformed TLV Value Status Code.

#### 4.5.1.2.3. Session KeepAlive Timer Expiration

This is a fatal error signaled by the KeepAlive Timer Expired Status Code.

#### 4.5.1.2.4. Unilateral Session Shutdown

This is a fatal event signaled by the Shutdown Status Code. The Notification message may optionally include an Extended Status TLV to provide a reason for the Shutdown. The sending LSR terminates the session immediately after sending the Notification.

#### 4.5.1.2.5. Initialization Message Events

The session initialization negotiation (see Section "Session Initialization") may fail if the session parameters received in the Initialization message are unacceptable. This is a fatal error. The specific Status Code depends on the parameter deemed unacceptable, and is defined in Section 4.5.3 "Initialization Message".

#### 4.5.1.2.6. Events Resulting from Other Messages

Messages other than the Initialization message may result in events that must be signaled to LDP peers via Notification messages. These events and the Status Codes used in the Notification messages to signal them are described in the sections that describe these messages.

#### 4.5.1.2.7. Internal Errors

An LDP implementation may be capable of detecting problem conditions specific to its implementation. When such a condition prevents an implementation from interacting correctly with a peer, the implementation should, when capable of doing so, use the Internal Error Status Code to signal the peer. This is a fatal error.

#### 4.5.1.2.8. Miscellaneous Events

These are events that fall into none of the categories above. There are no miscellaneous events defined in this version of the protocol.

#### 4.5.2. Hello Message

LDP Hello messages are exchanged as part of the LDP Discovery Mechanism; see Section 3.4 "LDP Discovery".

The encoding for the Hello message is:

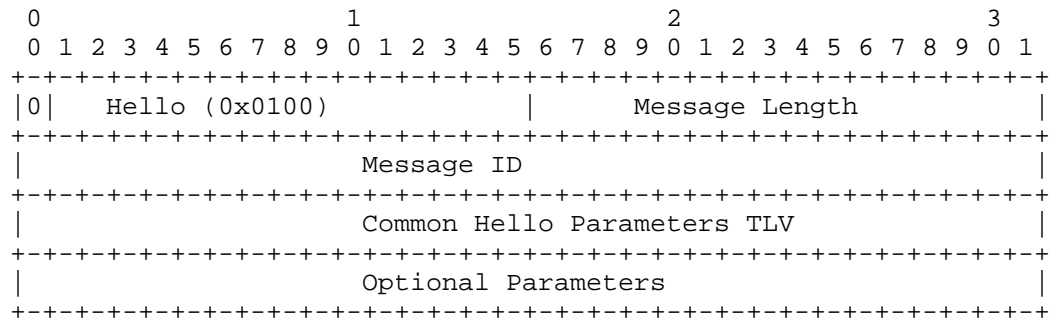


Figure 19: Hello Message

**Message ID**

32-bit value used to identify this message.

**Common Hello Parameters TLV**

Specifies parameters common to all Hello messages. The encoding for the Common Hello Parameters TLV is:

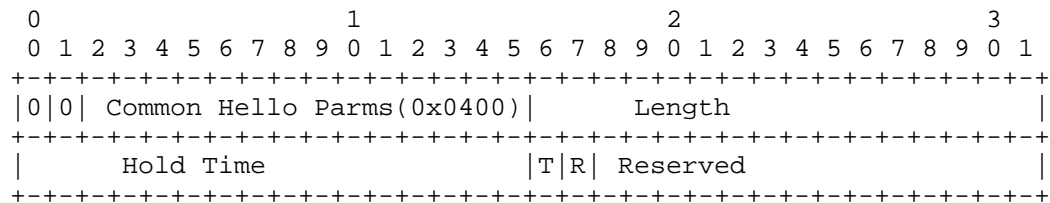


Figure 20: Hello Message

**Hold Time**

Hello hold time in seconds. An LSR maintains a record of Hellos received from potential peers (see Section 4.5.2.1 "Hello Message Procedures"). Hello Hold Time specifies the time the sending LSR will maintain its record of Hellos from the receiving LSR without receipt of another Hello.

A pair of LSRs negotiates the hold times they use for Hellos from each other. Each proposes a hold time. The hold time used is the minimum of the hold times proposed in their Hellos.

A value of 0 means use the default, which is 15 seconds for Link Hellos and 45 seconds for Targeted Hellos. A value of 0xffff means infinite.

## T, Targeted Hello

A value of 1 specifies that this Hello is a Targeted Hello. A value of 0 specifies that this Hello is a Link Hello.

## R, Request Send Targeted Hellos

A value of 1 requests the receiver to send periodic Targeted Hellos to the source of this Hello. A value of 0 makes no request.

An LSR initiating Extended Discovery sets R to 1. If R is 1, the receiving LSR checks whether it has been configured to send Targeted Hellos to the Hello source in response to Hellos with this request. If not, it ignores the request. If so, it initiates periodic transmission of Targeted Hellos to the Hello source.

## Reserved

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

## Optional Parameters

This variable length field of the Hello message contains 0 or more parameters, each encoded as a TLV. The optional parameters defined by this version of the protocol are:

Optional Parameters	Type	Length	Value
IPv4 Transport Address	0x0401	4	See below
Configuration Sequence Number	0x0402	4	See below
IPv6 Transport Address	0x0403	16	See below

Table 6: Optional Hello Message Parameters

## IPv4 Transport Address

Specifies the IPv4 address to be used for the sending LSR when opening the LDP session TCP connection. If this optional TLV is not present, the IPv4 source address for the UDP packet carrying the Hello SHOULD be used.

## Configuration Sequence Number

Specifies a 4 octet unsigned configuration sequence number that identifies the configuration state of the sending LSR. Used by the receiving LSR to detect configuration changes on the sending LSR.

#### IPv6 Transport Address

Specifies the IPv6 address to be used for the sending LSR when opening the LDP session TCP connection. If this optional TLV is not present the IPv6 source address for the UDP packet carrying the Hello SHOULD be used.

#### 4.5.2.1. Hello Message Procedures

An LSR receiving Hellos from another LSR maintains a Hello adjacency corresponding to the Hellos. The LSR maintains a hold timer with the Hello adjacency, which it restarts whenever it receives a Hello that matches the Hello adjacency. If the hold timer for a Hello adjacency expires the LSR discards the Hello adjacency: see Section 3.5.5 "Maintaining Hello Adjacencies" and Section 3.5.6 "Maintaining LDP Sessions".

We recommend that the interval between Hello transmissions be at most one third of the Hello hold time.

An LSR processes a received LDP Hello as follows:

1. The LSR checks whether the Hello is acceptable. The criteria for determining whether a Hello is acceptable are implementation dependent (see below for example criteria).
2. If the Hello is not acceptable, the LSR ignores it.
3. If the Hello is acceptable, the LSR checks whether it has a Hello adjacency for the Hello source. If so, it restarts the hold timer for the Hello adjacency. If not, it creates a Hello adjacency for the Hello source and starts its hold timer.
4. If the Hello carries any optional TLVs, the LSR processes them (see below).
5. Finally, if the LSR has no LDP session for the label space specified by the LDP Identifier in the PDU header for the Hello, it follows the procedures of Section "LDP Session Establishment".

The following are examples of acceptability criteria for Link and Targeted Hellos:

A Link Hello is acceptable if the interface on which it was received has been configured for label switching.

A Targeted Hello from source address A is acceptable if either:

- The LSR has been configured to accept Targeted Hellos, or
- The LSR has been configured to send Targeted Hellos to A.

The following describes how an LSR processes Hello optional TLVs:

#### Transport Address

The LSR associates the specified transport address with the Hello adjacency.

#### Configuration Sequence Number

The Configuration Sequence Number optional parameter is used by the sending LSR to signal configuration changes to the receiving LSR. When a receiving LSR playing the active role in LDP session establishment detects a change in the sending LSR configuration, it may clear the session setup backoff delay, if any, associated with the sending LSR (see Section 3.5.3 "Session Initialization").

A sending LSR using this optional parameter is responsible for maintaining the configuration sequence number it transmits in Hello messages. Whenever there is a configuration change on the sending LSR, it increments the configuration sequence number.

#### 4.5.3. Initialization Message

Note: We have an open discussion on whether we can remove ATM and FR from this document, if we decide to do that this section needs to be revisited.

The LDP Initialization message is exchanged as part of the LDP session establishment procedure; see Section 3.5.1 "LDP Session Establishment".

The encoding for the Initialization message is:

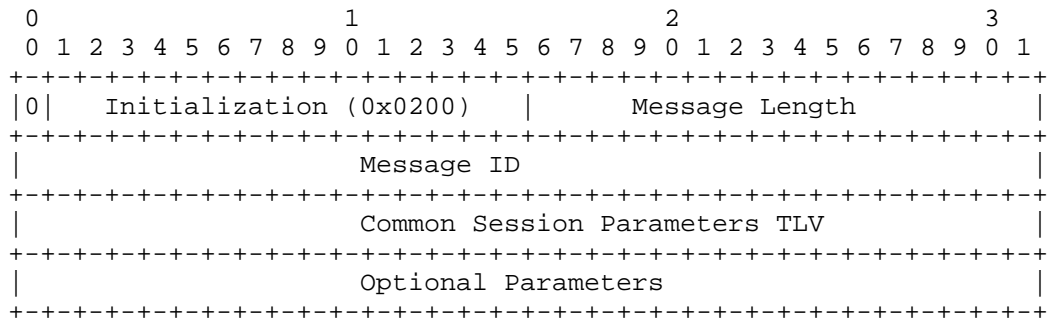


Figure 21: Initialization Message

**Message ID**

32-bit value used to identify this message.

**Common Session Parameters TLV**

Specifies values proposed by the sending LSR for parameters that must be negotiated for every LDP session.

The encoding for the Common Session Parameters TLV is:

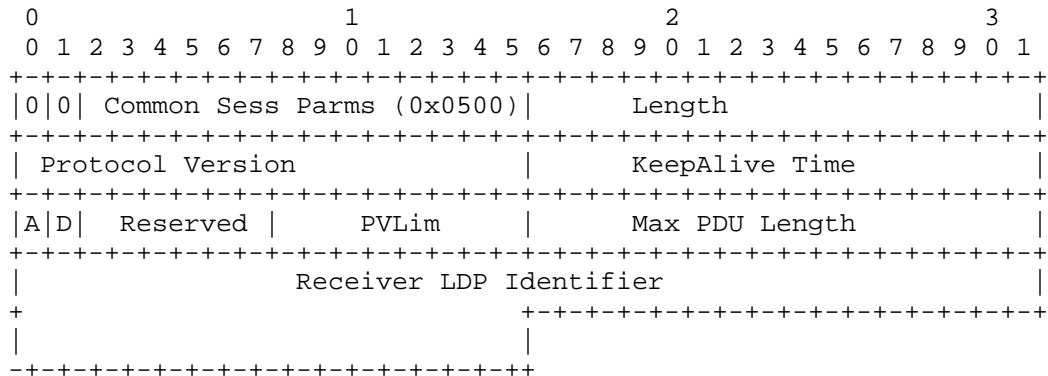


Figure 22: Common Session Parameters TLV

**Protocol Version**

Two octet unsigned integer containing the version number of the protocol. This version of the specification specifies LDP protocol version 1.

**KeepAlive Time**

Two octet unsigned non zero integer that indicates the number of seconds that the sending LSR proposes for the value of the KeepAlive Time. The receiving LSR MUST calculate the value of the KeepAlive Timer by using the smaller of its proposed KeepAlive Time and the KeepAlive Time received in the PDU. The value chosen for KeepAlive Time indicates the maximum number of seconds that may elapse between the receipt of successive PDUs from the LDP peer on the session TCP connection. The KeepAlive Timer is reset each time a PDU arrives.

#### A, Label Advertisement Discipline

Indicates the type of Label advertisement. A value of 0 means Downstream Unsolicited advertisement; a value of 1 means Downstream On Demand.

If one LSR proposes Downstream Unsolicited and the other proposes Downstream on Demand, the rules for resolving this difference is:

- If the session is for a label-controlled ATM link or a label-controlled Frame Relay link, then Downstream on Demand MUST be used.
- Otherwise, Downstream Unsolicited MUST be used.

If the label advertisement discipline determined in this way is unacceptable to an LSR, it MUST send a Session Rejected/Parameters Advertisement Mode Notification message in response to the Initialization message and not establish the session.

#### D, Loop Detection

Indicates whether Loop Detection based on Path Vectors is enabled. A value of 0 means that Loop Detection is disabled; a value of 1 means that Loop Detection is enabled.

#### PVLim, Path Vector Limit

The configured maximum Path Vector length. MUST be 0 if Loop Detection is disabled (D = 0). If the Loop Detection procedures would require the LSR to send a Path Vector that exceeds this limit, the LSR will behave as if a loop had been detected for the FEC in question.

When Loop Detection is enabled in a portion of a network, it is recommended that all LSRs in that portion of the network be configured with the same Path Vector limit. Although knowledge of



a peer's Path Vector limit will not change an LSR's behavior, it does enable the LSR to alert an operator to a possible misconfiguration.

#### Reserved

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

#### Max PDU Length

Two octet unsigned integer that proposes the maximum allowable length for LDP PDUs for the session. A value of 255 or less specifies the default maximum length of 4096 octets.

The receiving LSR MUST calculate the maximum PDU length for the session by using the smaller of its and its peer's proposals for Max PDU Length. The default maximum PDU length applies before session initialization completes. If the maximum PDU length determined this way is unacceptable to an LSR, it MUST send a Session Rejected/Parameters Max PDU Length Notification message in response to the Initialization message and not establish the session.

#### Receiver LDP Identifier

Identifies the receiver's label space. This LDP Identifier, together with the sender's LDP Identifier in the PDU header, enables the receiver to match the Initialization message with one of its Hello adjacencies; see Section 4.5.2.1 "Hello Message Procedures".

If there is no matching Hello adjacency, the LSR MUST send a Session Rejected/No Hello Notification message in response to the Initialization message and not establish the session.

#### Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. The optional parameters are:

Optional Parameters	Type	Length	Value
ATM Session Parameters	0x0501	var	See below
Frame Relay Session Parameters	0x0502	var	See below

Table 7: Initialization Message; Optional Parameters

## ATM Session Parameters

Used when an LDP session manages label exchange for an ATM link to specify ATM-specific session parameters.

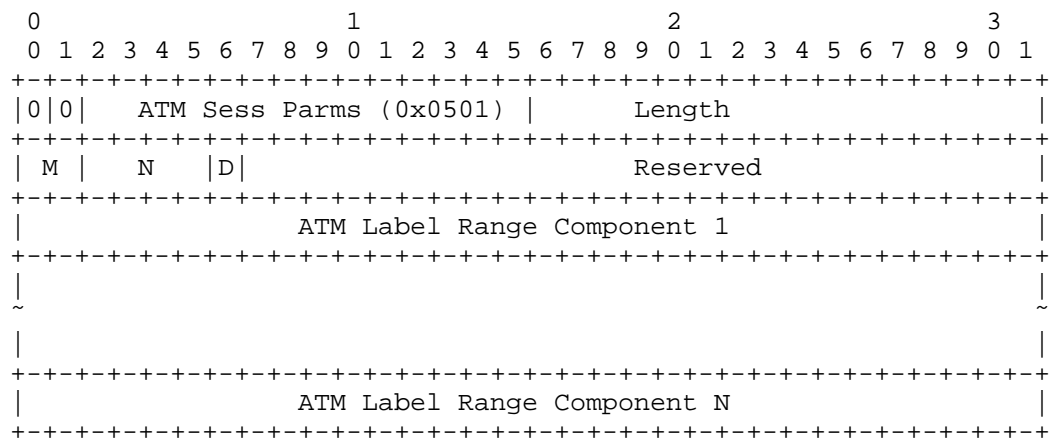


Figure 23: ATM Session Parameters (optional)

## M, ATM Merge Capabilities

Specifies the merge capabilities of an ATM switch. The following values are supported in this version of the specification:

Value	Meaning
0	Merge not supported
1	VP Merge supported
2	VC Merge supported
3	VP & VC Merge supported

Table 8: ATM Merge Capabilities

If the merge capabilities of the LSRs differ, then:

- Non-merge and VC-merge LSRs may freely interoperate.
- The interoperability of VP-merge-capable switches with non- VP-merge-capable switches is a subject for future study. When the LSRs differ on the use of VP merge, the session is established, but VP merge is not used.

Note that if VP merge is used, it is the responsibility of the ingress node to ensure that the chosen VCI is unique within the LSR domain.

N, Number of label range components

Specifies the number of ATM Label Range Components included in the TLV.

D, VC Directionality

A value of 0 specifies bidirectional VC capability, meaning the LSR can (within a given VPI) support the use of a given VCI as a label for both link directions independently. A value of 1 specifies unidirectional VC capability, meaning (within a given VPI) a given VCI may appear in a label mapping for one direction on the link only. When either or both of the peers specifies unidirectional VC capability, both LSRs use unidirectional VC label assignment for the link as follows. The LSRs compare their LDP Identifiers as unsigned integers. The LSR with the larger LDP Identifier may assign only odd- numbered VCIs in the VPI/VCI range as labels. The system with the smaller LDP Identifier may assign only even-numbered VCIs in the VPI/VCI range as labels.

Reserved

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

One or more ATM Label Range Components

A list of ATM Label Range Components that together specify the Label range supported by the transmitting LSR.

A receiving LSR MUST calculate the intersection between the received range and its own supported label range. The intersection is the range in which the LSR may allocate and accept labels. LSRs MUST NOT establish a session with neighbors for which the intersection of ranges is NULL. In this case, the LSR

MUST send a Session Rejected/Parameters Label Range Notification message in response to the Initialization message and not establish the session.

The encoding for an ATM Label Range Component is:

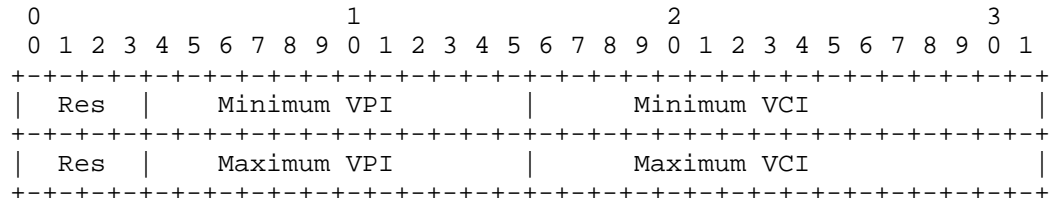


Figure 24: ATM Label Range Component

#### Res

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

#### Minimum VPI (12 bits)

This 12-bit field specifies the lower bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12 bits, it SHOULD be right justified in this field and preceding bits SHOULD be set to 0.

#### Minimum VCI (16 bits)

This 16-bit field specifies the lower bound of a block of Virtual Channel Identifiers that is supported on the originating switch. If the VCI is less than 16 bits, it SHOULD be right justified in this field and preceding bits SHOULD be set to 0.

#### Maximum VPI (12 bits)

This 12-bit field specifies the upper bound of a block of Virtual Path Identifiers that is supported on the originating switch. If the VPI is less than 12 bits, it SHOULD be right justified in this field and preceding bits SHOULD be set to 0.

#### Maximum VCI (16 bits)

This 16-bit field specifies the upper bound of a block of Virtual Connection Identifiers that is supported on the originating switch. If the VCI is less than 16 bits, it SHOULD be right justified in this field and preceding bits SHOULD be set to 0.

When peer LSRs are connected indirectly by means of an ATM VP, the sending LSR SHOULD set the Minimum and Maximum VPI fields to 0, and the receiving LSR MUST ignore the Minimum and Maximum VPI fields.

#### Frame Relay Session Parameters

Used when an LDP session manages label exchange for a Frame Relay link to specify Frame Relay-specific session parameters.

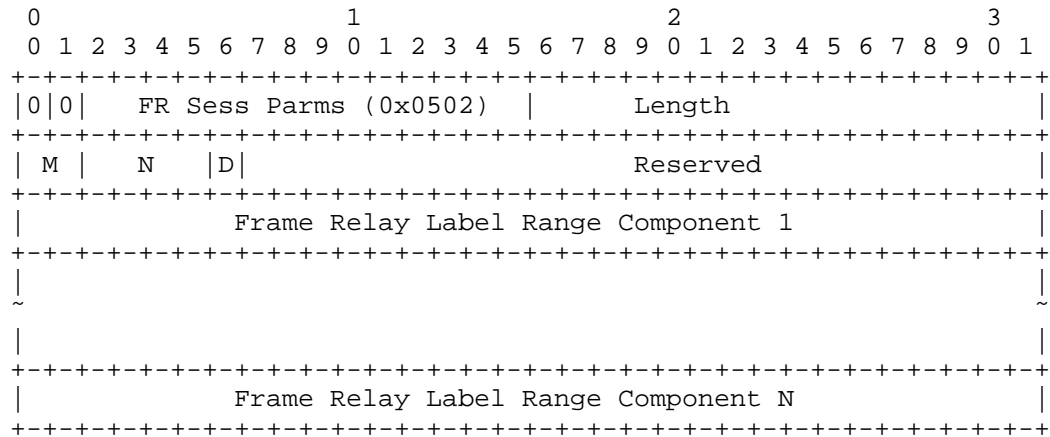


Figure 25: Frame Relay Session Parameters

#### M, Frame Relay Merge Capabilities

Specifies the merge capabilities of a Frame Relay switch. The following values are supported in this version of the specification:

Value	Meaning
0	Merge not supported
1	VP Merge supported
2	Merge supported

Table 9: Frame Relay Merge Capabilities

Non-merge and merge Frame Relay LSRs may freely interoperate.

#### N, Number of label range components

Specifies the number of Frame Relay Label Range Components included in the TLV.

#### D, VC Directionality

A value of 0 specifies bidirectional VC capability, meaning the LSR can support the use of a given DLCI as a label for both link directions independently. A value of 1 specifies unidirectional VC capability, meaning a given DLCI may appear in a label mapping for one direction on the link only. When either or both of the peers specifies unidirectional VC capability, both LSRs use unidirectional VC label assignment for the link as follows. The LSRs compare their LDP Identifiers as unsigned integers. The LSR with the larger LDP Identifier may assign only odd-numbered DLCIs in the range as labels. The system with the smaller LDP Identifier may assign only even-numbered DLCIs in the range as labels.

#### Reserved

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

#### One or more Frame Relay Label Range Components

A list of Frame Relay Label Range Components that together specify the Label range supported by the transmitting LSR.

A receiving LSR MUST calculate the intersection between the received range and its own supported label range. The intersection is the range in which the LSR may allocate and accept labels. LSRs MUST NOT establish a session with neighbors for which the intersection of ranges is NULL. In this case, the LSR MUST send a Session Rejected/Parameters Label Range Notification message in response to the Initialization message and not establish the session.

The encoding for a Frame Relay Label Range Component is:

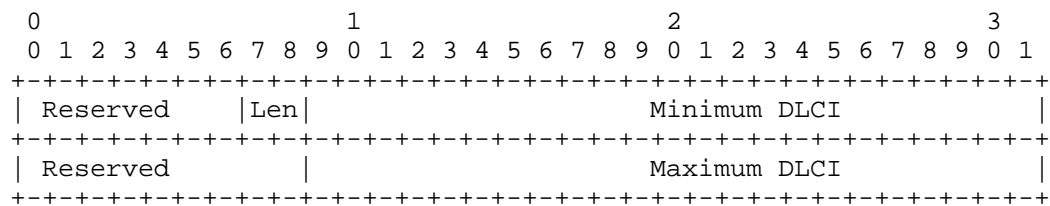


Figure 26: Frame Relay Label Range Component

**Reserved**

This field is reserved. It MUST be set to zero on transmission and ignored on receipt.

**Len**

This field specifies the number of bits of the DLCI. The following values are supported:

Len	DLCI bits
0	10
1	reserved
2	23
3	reserved

Table 10: Number of DLCI bits

**Minimum DLCI**

This 23-bit field specifies the lower bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI SHOULD be right justified in this field and unused bits SHOULD be set to 0.

**Maximum DLCI**

This 23-bit field specifies the upper bound of a block of Data Link Connection Identifiers (DLCIs) that is supported on the originating switch. The DLCI SHOULD be right justified in this field and unused bits SHOULD be set to 0.

Note that there is no Generic Session Parameters TLV for sessions that advertise Generic Labels.

**4.5.3.1. Initialization Message Procedures**

See Section 3.5.1 "LDP Session Establishment" and particularly Section 3.5.3 "Session Initialization" for general procedures for handling the Initialization message.

#### 4.5.4. KeepAlive Message

An LSR sends KeepAlive messages as part of a mechanism that monitors the integrity of the LDP session transport connection.

The encoding for the KeepAlive message is:

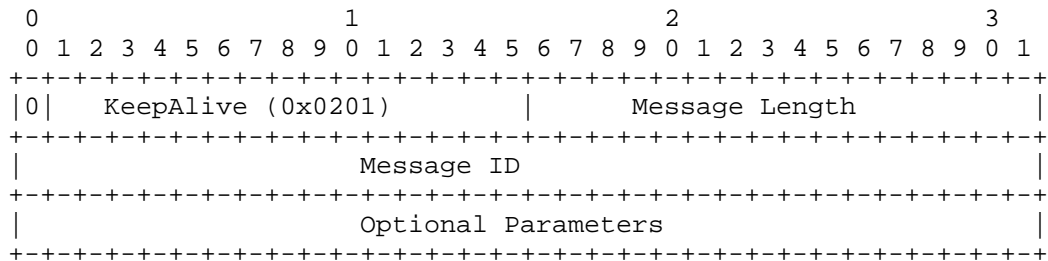


Figure 27: KeepAlive Message

Message ID

32-bit value used to identify this message.

Optional Parameters

No optional parameters are defined for the KeepAlive message.

##### 4.5.4.1. KeepAlive Message Procedures

The KeepAlive Timer mechanism described in Section 3.5.6 "Maintaining LDP Sessions" resets a session KeepAlive Timer every time an LDP PDU is received on the session TCP connection. The KeepAlive message is provided to allow reset of the KeepAlive Timer in circumstances where an LSR has no other information to communicate to an LDP peer.

An LSR MUST arrange that its peer receive an LDP message from it at least every KeepAlive Time period. Any LDP protocol message will do but, in circumstances where no other LDP protocol messages have been sent within the period, a KeepAlive message MUST be sent.

#### 4.5.5. Address Message

An LSR sends the Address message to an LDP peer to advertise its interface addresses.

The encoding for the Address message is:



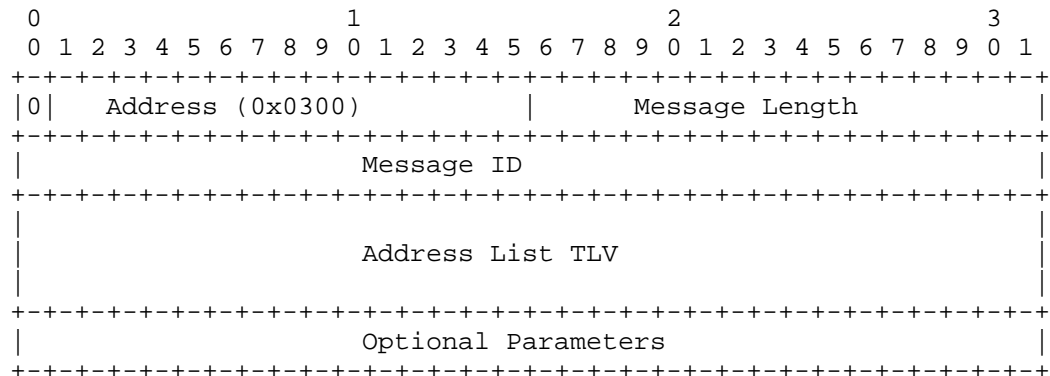


Figure 28: Address Message

**Message ID**

32-bit value used to identify this message.

**Address List TLV**

The list of interface addresses being advertised by the sending LSR. The encoding for the Address List TLV is specified in Section 4.4.3 "Address List TLV".

**Optional Parameters**

No optional parameters are defined for the Address message.

**4.5.5.1. Address Message Procedures**

An LSR that receives an Address message uses the addresses it learns to maintain a database for mapping between peer LDP Identifiers and next hop addresses; see Section 3.7 LDP Identifiers and Next Hop Addresses".

When a new LDP session is initialized and before sending Label Mapping or Label Request messages, an LSR SHOULD advertise its interface addresses with one or more Address messages.

Whenever an LSR "activates" a new interface address, it SHOULD advertise the new address with an Address message.

Whenever an LSR "de-activates" a previously advertised address, it SHOULD withdraw the address with an Address Withdraw message; see Section "Address Withdraw Message".

If an LSR does not support the Address Family specified in the Address List TLV, it SHOULD send an Unsupported Address Family Notification to its LDP signaling an error and abort processing the message.

An LSR may re-advertise an address (A) that it has previously advertised without explicitly withdrawing the address. If the receiver already has address binding (LSR, A), it SHOULD take no further action.

An LSR may withdraw an address (A) without having previously advertised it. If the receiver has no address binding (LSR, A), it SHOULD take no further action.

#### 4.5.6. Address Withdraw Message

An LSR sends the Address Withdraw message to an LDP peer to withdraw previously advertised interface addresses.

The encoding for the Address Withdraw message is:

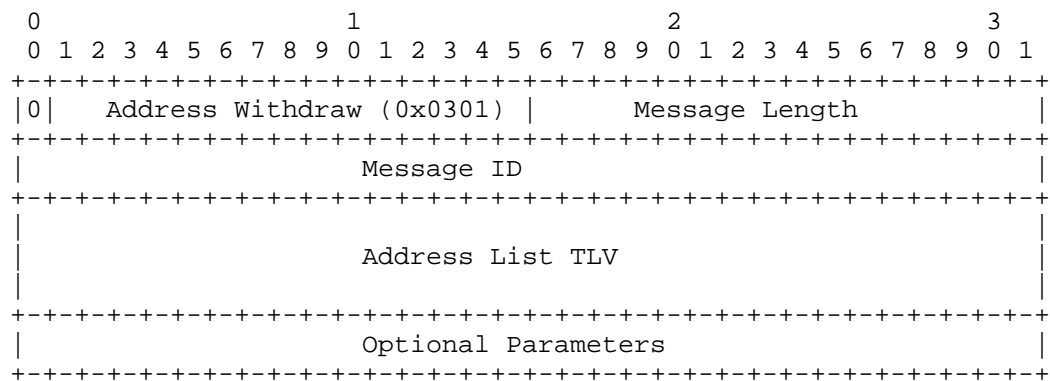


Figure 29: Address Withdraw Message

##### Message ID

32-bit value used to identify this message.

##### Address List TLV

The list of interface addresses being withdrawn by the sending LSR. The encoding for the Address List TLV is specified in Section "Address List TLV".

##### Optional Parameters

No optional parameters are defined for the Address Withdraw message.

#### 4.5.6.1. Address Withdraw Message Procedures

See Section 4.5.5.1 "Address Message Procedures".

#### 4.5.7. Label Mapping Message

An LSR sends a Label Mapping message to an LDP peer to advertise FEC-label bindings to the peer.

The encoding for the Label Mapping message is:

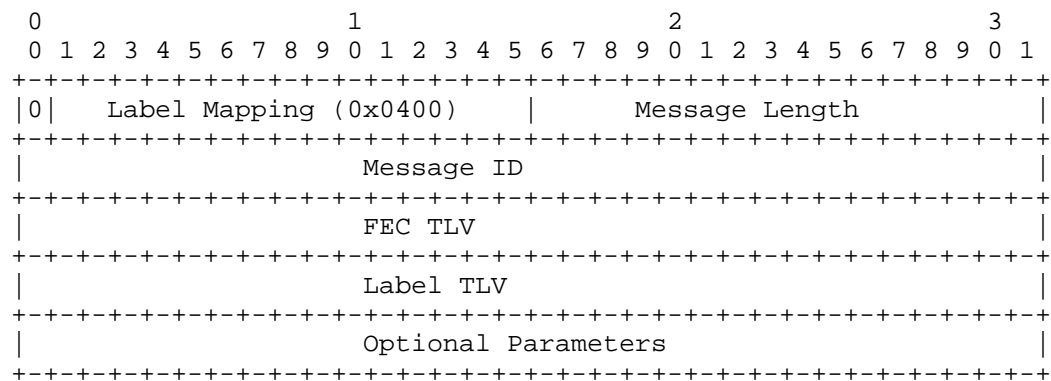


Figure 30: Label Mapping Message

##### Message ID

32-bit value used to identify this message.

##### FEC TLV

Specifies the FEC component of the FEC-Label mapping being advertised. See Section 4.4.1 "FEC TLVs" for encoding.

##### Label TLV

Specifies the Label component of the FEC-Label mapping. See Section 4.4.2 "Label TLVs" for encoding.

##### Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. The optional parameters are:

Optional Parameter	Length	Value
Label Request Message ID TLV	4	See below
Hop Count TLV	1	See below
Path Vector TLV	variable	See below

Table 11: Label Mapping Message: Optional Parameters

The encodings for the Hop Count and Path Vector TLVs can be found in Section 4.4 "TLV Encodings for Commonly Used Parameters".

#### Label Request Message ID

If this Label Mapping message is a response to a Label Request message, it **MUST** include the Label Request Message ID optional parameter. The value of this optional parameter is the Message ID of the corresponding Label Request message.

#### Hop Count

Specifies the running total of the number of LSR hops along the LSP being set up by the Label message. Section "Hop Count Procedures" describes how to handle this TLV.

#### Path Vector

Specifies the LSRs along the LSP being set up by the Label message. Section 4.4.5.1 "Path Vector Procedures" describes how to handle this TLV.

#### 4.5.7.1. Label Mapping Message Procedures

The Mapping message is used by an LSR to distribute a label mapping for a FEC to an LDP peer. If an LSR distributes a mapping for a FEC to multiple LDP peers, it is a local matter whether it maps a single label to the FEC, and distributes that mapping to all its peers, or whether it uses a different mapping for each of its peers.

An LSR is responsible for the consistency of the label mappings it has distributed and that its peers have these mappings.

An LSR receiving a Label Mapping message from a downstream LSR for a Prefix **SHOULD NOT** use the label for forwarding unless its routing table contains an entry that exactly matches the FEC Element.

See Appendix A, "LDP Label Distribution Procedures", for more details.

#### 4.5.7.1.1. Independent Control Mapping

If an LSR is configured for independent control, a mapping message is transmitted by the LSR upon any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table, and the label advertisement mode is Downstream Unsolicited advertisement.
2. The LSR receives a Request message from an upstream peer for a FEC present in the LSR's forwarding table.
3. The next hop for a FEC changes to another LDP peer, and Loop detection is configured.
4. The attributes of a mapping change.
5. The receipt of a mapping from the downstream next hop AND
  - a.) no upstream mapping has been created OR
  - b.) loop detection is configured OR
  - c.) the attributes of the mapping have changed.

#### 4.5.7.1.2. Ordered Control Mapping

If an LSR is doing Ordered Control, a Mapping message is transmitted by downstream LSRs upon any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table and is the egress for that FEC.
2. The LSR receives a Request message from an upstream peer for a FEC present in the LSR's forwarding table, and the LSR is the egress for that FEC OR has a downstream mapping for that FEC.
3. The next hop for a FEC changes to another LDP peer, and Loop Detection is configured.
4. The attributes of a mapping change.
5. The receipt of a mapping from the downstream next hop AND
  - a.) no upstream mapping has been created OR

- b.) loop detection is configured OR
- c.) the attributes of the mapping have changed.

#### 4.5.7.1.3. Downstream on Demand Label Advertisement

In general, the upstream LSR is responsible for requesting label mappings when operating in Downstream on Demand mode. However, unless some rules are followed, it is possible for neighboring LSRs with different advertisement modes to get into a livelock situation where everything is functioning properly, but no labels are distributed. For example, consider two LSRs Ru and Rd where Ru is the upstream LSR and Rd is the downstream LSR for a particular FEC. In this example, Ru is using Downstream Unsolicited advertisement mode and Rd is using Downstream on Demand mode. In this case, Rd may assume that Ru will request a label mapping when it wants one and Ru may assume that Rd will advertise a label if it wants Ru to use one. If Rd and Ru operate as suggested, no labels will be distributed from Rd to Ru.

This livelock situation can be avoided if the following rule is observed: an LSR operating in Downstream on Demand mode SHOULD NOT be expected to send unsolicited mapping advertisements. Therefore, if the downstream LSR is operating in Downstream on Demand mode, the upstream LSR is responsible for requesting label mappings as needed.

#### 4.5.7.1.4. Downstream Unsolicited Label Advertisement

In general, the downstream LSR is responsible for advertising a label mapping when it wants an upstream LSR to use the label. An upstream LSR may issue a mapping request if it so desires.

The combination of Downstream Unsolicited mode and Conservative Label retention can lead to a situation where an LSR releases the label for a FEC that it later needs. For example, if LSR Rd advertises to LSR Ru the label for a FEC for which it is not Ru's next hop, Ru will release the label. If Ru's next hop for the FEC later changes to Rd, it needs the previously released label.

To deal with this situation, either Ru can explicitly request the label when it needs it, or Rd can periodically re-advertise it to Ru. In many situations Ru will know when it needs the label from Rd. For example, when its next hop for the FEC changes to Rd. However, there could be situations when Ru does not. For example, Rd may be attempting to establish an LSP with non-standard properties. Forcing Ru to explicitly request the label in this situation would require it to maintain state about a potential LSP with non-standard properties.

In situations where Ru knows it needs the label, it is responsible for explicitly requesting the label by means of a Label Request message. In situations where Ru may not know that it needs the label, Rd is responsible for periodically re-advertising the label to Ru.

For this version of LDP, the only situation where Ru knows it needs a label for a FEC from Rd is when Rd is its next hop for the FEC, Ru does not have a label from Rd, and the LSP for the FEC is one that can be established with TLVs defined in this document.

#### 4.5.8. Label Request Message

An LSR sends the Label Request message to an LDP peer to request a binding (mapping) for a FEC.

The encoding for the Label Request message is:

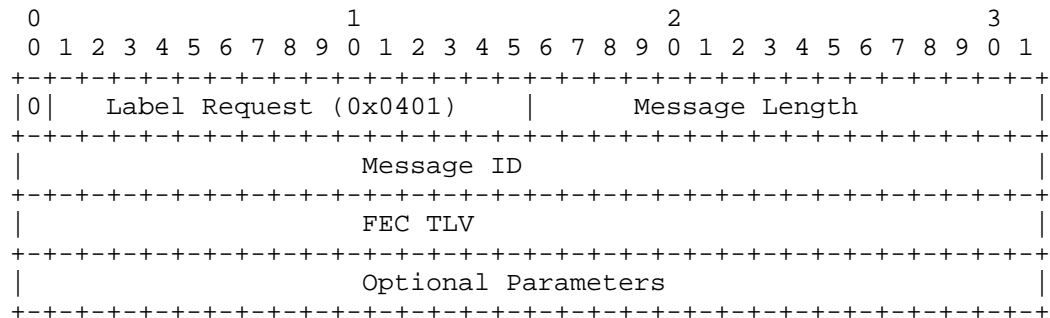


Figure 31: Label Request Message

Message ID

32-bit value used to identify this message.

## FEC TLV

The FEC for which a label is being requested. See Section 4.4.1 "FEC TLV" for encoding.

## Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. The optional parameters are:

Optional Parameter	Length	Value
Hop Count TLV	1	See below
Path Vector TLV	variable	See below

Table 12: Label Request Message: Optional Parameters

The encodings for the Hop Count and Path Vector TLVs can be found in Section "TLV Encodings for Commonly Used Parameters".

#### Hop Count

Specifies the running total of the number of LSR hops along the LSP being set up by the Label Request message. Section "Hop Count Procedures" describes how to handle this TLV.

#### Path Vector

Specifies the LSRs along the LSP being set up by the Label Request message. Section "Path Vector Procedures" describes how to handle this TLV.

#### 4.5.8.1. Label Request Message Procedures

The Request message is used by an upstream LSR to explicitly request that the downstream LSR assign and advertise a label for a FEC.

An LSR may transmit a Request message under any of the following conditions:

1. The LSR recognizes a new FEC via the forwarding table, and the next hop is an LDP peer, and the LSR doesn't already have a mapping from the next hop for the given FEC.
2. The next hop to the FEC changes, and the LSR doesn't already have a mapping from that next hop for the given FEC.

Note that if the LSR already has a pending Label Request message for the new next hop, it SHOULD NOT issue an additional Label Request in response to the next hop change.

3. The LSR receives a Label Request for a FEC from an upstream LDP peer, the FEC next hop is an LDP peer, and the LSR doesn't already have a mapping from the next hop.  
Note that since a non-merge LSR must set up a separate LSP for each upstream peer requesting a label, it must send a separate Label Request for each such peer. A consequence of this is that



a non-merge LSR may have multiple Label Request messages for a given FEC outstanding at the same time.

The receiving LSR SHOULD respond to a Label Request message with a Label Mapping for the requested label or with a Notification message indicating why it cannot satisfy the request.

When the FEC for which a label is requested is a Prefix FEC Element, the receiving LSR uses its routing table to determine its response. Unless its routing table includes an entry that exactly matches the requested Prefix, the LSR MUST respond with a No Route Notification message.

The message ID of the Label Request message serves as an identifier for the Label Request transaction. When the receiving LSR responds with a Label Mapping message, the mapping message MUST include a Label Request/Returned Message ID TLV optional parameter that includes the message ID of the Label Request message. Note that since LSRs use Label Request message IDs as transaction identifiers, an LSR SHOULD NOT reuse the message ID of a Label Request message until the corresponding transaction completes.

This version of the protocol defines the following Status Codes for the Notification message that signals a request cannot be satisfied:

**No Route**

The FEC for which a label was requested includes a FEC Element for which the LSR does not have a route.

**No Label Resources**

The LSR cannot provide a label because of resource limitations. When resources become available, the LSR MUST notify the requesting LSR by sending a Notification message with the Label Resources Available Status Code.

An LSR that receives a No Label Resources response to a Label Request message MUST NOT issue further Label Request messages until it receives a Notification message with the Label Resources Available Status Code.

**Loop Detected**

The LSR has detected a looping Label Request message.

See Appendix A, "LDP Label Distribution Procedures", for more details.

#### 4.5.9. Label Abort Request Message

The Label Abort Request message may be used to abort an outstanding Label Request message.

The encoding for the Label Abort Request message is:

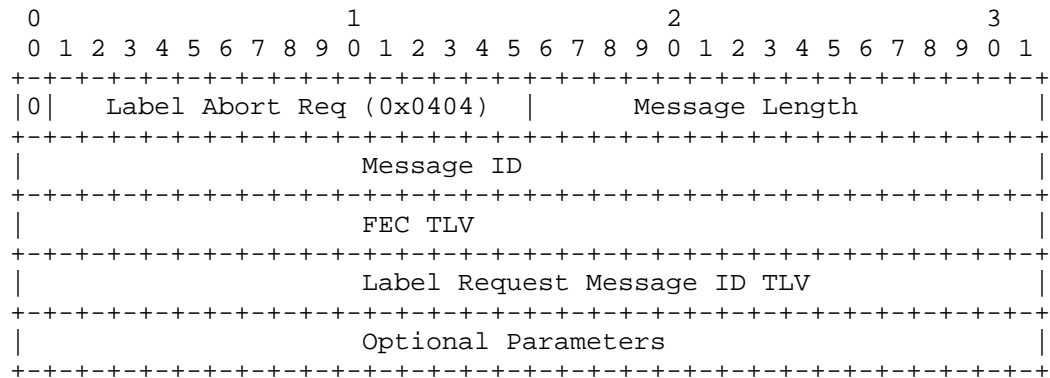


Figure 32: Label Abort Request Message

##### Message ID

32-bit value used to identify this message.

##### FEC TLV

Identifies the FEC for which the Label Request is being aborted.

##### Label Request Message ID TLV

Specifies the message ID of the Label Request message to be aborted.

##### Optional Parameters

No optional parameters are defined for the Label Abort Req message.

#### 4.5.9.1. Label Abort Request Message Procedures

An LSR Ru may send a Label Abort Request message to abort an outstanding Label Request message for a FEC sent to an LSR Rd in the following circumstances:

1. Ru's next hop for the FEC has changed from LSR Rd to LSR X; or
2. Ru is a non-merge, non-ingress LSR and has received a Label Abort Request for the FEC from an upstream peer Y.

3. Ru is a merge, non-ingress LSR and has received a Label Abort Request for the FEC from an upstream peer Y and Y is the only (last) upstream LSR requesting a label for the FEC.

There may be other situations where an LSR may choose to abort an outstanding Label Request message in order to reclaim resource associated with the pending LSP. However, specification of general strategies for using the abort mechanism is beyond the scope of LDP.

When an LSR receives a Label Abort Request message, if it has not previously responded to the Label Request being aborted with a Label Mapping message or some other Notification message, it MUST acknowledge the abort by responding with a Label Request Aborted Notification message. The Notification MUST include a Label Request Message ID TLV that carries the message ID of the aborted Label Request message.

If an LSR receives a Label Abort Request Message after it has responded to the Label Request in question with a Label Mapping message or a Notification message, it ignores the abort request.

If an LSR receives a Label Mapping message in response to a Label Request message after it has sent a Label Abort Request message to abort the Label Request, the label in the Label Mapping message is valid. The LSR may choose to use the label or to release it with a Label Release message.

An LSR aborting a Label Request message may not reuse the Message ID for the Label Request message until it receives one of the following from its peer:

- A Label Request Aborted Notification message acknowledging the abort;
- A Label Mapping message in response to the Label Request message being aborted;
- A Notification message in response to the Label Request message being aborted (e.g., Loop Detected, No Label Resources, etc.).

To protect itself against tardy peers or faulty peer implementations an LSR may choose to time out receipt of the above. The timeout period should be relatively long (several minutes). If the timeout period elapses with no reply from the peer, the LSR may reuse the Message ID of the Label Request message; if it does so, it should also discard any record of the outstanding Label Request and Label Abort messages.

Note that the response to a Label Abort Request message is never "ordered". That is, the response does not depend on the downstream state of the LSP setup being aborted. An LSR receiving a Label Abort Request message MUST process it immediately, regardless of the downstream state of the LSP, responding with a Label Request Aborted Notification or ignoring it, as appropriate.

#### 4.5.10. Label Withdraw Message

An LSR sends a Label Withdraw Message to an LDP peer to signal the peer that the peer may not continue to use specific FEC-label mappings the LSR had previously advertised. This breaks the mapping between the FECs and the labels.

The encoding for the Label Withdraw Message is:

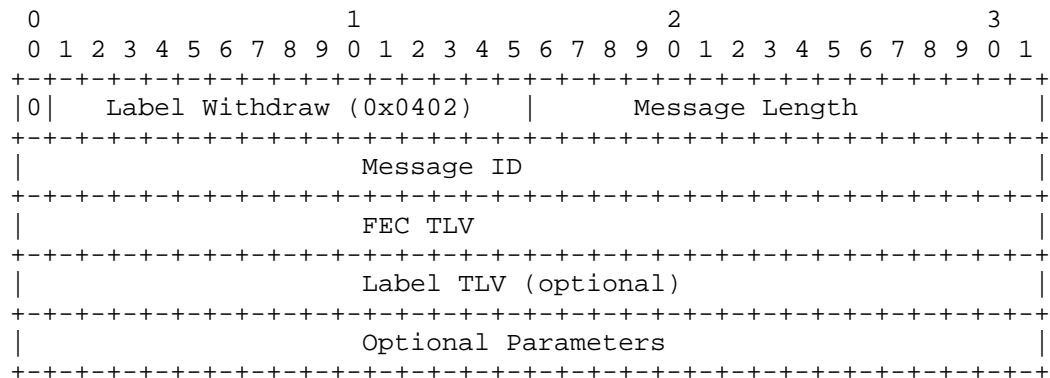


Figure 33: Label Withdraw Message

##### Message ID

32-bit value used to identify this message.

##### FEC TLV

Identifies the FEC for which the FEC-label mapping is being withdrawn.

##### Optional Parameters

This variable length field contains 0 or more parameters, each encoded as a TLV. The optional parameters are:

Optional Parameter	Length	Value
Label TLV	variable	See below

Table 13: Label Withdraw Message: Optional Parameters

The encoding for Label TLVs are found in Section 4.4.2 "Label TLVs"

#### Label

If present, specifies the label being withdrawn (see procedures below).

#### 4.5.10.1. Label Withdraw Message Procedures

An LSR transmits a Label Withdraw message under the following conditions:

1. The LSR no longer recognizes a previously known FEC for which it has advertised a label.
2. The LSR has decided unilaterally (e.g., via configuration) to no longer label switch a FEC (or FECs) with the label mapping being withdrawn.

The FEC TLV specifies the FEC for which labels are to be withdrawn. If no Label TLV follows the FEC, all labels associated with the FEC are to be withdrawn; otherwise, only the label specified in the optional Label TLV is to be withdrawn.

The FEC TLV may contain the Wildcard FEC Element; if so, it may contain no other FEC Elements. In this case, if the Label Withdraw message contains an optional Label TLV, then the label is to be withdrawn from all FECs to which it is bound. If there is not an optional Label TLV in the Label Withdraw message, then the sending LSR is withdrawing all label mappings previously advertised to the receiving LSR.

An LSR that receives a Label Withdraw message MUST respond with a Label Release message.

See Appendix A, "LDP Label Distribution Procedures", for more details.



#### 4.5.11.1. Label Release Message Procedures

An LSR transmits a Label Release message to a peer when it no longer needs a label previously received from or requested of that peer.

An LSR MUST transmit a Label Release message under any of the following conditions:

1. The LSR that sent the label mapping is no longer the next hop for the mapped FEC, and the LSR is configured for conservative operation.
2. The LSR receives a label mapping from an LSR that is not the next hop for the FEC, and the LSR is configured for conservative operation.
3. The LSR receives a Label Withdraw message.

Note that if an LSR is configured for "liberal mode", a release message will never be transmitted in the case of conditions (1) and (2) as specified above. In this case, the upstream LSR keeps each unused label, so that it can immediately be used later if the downstream peer becomes the next hop for the FEC.

The FEC TLV specifies the FEC for which labels are to be released. If no Label TLV follows the FEC, all labels associated with the FEC are to be released; otherwise, only the label specified in the optional Label TLV is to be released.

The FEC TLV may contain the Wildcard FEC Element; if so, it may contain no other FEC Elements. In this case, if the Label Release message contains an optional Label TLV, then the label is to be released for all FECs to which it is bound. If there is not an optional Label TLV in the Label Release message, then the sending LSR is releasing all label mappings previously learned from the receiving LSR.

See Appendix A, "LDP Label Distribution Procedures", for more details.

#### 4.6. Messages and TLVs for Extensibility

Support for LDP extensibility includes the rules for the U- and F-bits that specify how an LSR handles unknown TLVs and messages.

This section specifies TLVs and messages for vendor-private and experimental use.

#### 4.6.1. LDP Vendor-Private Extensions

Vendor-private TLVs and messages are used to convey vendor-private information between LSRs.

#### 4.6.1.1. LDP Vendor-Private TLVs

The Type range 0x3E00 through 0x3EFF is reserved for vendor-private TLVs.

The encoding for a vendor-private TLV is:

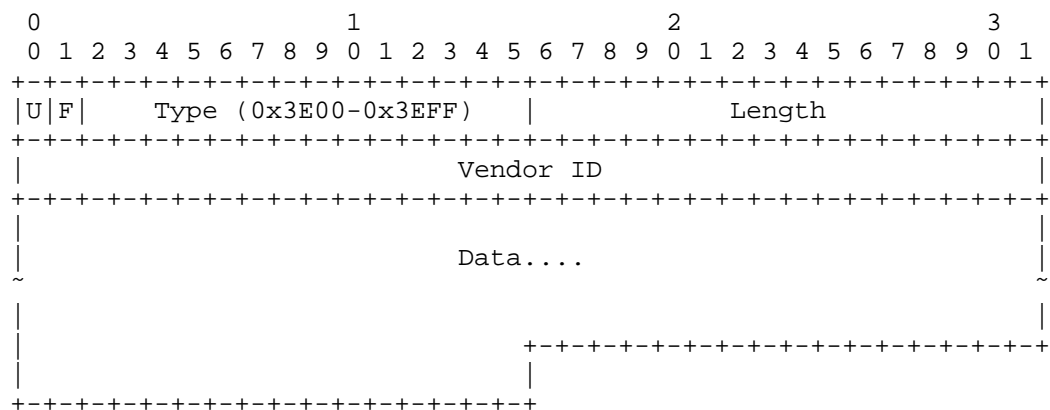


Figure 35: LDP Vendor-Private TLVs

## U-bit

Unknown TLV bit. Upon receipt of an unknown TLV, if U is clear (=0), a notification **MUST** be returned to the message originator and the entire message **MUST** be ignored; if U is set (=1), the unknown TLV is silently ignored and the rest of the message is processed as if the unknown TLV did not exist.

The determination as to whether a vendor-private message is understood is based on the Type and the mandatory Vendor ID field.

Implementations that support vendor-private TLVs MUST support a user-accessible configuration interface that causes the U-bit to be set on all transmitted vendor-private TLVs; this requirement MAY be satisfied by a user-accessible configuration interface that prevents transmission of all vendor-private TLVs for which the U-bit is clear.

## F-bit



Forward unknown TLV bit. This bit only applies when the U-bit is set and the LDP message containing the unknown TLV is to be forwarded. If F is clear (=0), the unknown TLV is not forwarded with the containing message; if F is set (=1), the unknown TLV is forwarded with the containing message.

Type

Type value in the range 0x3E00 through 0x3EFF. Together, the Type and Vendor ID field specify how the Data field is to be interpreted.

Length

Specifies the cumulative length in octets of the Vendor ID and Data fields.

Vendor ID

802 Vendor ID as assigned by the IEEE.

Data

The remaining octets after the Vendor ID in the Value field are optional vendor-dependent data.

#### 4.6.1.2. LDP Vendor-Private Messages

The Message Type range 0x3E00 through 0x3EFF is reserved for Vendor-Private messages.

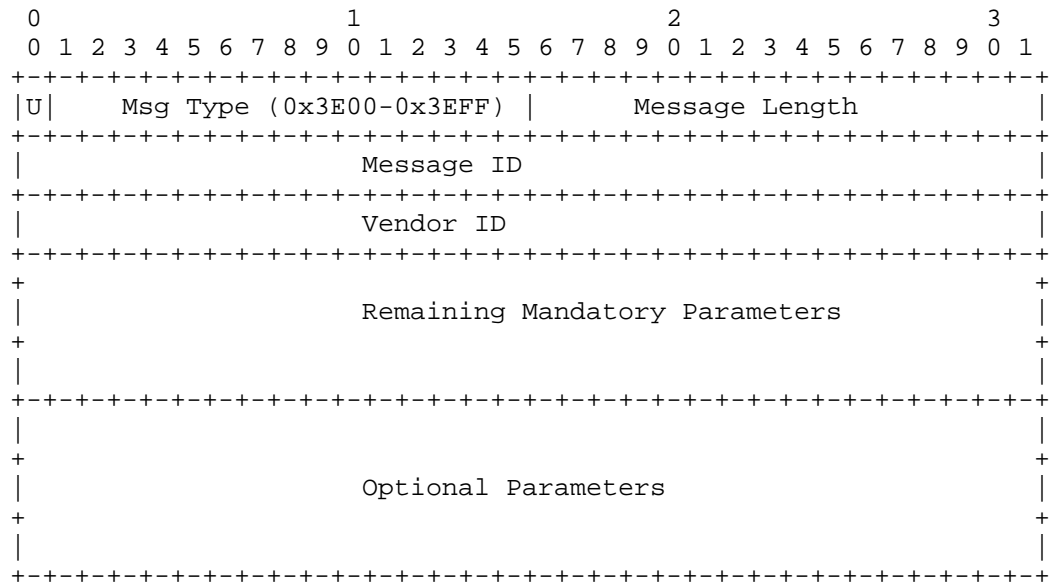


Figure 36: LDP Vendor-Private Messages

**U-bit**

Unknown message bit. Upon receipt of an unknown message, if U is clear (=0), a notification is returned to the message originator; if U is set (=1), the unknown message is silently ignored.

The determination as to whether a Vendor-Private message is understood is based on the Msg Type and the Vendor ID parameter.

Implementations that support Vendor-Private messages MUST support a user-accessible configuration interface that causes the U-bit to be set on all transmitted Vendor-Private messages; this requirement MAY be satisfied by a user-accessible configuration interface that prevents transmission of all Vendor-Private messages for which the U-bit is clear.

**Msg Type**

Message Type value in the range 0x3E00 through 0x3EFF. Together, the Msg Type and the Vendor ID specify how the message is to be interpreted.

**Message Length**

Specifies the cumulative length in octets of the Message ID, Vendor ID, Remaining Mandatory Parameters, and Optional Parameters.

**Message ID**

32-bit integer used to identify this message. Used by the sending LSR to facilitate identifying Notification messages that may apply to this message. An LSR sending a Notification message in response to this message will include this Message ID in the notification message; see Section "Notification Message".

**Vendor ID**

802 Vendor ID as assigned by the IEEE.

**Remaining Mandatory Parameters**

Variable length set of remaining required message parameters.

**Optional Parameters**

Variable length set of optional message parameters.

**4.6.2. LDP Experimental Extensions**

LDP support for experimentation is similar to support for vendor-private extensions with the following differences:

- The Type range 0x3F00 through 0x3FFF is reserved for experimental TLVs.
- The Message Type range 0x3F00 through 0x3FFF is reserved for experimental messages.
- The encodings for experimental TLVs and messages are similar to the vendor-private encodings with the following difference.

Experimental TLVs and messages use an Experiment ID field in place of a Vendor ID field. The Experiment ID field is used with the Type or Message Type field to specify the interpretation of the experimental TLV or Message.

Administration of Experiment IDs is the responsibility of the experimenters.

**4.7. Message Summary**

The following are the LDP messages defined in this version of the protocol.

Message Name	Type	Section Number	Section Title
Notification	0x0001	4.5.1	Notification Message
Hello	0x0100	4.5.2	Hello Message
Initialization	0x0200	4.5.3	Initialization Message
KeepAlive	0x0201	4.5.4	KeepAlive Message
Address	0x0300	4.5.5	Address Message
Address Withdraw	0x0301	4.5.6	Address Withdraw Message
Label Mapping	0x0400	4.5.7	Label Mapping Message
Label Request	0x0401	4.5.8	Label Request Message
Label Withdraw	0x0402	4.5.10	Label Withdraw Message
Label Release	0x0403	4.5.11	Label Release Message
Label Abort Request	0x0404	4.5.9	Label Abort Request Message
Vendor-Private	0x3E00-0x3EFF	4.6.1	LDP Vendor-Private Extensions
Experimental	0x3F00-0x3FFF	4.6.2	LDP Experimental Extensions

Table 15: Message Summary

#### 4.8. TLV Summary

The following are the TLVs defined in this version of the protocol.

Message Name	Type	Section Number	Section Title
FEC	0x0100	4.4.1	FEC TLV
Address List	0x0101	4.4.3	Address List TLV
Hop Count	0x0103	4.4.4	Hop Count TLV
Path Vector	0x0104	4.4.5	Path Vector TLV
Generic Label	0x0200	4.4.2.1	Generic Label TLV
ATM Label	0x0201	4.4.2.2	ATM Label TLV
Frame Relay	0x0202	4.4.2.3	Frame Relay Label TLV
Status	0x0300	4.4.6	Status TLV
Extended	0x0301	4.5.1	Notification Message
Status			
Returned PDU	0x0302	4.5.1	Notification Message
Returned	0x0303	4.5.1	Notification Message
Message			
Common Hello	0x0400	4.5.2	Hello Message
Parameters			
IPv4	0x0401	4.5.2	Hello Message
Transport			
Address			
Configuration	0x0402	4.5.2	Hello Message
Sequence			
Number			
IPv6	0x0403	4.5.2	Hello Message
Transport			
Address			
Common	0x0500	4.5.3	Initialization Message
Session			
Parameters			
ATM Session	0x0501	4.5.3	Initialization Message
Parameters			
Frame Relay	0x0502	4.5.3	Initialization Message
Session			
Parameters			
Label Request	0x0600	4.5.7	Label Mapping Message
Message ID			
Vendor-	0x3E00-	4.6.1	LDP Vendor-Private Extensions
Private	0x3EFF		
Experimental	0x3F00-	4.6.2	LDP Experimental Extensions
	0x3FFF		

Table 16: TLV Summary

## 4.9. Status Code Summary

The following are the Status Codes defined in this version of the protocol.

The "E" column is the required setting of the Status Code E-bit; the "Status Data" column is the value of the 30-bit Status Data field in the Status Code TLV. Note that the setting of the Status Code F-bit is at the discretion of the LSR originating the Status TLV.

Status Code	E	Status Data	Section Number
Success	0	0x00000000	4.4.6
Bad LDP Identifier	1	0x00000001	4.5.1.2
Bad Protocol Version	1	0x00000002	4.5.1.2
Bad PDU Length	1	0x00000003	4.5.1.2
Unknown Message Type	0	0x00000004	4.5.1.2
Bad Message Length	1	0x00000005	4.5.1.2
Unknown TLV	0	0x00000006	4.5.1.2
Bad TLV Length	1	0x00000007	Section 4.5.1.2
Malformed TLV Value	1	0x00000008	4.5.1.2
Hold Timer Expired	1	0x00000009	4.5.1.2
Shutdown	1	0x0000000A	4.5.1.2
Loop Detected	0	0x0000000B	3.8
Unknown FEC	0	0x0000000C	4.4.1.1
No Route	0	0x0000000D	4.5.8
No Label Resources	0	0x0000000E	4.5.8
Label Resources / Available	0	0x0000000F	4.5.8
Session Rejected / No Hello	1	0x00000010	3.5.3
Session Rejected / Parameters	1	0x00000011	3.5.3
Advertisement Mode			
Session Rejected / Parameters	1	0x00000012	3.5.3
Max PDU Length			
Session Rejected / Parameters	1	0x00000013	Section 3.5.3
Label Range			
KeepAlive Timer Expired	1	0x00000014	3.5.3
Label Request Aborted	0	0x00000015	4.5.9
Missing Message Parameters	0	0x00000016	4.5.1.2
Unsupported Address Family	0	0x00000017	4.4.1.1
Session Rejected / Bad Keep	0	0x00000018	3.5.3
Alive Time			
Internal Error	0	0x00000019	4.5.1.2

Table 17: Status Code Summary

#### 4.10. Well-Known Numbers

prov text

##### 4.10.1. UDP and TCP Ports

The UDP port for LDP Hello messages is 646.

The TCP port for establishing LDP session connections is 646.

##### 4.10.2. Implicit NULL Label

The Implicit NULL label is defined in RFC 3031 [RFC3031] as follows:

"The Implicit NULL label is a label with special semantics which an LSR can bind to an address prefix. If LSR Ru, by consulting its ILM (Incoming Label Map) sees that labeled packet P must be forwarded next to Rd, but that Rd has distributed a binding of Implicit NULL to the corresponding address prefix, then instead of replacing the value of the label on top of the label stack, Ru pops the label stack, and then forwards the resulting packet to Rd."

The implicit NULL label is represented in LDP as a Generic Label TLV with a Label field value of 3, as defined in RFC 3032 [RFC3032].

#### 5. RFC 5036 IANA Considerations

Note: In version -00 of this document does only minimal changes to the RFC 5036 IANA considerationns. The author believe that some further minor changes will be made eventually. The "IANA consideration" section (see Section 9) is included to capture anything new that relates to IANA, Before publication the two section will be merged.

The LDP specification defines the following name spaces that are managed by IANA and found at [LDP\_NAME\_SPACE]:

- Message Type Name Space, found at [MSG\_TYPE\_NAME\_SPACE]
- TLV Type Name Space, found at [TLV\_TYPE\_NAME\_SPACE]
- FEC Type Name Space, found at [FEC\_TYPE\_NAME\_SPACE]
- Status Code Name Space, found at [STATUS\_CODE\_NAME\_SPACE]
- Experiment ID Name Space, found at [EXP\_ID\_NAME\_SPACE]

Section 5.1 "Message Type Name Space" to Section 5.5

"Experiment ID Name Space" provide guidelines for managing these name spaces.

LDP Name Spaces have also been defined by other RFCs since the LDP Specification was first published and are managed by IANA as addition to the original LDP Name Space at [LDP\_NAME\_SPACE].

RFC 6388 [RFC6388] defined:

LDP MP Opaque Value Element basic type, found at  
[MP\_BASIC\_OPAQUE]

LDP MP Opaque Value Element extended type, found at  
[EXT\_BASIC\_OPAQUE]

LDP MP Status Value Element type, found at [MP\_STATUS\_VALUE]

RFC 7361 [RFC7361]defined

MAC Flush Flags, found at [MAC\_FLUSH]

The guidelines for how to manage the name spaces defined in other RFCs than RFC 5036 (or this document when it gets approved) is found in the RFCs that defined the name spaces.

#### 5.1. Message Type Name Space

LDP divides the name space for message types into three ranges. The following are the guidelines for managing these ranges:

- Message Types 0x0000 - 0x3DFF. Message types in this range are part of the LDP base protocol. Following the policies outlined in RFC 5226 [RFC5226] and RFC 2434 [RFC2434], Message types in this range are allocated through an IETF Consensus action.
- Message Types 0x3E00 - 0x3EFF. Message types in this range are reserved for Vendor-Private extensions and are the responsibility of the individual vendors (see Section "LDP Vendor-Private Messages"). IANA management of this range of the Message Type Name Space is unnecessary.
- Message Types 0x3F00 - 0x3FFF. Message types in this range are reserved for Experimental extensions and are the responsibility of the individual experimenters (see Sections "LDP Experimental Extensions" and "Experiment ID Name Space"). IANA management of this range of the Message Type Name Space is unnecessary; however,



IANA is responsible for managing part of the Experiment ID Name Space (see below).

#### 5.2. TLV Type Name Space

LDP divides the name space for TLV types into three ranges. The following are the guidelines for managing these ranges:

- TLV Types 0x0000 - 0x3DFF. TLV types in this range are part of the LDP base protocol. Following the policies outlined in RFC 5226 [RFC5226] and RFC 2434 [RFC2434], TLV types in this range are allocated through an IETF Consensus action.
- TLV Types 0x3E00 - 0x3EFF. TLV types in this range are reserved for Vendor-Private extensions and are the responsibility of the individual vendors (see Section "LDP Vendor-Private TLVs"). IANA management of this range of the TLV Type Name Space is unnecessary.
- TLV Types 0x3F00 - 0x3FFF. TLV types in this range are reserved for Experimental extensions and are the responsibility of the individual experimenters (see Sections "LDP Experimental Extensions" and "Experiment ID Name Space"). IANA management of this range of the TLV Name Space is unnecessary; however, IANA is responsible for managing part of the Experiment ID Name Space (see below).

#### 5.3. FEC Type Name Space

The range for FEC types is 0 - 255.

Following the policies outlined in RFC 5226 [RFC5226] and RFC 2434 [RFC2434], FEC types in the range 0 - 127 are allocated through an IETF Consensus action, types in the range 128 - 191 are allocated as First Come First Served, and types in the range 192 - 255 are reserved for Private Use.

#### 5.4. Status Code Name Space

The range for Status Codes is 0x00000000 - 0x3FFFFFFF.

Following the policies outlined in RFC 5226 [RFC5226] and RFC 2434 [RFC2434], Status Codes in the range 0x00000000 - 0x1FFFFFFF are allocated through an IETF Consensus action, codes in the range 0x20000000 - 0x3EFFFFFF are allocated as First Come First Served, and codes in the range 0x3F000000 - 0x3FFFFFFF are reserved for Private Use.

### 5.5. Experiment ID Name Space

The range for Experiment IDs is 0x00000000 - 0xffffffff.

Following the policies outlined in RFC 5226 [RFC5226] and RFC 2434 [RFC2434], Experiment IDs in the range 0x00000000 - 0xffffffff are allocated as First Come First Served and Experiment IDs in the range 0xf0000000 - 0xffffffff are reserved for Private Use.

## 6. Security Considerations

Editors Note: This is still working in progress.

This section identifies threats to which LDP may be vulnerable and discusses means by which those threats might be mitigated.

### 6.1. Spoofing

There are two types of LDP communication that could be the target of a spoofing attack.

#### 1. Discovery exchanges carried by UDP

LSRs indicate their willingness to establish and maintain LDP sessions by periodically sending Hello messages. Receipt of a Hello serves to create a new "Hello adjacency", if one does not already exist, or to refresh an existing one. Spoofing a Hello packet for an existing adjacency can cause the adjacency to time out and that can result in termination of the associated session. This can occur when the spoofed Hello specifies a small Hold Time, causing the receiver to expect Hellos within this interval, while the true neighbor continues sending Hellos at the lower, previously agreed to, frequency.

LSRs directly connected at the link level exchange Basic Hello messages over the link. The threat of spoofed Basic Hellos can be reduced by:

- o Accepting Basic Hellos only on interfaces to which LSRs that can be trusted are directly connected.
- o Ignoring Basic Hellos not addressed to the All Routers on this Subnet multicast group.

LSRs not directly connected at the link level may use Extended Hello messages to indicate willingness to establish an LDP session. An LSR can reduce the threat of spoofed Extended Hellos

by filtering them and accepting only those originating at sources permitted by an access list.

2. LSRs not directly connected at the link level may use Extended Hello messages to indicate willingness to establish an LDP session. An LSR can reduce the threat of spoofed Extended Hellos by filtering them and accepting only those originating at sources permitted by an access list.
3. Session communication carried by TCP

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.

## 6.2. Privacy

LDP provides no mechanism for protecting the privacy of label distribution.

The security requirements of label distribution protocols are essentially identical to those of the protocols that distribute routing information. By providing a mechanism to ensure the authenticity and integrity of its messages, LDP provides a level of security that is at least as good as, though no better than, that which can be provided by the routing protocols themselves. The more general issue of whether privacy should be required for routing protocols is beyond the scope of this document.

One might argue that label distribution requires privacy to address the threat of label spoofing. However, that privacy would not protect against label spoofing attacks since data packets carry labels in the clear. Furthermore, label spoofing attacks can be made without knowledge of the FEC bound to a label.

To avoid label spoofing attacks, it is necessary to ensure that labeled data packets are labeled by trusted LSRs and that the labels placed on the packets are properly learned by the labeling LSRs.

### 6.3. Denial of Service

LDP provides two potential targets for Denial of Service (DoS) attacks:

#### 1. Well-known UDP Port for LDP Discovery

An LSR administrator can address the threat of DoS attacks via Basic Hellos by ensuring that the LSR is directly connected only to peers that can be trusted to not initiate such an attack. Interfaces to peers interior to the administrator's domain should not represent a threat since interior peers are under the administrator's control. Interfaces to peers exterior to the domain represent a potential threat since exterior peers are not. An administrator can reduce that threat by connecting the LSR only to exterior peers that can be trusted to not initiate a Basic Hello attack.

DoS attacks via Extended Hellos are potentially a more serious threat. This threat can be addressed by filtering Extended Hellos using access lists that define addresses with which Extended Discovery is permitted. However, performing the filtering requires LSR resource.

In an environment where a trusted MPLS cloud can be identified, LSRs at the edge of the cloud can be used to protect interior LSRs against DoS attacks via Extended Hellos by filtering out Extended Hellos originating outside of the trusted MPLS cloud, accepting only those originating at addresses permitted by access lists. This filtering protects LSRs in the interior of the cloud but consumes resources at the edges.

#### 2. Well-known TCP port for LDP Session Establishment

Like other control plane protocols that use TCP, LDP may be the target of DoS attacks, such as SYN attacks. LDP is no more or less vulnerable to such attacks than other control plane protocols that use TCP.

The threat of such attacks can be mitigated somewhat by the following:

- o An LSR SHOULD avoid promiscuous TCP listens for LDP session establishment. It SHOULD use only listens that are specific to discovered peers. This enables it to drop attack packets early in their processing since they are less likely to match existing or in-progress connections.

- o The use of the MD5 option helps somewhat since it prevents a SYN from being accepted unless the MD5 segment checksum is valid. However, the receiver must compute the checksum before it can decide to discard an otherwise acceptable SYN segment.
- o The use of access list mechanisms applied at the boundary of the MPLS cloud in a manner similar to that suggested above for Extended Hellos can protect the interior against attacks originating from outside the cloud.

## 7. Areas for Future Study

The following topics not addressed in this version of LDP are possible areas for future study:

Note: in the -00 version of this document this section has not been changed from RFC 5036. It will need to be reviewed and updated.

- Section 2.16 of the MPLS architecture RFC 3031 [RFC3031] requires that the initial label distribution protocol negotiation between peer LSRs enable each LSR to determine whether its peer is capable of popping the label stack. This version of LDP assumes that LSRs support label popping for all link types except ATM and Frame Relay. A future version may specify means to make this determination part of the session initiation negotiation.
- LDP support for CoS (Class of Service) is not specified in this version. CoS support may be addressed in a future version.
- LDP support for multicast is not specified in this version. Multicast support may be addressed in a future version.
- LDP support for multipath label switching is not specified in this version. Multipath support may be addressed in a future version.
- LDP support for signaling the maximum transmission unit is not specified in this version. It is discussed in the experimental document RFC 3988 [RFC3988].
- The current specification does not address basic peer discovery on Non-Broadcast Multi-Access (NBMA) media. The solution available in the current specification is to use extended peer discovery in such setups. The issue of defining a mechanism semantically similar to Basic Discovery (1 hop limit, bind the hello adjacency to an interface) that uses preconfigured neighbor addresses is left for further study.

- The current specification does not support shutting down an adjacency. The motivation for doing it and the mechanisms for achieving it are left for further study.
- The current specification does not include a method for securing Hello messages, to detect spoofing of Hellos. The scenarios where this is necessary, as well as the mechanism for achieving it are left for future study.
- The current specification does not have the ability to detect a stateless fast control plane restart. The method for achieving this, possibly through an "incarnation/instance" number carried in the Hello message, is left for future study.
- The current specification does not support an "end of LIB" message, analogous to BGP's "end of RIB" message that an LDP LSR (operating in DU mode) would use following session establishment. The discussion on the need for such a mechanism and its implementation is left for future study.
- The current specification does not deal with situations where different LSRs advertise the same address. Such situations typically occur as the result of configuration errors, and the goal in this case is to provide the LSRs advertising the same address with enough information to enable operators to take corrective action. The specification of this mechanism is left for a separate document.

## 8. Changes from RFC 5036

Here is a list of changes from RFC 5036

1. Some editorial changes has been made, e.g. internal references is more frequently used, some implicit lists has been replaced by tables, e.g. for Optional Parameters carried in LDP messages.
2. The refrence to CR-LDP has been removed.
3. References to the LDP registries create outside the LDP Specification has been added.

## 9. IANA Considerations

There are no requests for IANA actions in this document.

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

## 10. References

### 10.1. Normative References

- [ASSIGNED\_AF] "IANA Assigned Address Families",  
<<http://www.iana.org/assignments/address-family-numbers>>.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<http://www.rfc-editor.org/info/rfc1321>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <<http://www.rfc-editor.org/info/rfc2328>>.
- [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <<http://www.rfc-editor.org/info/rfc2385>>.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, DOI 10.17487/RFC2434, October 1998, <<http://www.rfc-editor.org/info/rfc2434>>.
- [RFC2702] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, DOI 10.17487/RFC2702, September 1999, <<http://www.rfc-editor.org/info/rfc2702>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<http://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, <<http://www.rfc-editor.org/info/rfc3032>>.

- [RFC3034] Conta, A., Doolan, P., and A. Malis, "Use of Label Switching on Frame Relay Networks Specification", RFC 3034, DOI 10.17487/RFC3034, January 2001, <<http://www.rfc-editor.org/info/rfc3034>>.
- [RFC3035] Davie, B., Lawrence, J., McCloghrie, K., Rosen, E., Swallow, G., Rekhter, Y., and P. Doolan, "MPLS using LDP and ATM VC Switching", RFC 3035, DOI 10.17487/RFC3035, January 2001, <<http://www.rfc-editor.org/info/rfc3035>>.
- [RFC3037] Thomas, B. and E. Gray, "LDP Applicability", RFC 3037, DOI 10.17487/RFC3037, January 2001, <<http://www.rfc-editor.org/info/rfc3037>>.
- [RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <<http://www.rfc-editor.org/info/rfc3988>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5860] Vigoureux, M., Ed., Ward, D., Ed., and M. Betts, Ed., "Requirements for Operations, Administration, and Maintenance (OAM) in MPLS Transport Networks", RFC 5860, DOI 10.17487/RFC5860, May 2010, <<http://www.rfc-editor.org/info/rfc5860>>.
- [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, <<http://www.rfc-editor.org/info/rfc5921>>.
- [RFC6371] Busi, I., Ed. and D. Allan, Ed., "Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks", RFC 6371, DOI 10.17487/RFC6371, September 2011, <<http://www.rfc-editor.org/info/rfc6371>>.
- [RFC6372] Sprecher, N., Ed. and A. Farrel, Ed., "MPLS Transport Profile (MPLS-TP) Survivability Framework", RFC 6372, DOI 10.17487/RFC6372, September 2011, <<http://www.rfc-editor.org/info/rfc6372>>.



## 10.2. Informative References

- [EXP\_ID\_NAME\_SPACE]  
"Experiment ID Name Space",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-10>>.
- [EXT\_BASIC\_OPAQUE]  
"LDP MP Opaque Value Element extended type",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-13>>.
- [FEC\_TYPE\_NAME\_SPACE]  
"Forwarding Equivalence Class (FEC) Type Name Space",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#fec-type>>.
- [LDP\_NAME\_SPACE]  
"Label Distribution Protocol (LDP) Parameters",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml>>.
- [MAC\_FLUSH]  
"MAC Flush Flags", <<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#mac-flush-flags>>.
- [MP\_BASIC\_OPAQUE]  
"LDP MP Opaque Value Element basic type",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-11>>.
- [MP\_STATUS\_VALUE]  
"LDP MP Opaque Value Element extended type",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-14>>.
- [MSG\_TYPE\_NAME\_SPACE]  
"Message Type Name Space",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-2>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<http://www.rfc-editor.org/info/rfc4271>>.

- [RFC4278] Bellovin, S. and A. Zinin, "Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification", RFC 4278, DOI 10.17487/RFC4278, January 2006, <<http://www.rfc-editor.org/info/rfc4278>>.
- [RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <<http://www.rfc-editor.org/info/rfc6388>>.
- [RFC7361] Dutta, P., Balus, F., Stokes, O., Calvignac, G., and D. Fedyk, "LDP Extensions for Optimized MAC Address Withdrawal in a Hierarchical Virtual Private LAN Service (H-VPLS)", RFC 7361, DOI 10.17487/RFC7361, September 2014, <<http://www.rfc-editor.org/info/rfc7361>>.
- [STATUS\_CODE\_NAME\_SPACE]  
"Status Code Name Space",  
<<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#status-codes>>.
- [TLV\_TYPE\_NAME\_SPACE]  
"TLV Type Name Space", <<http://www.iana.org/assignments/ldp-namespaces/ldp-namespaces.xhtml#ldp-namespaces-4>>.

## Appendix A. LDP Label Distribution Procedures

This section specifies label distribution behavior in terms of LSR response to the following events:

- Receive Label Request Message;
- Receive Label Mapping Message;
- Receive Label Abort Request Message;
- Receive Label Release Message;
- Receive Label Withdraw Message;
- Recognize new FEC;
- Detect change in FEC next hop;
- Receive Notification Message / Label Request Aborted;
- Receive Notification Message / No Label Resources;
- Receive Notification Message / No Route;
- Receive Notification Message / Loop Detected;
- Receive Notification Message / Label Resources Available;
- Detect local label resources have become available;
- LSR decides to no longer label switch a FEC;
- Timeout of deferred label request.

The specification of LSR behavior in response to an event has three parts:

1. Summary. Prose that describes LSR response to the event in overview.
2. Context. A list of elements referred to by the Algorithm part of the specification. (See 3.)
3. Algorithm. An algorithm for LSR response to the event.

The summary may omit details of the LSR response, such as bookkeeping action or behavior dependent on the LSR label advertisement mode, control mode, or label retention mode in use. The intent is that the Algorithm fully and unambiguously specify the LSR response.

The algorithms in this section use procedures defined in the MPLS architecture specification RFC 3031 [RFC3031] for hop-by-hop routed traffic. These procedures are:

- Label Distribution procedure, which is performed by a downstream LSR to determine when to distribute a label for a FEC to LDP peers. The architecture defines four Label Distribution procedures:
  - o Downstream Unsolicited Independent Control, called PushUnconditional in RFC 3031 [RFC3031].
  - o Downstream Unsolicited Ordered Control, called PushConditional in RFC 3031 [RFC3031].
  - o Downstream On Demand Independent Control, called PulledUnconditional in RFC 3031 [RFC3031].
  - o Downstream On Demand Ordered Control, called PulledConditional in RFC 3031 [RFC3031].
- Label Withdrawal procedure, which is performed by a downstream LSR to determine when to withdraw a FEC label mapping previously distributed to LDP peers. The architecture defines a single Label Withdrawal procedure. Whenever an LSR breaks the binding between a label and a FEC, it MUST withdraw the FEC label mapping from all LDP peers to which it has previously sent the mapping.
- Label Request procedure, which is performed by an upstream LSR to determine when to explicitly request that a downstream LSR bind a label to a FEC and send it the corresponding label mapping. The architecture defines three Label Request procedures:

- o Request Never. The LSR never requests a label.
- o Request When Needed. The LSR requests a label whenever it needs one.
- o Request On Request. This procedure is used by non-label merging LSRs. The LSR requests a label when it receives a request for one, in addition to whenever it needs one.
- Label Release procedure, which is performed by an upstream LSR to determine when to release a previously received label mapping for a FEC. The architecture defines two Label Release procedures:
  - o Conservative Label retention, called ReleaseOnChange in RFC 3031 [RFC3031].
  - o Liberal Label retention, called NoReleaseOnChange in RFC 3031 [RFC3031].
- Label Use procedure, which is performed by an LSR to determine when to start using a FEC label for forwarding/switching. The architecture defines three Label Use procedures:
  - o Use Immediate. The LSR immediately uses a label received from a FEC next hop for forwarding/switching.
  - o Use If Loop Free. The LSR uses a FEC label received from a FEC next hop for forwarding/switching only if it has determined that by doing so it will not cause a forwarding loop.
  - o Use If Loop Not Detected. This procedure is the same as Use Immediate unless the LSR has detected a loop in the FEC LSP. Use of the FEC label for forwarding/switching will continue until the next hop for the FEC changes or the loop is no longer detected.

This version of LDP does not include a loop prevention mechanism; therefore, the procedures below do not make use of the Use If Loop Free procedure.

- Label No Route procedure (called the NotAvailable procedure in RFC 3031 [RFC3031]), which is performed by an upstream LSR to determine how to respond to a No Route notification from a downstream LSR in response to a request for a FEC label mapping. The architecture specification defines two Label No Route procedures:

- o Request Retry. The LSR should issue the label request at a later time.
- o No Request Retry. The LSR should assume that the downstream LSR will provide a label mapping when the downstream LSR has a next hop, and it should not reissue the request.

#### A.1. Handling Label Distribution Events

This section defines LDP label distribution procedures by specifying an algorithm for each label distribution event. The requirement on an LDP implementation is that its event handling must have the effect specified by the algorithms. That is, an implementation need not follow exactly the steps specified by the algorithms as long as the effect is identical.

The algorithms for handling label distribution events share common actions. The specifications below package these common actions into procedure units. Specifications for these common procedures are in their own Section, "Common Label Distribution Procedures", which follows this.

An implementation would use data structures to store information about protocol activity. This appendix specifies the information to be stored in sufficient detail to describe the algorithms, and assumes the ability to retrieve the information as needed. It does not specify the details of the data structures.

##### A.1.1. Receive Label Request

###### Summary:

The response by an LSR to receipt of a FEC label request from an LDP peer may involve one or more of the following actions:

- Transmission of a notification message to the requesting LSR indicating why a label mapping for the FEC cannot be provided;
- Transmission of a FEC label mapping to the requesting LSR;
- Transmission of a FEC label request to the FEC next hop;
- Installation of labels for forwarding/switching use by the LSR.

###### Context:

- LSR. The LSR handling the event.
- MsgSource. The LDP peer that sent the message.
- FEC. The FEC specified in the message.
- RAttributes. Attributes received with the message, e.g., Hop Count, Path Vector.
- SAttributes. Attributes to be included in the Label Request message, if any, propagated to FEC Next Hop.
- StoredHopCount. The hop count, if any, previously recorded for the FEC.

Algorithm:

- LRq.1: Execute procedure Check\_Received\_Attributes (MsgSource, LabelRequest, RAttributes).  
If Loop Detected, goto LRq.4.
- LRq.2: Is there a Next Hop for FEC?  
If not, goto LRq.5.
- LRq.3: Is MsgSource the Next Hop?  
If not, goto LRq.6.
- LRq.4: Execute procedure Send\_Notification (MsgSource, Loop Detected).  
Goto LRq.13
- LRq.5: Execute procedure Send\_Notification (MsgSource, No Route).  
Goto LRq.13.
- LRq.6: Has LSR previously received a label request for FEC from MsgSource?  
If not, goto LRq.8. (See Note 1.)
- LRq.7: Is the label request a duplicate request?  
If so, goto LRq.13. (See Note 2.)
- LRq.8: Record label request for FEC received from MsgSource and mark it pending.
- LRq.9: Perform LSR Label Distribution procedure:

For Downstream Unsolicited Independent Control OR For  
Downstream On Demand Independent Control

1. Has LSR previously received and retained a label mapping for FEC from Next Hop?  
Is so, set Propagating to IsPropagating.  
If not, set Propagating to NotPropagating.
2. Execute procedure  
Is so, set Propagating to IsPropagating.  
If not, set Propagating to NotPropagating.
3. Execute procedure Send\_Label (MsgSource, FEC, SAttributes).
4. Is LSR egress for FEC? OR Has LSR previously received and retained a label mapping for FEC from Next Hop?  
Is so, goto LRq.11.  
If not, goto LRq.10.

For Downstream Unsolicited Ordered Control OR For  
Downstream On Demand Ordered Control

1. Is LSR egress for FEC? OR Has LSR previously received and retained a label mapping for FEC from Next Hop?  
(See Note 3.)  
If not, goto LRq.10.
2. Execute procedure  
Prepare\_Label\_Mapping\_Attributes(MsgSource, FEC, RAttributes, SAttributes, IsPropagating, StoredHopCount)
3. Execute procedure Send\_Label (MsgSource, FEC, SAttributes).  
Goto LRq.11.

LRq.10: Perform LSR Label Request procedure:

For Request Never

1. Goto LRq.13.

For Request When Needed OR  
For Request On Request

1. Execute procedure `Prepare_Label_Request_Attributes` (Next Hop, FEC, RAttributes, SAttributes);
2. Execute procedure `Send_Label_Request` (Next Hop, FEC, SAttributes).  
Goto LRq.13.

LRq.11: Has LSR successfully sent a label for FEC to MsgSource?  
If not, goto LRq.13. (See Note 4.)

LRq.12: Perform LSR Label Use procedure.

For Use Immediate OR For Use If Loop Not Detected

1. Install label sent to MsgSource and label from Next Hop (if LSR is not egress) for forwarding/switching use.

LRq.13: DONE.

Notes:

1. In the case where MsgSource is a non-label merging LSR, it will send a label request for each upstream LDP peer that has requested a label for FEC from it. The LSR must be able to distinguish such requests from a non-label merging MsgSource from duplicate label requests.

The LSR uses the message ID of received Label Request messages to detect duplicate requests. This means that an LSR (the upstream peer) may not reuse the message ID used for a Label Request until the Label Request transaction has completed.

2. When an LSR sends a label request to a peer, it records that the request has been sent and marks it as outstanding. As long as the request is marked outstanding, the LSR SHOULD NOT send another request for the same label to the peer. Such a second request would be a duplicate. The `Send_Label_Request` procedure described below obeys this rule.

A duplicate label request is considered a protocol error and SHOULD be dropped by the receiving LSR (perhaps with a suitable notification returned to MsgSource).

3. If the LSR is not merge-capable, this test will fail.
4. The `Send_Label` procedure may fail due to lack of label resources, in which case the LSR SHOULD NOT perform the Label Use procedure.



## A.1.1.2. Receive Label Mapping

## Summary:

The response by an LSR to receipt of a FEC label mapping from an LDP peer may involve one or more of the following actions:

- Transmission of a Label Release message for the FEC label to the LDP peer;
- Transmission of Label Mapping messages for the FEC to one or more LDP peers;
- Installation of the newly learned label for forwarding/switching use by the LSR.

## Context:

- LSR. The LSR handling the event.
- MsgSource. The LDP peer that sent the message.
- FEC. The FEC specified in the message.
- Label. The label specified in the message.
- PrevAdvLabel. The label for the FEC, if any, previously advertised to an upstream peer. Assuming no label was previously advertised, this is the same label as the one in the Label Mapping message being processed.
- StoredHopCount. The hop count previously recorded for the FEC.
- RAttributes. Attributes received with the message, e.g., Hop Count, Path Vector.
- SAttributes to be included in the Label Mapping message, if any, propagated to upstream peers.

## Algorithm:

LMp.1: Does the received label mapping match an outstanding label request for FEC previously sent to MsgSource?

If not, goto LMp.3.

LMp.2: Delete record of outstanding FEC label request.

- LMp.3: Execute procedure Check\_Received\_Attributes (MsgSource, LabelMapping, RAttributes).
- If No Loop Detected, goto LMp.9.
- LMp.4: Does the LSR have a previously received label mapping for FEC from MsgSource? (See Note 1.)
- If not, goto LMp.8. (See Note 2.)
- LMp.5: Does the label previously received from MsgSource match Label (i.e., the label received in the message)? (See Note 3.)
- If not, goto LMp.8. (See Note 4.)
- LMp.6: Delete matching label mapping for FEC previously received from MsgSource.
- LMp.7: Remove Label from forwarding/switching use. (See Note 5.)
- LMp.8: Execute procedure Send\_Message (MsgSource, Label Release, FEC, Label, Loop Detected Status code).
- Goto LMp.33.
- LMp.9: Does LSR have a previously received label mapping for FEC from MsgSource for the LSP in question? (See Note 6.)
- If not, goto LMp.11.
- LMp.10: Does the label previously received from MsgSource match Label (i.e., the label received in the message)? (See Note 3.)
- OR
- Is the received label mapping in response to a previously outstanding label request sent to MsgSource? (See Note 12.)
- If so, goto LMp.11.
- LMp.10a: Is LSR operating in Downstream Unsolicited mode? If so, delete the label mapping for the label previously received from MsgSource and remove it from forwarding/switching use. Execute procedure Send\_Message (MsgSource, Label Release, FEC, label previously received from MsgSource).
- LMp.11: Determine the Next Hop for FEC.

- LMp.12: Is MsgSource the Next Hop for FEC?  
If so, goto LMp.14.
- LMp.13: Perform LSR Label Release procedure:  
For Conservative Label retention:  
1. Goto LMp.32.  
For Liberal Label retention:  
1. Record label mapping for FEC with Label and RAttributes has been received from MsgSource.  
Goto LMp.33.
- LMp.14: Is LSR an ingress for FEC?  
If not, goto LMp.16.
- LMp.15: Install Label for forwarding/switching use.
- LMp.16: Record label mapping for FEC with Label and RAttributes has been received from MsgSource.
- LMp.17: Iterate through LMp.31 for each Peer. (See Note 7).
- LMp.18: Has LSR previously sent a label mapping for FEC to Peer for the LSP in question? (See Note 8.)  
If so, goto LMp.22.
- LMp.19: Is the Downstream Unsolicited Ordered Control Label Distribution procedure being used by LSR?  
If not, goto LMp.28.
- LMp.20: Execute procedure Prepare\_Label\_Mapping\_Attributes (Peer, FEC, RAttributes, SAttributes, IsPropagating, StoredHopCount).
- LMp.21: Execute procedure Send\_Message (Peer, Label Mapping, FEC, PrevAdvLabel, SAttributes). (See Note 13.)  
Goto LMp.28.
- LMp.22: Iterate through LMp.27 for each label mapping for FEC previously sent to Peer.

- LMp.23: Are RAttributes in the received label mapping consistent with those previously sent to Peer? If so, continue iteration from LMp.22 for next label mapping. (See Note 9.)
- LMp.24: Execute procedure Prepare\_Label\_Mapping\_Attributes (Peer, FEC, RAttributes, SAttributes, IsPropagating, StoredHopCount).
- LMp.25: Execute procedure Send\_Message (Peer, Label Mapping, FEC, PrevAdvLabel, SAttributes). (See Note 10.)
- LMp.26: Update record of label mapping for FEC previously sent to Peer to include the new attributes sent.
- LMp.27: End iteration from LMp.22.
- LMp.28: Does LSR have any label requests for FEC from Peer marked as pending?
- If not, goto LMp.30.
- LMp.29: Perform LSR Label Distribution procedure:
- For Downstream Unsolicited Independent Control OR For Downstream Unsolicited Ordered Control
1. Execute procedure Prepare\_Label\_Mapping\_Attributes (Peer, FEC, RAttributes, SAttributes, IsPropagating, UnknownHopCount).
  2. Execute procedure Send\_Label (Peer, FEC, SAttributes).
- If the procedure fails, continue iteration for next Peer at LMp.17.
3. If no pending requests exist for Peer, goto LMp.30. (See Note 11.)
- For Downstream On Demand Independent Control  
OR  
For Downstream On Demand Ordered Control
1. Iterate through Step 5 for each pending label request for FEC from Peer marked as pending.

2. Execute procedure `Prepare_Label_Mapping_Attributes` (Peer, FEC, RAttributes, SAttributes, IsPropagating, UnknownHopCount)
3. Execute procedure `Send_Label` (Peer, FEC, SAttributes). If the procedure fails, continue iteration for next Peer at LMp.17.
4. Delete record of pending request.
5. End iteration from Step 1.
6. Goto LMp.30.

LMp.30: Perform LSR Label Use procedure:

For Use Immediate OR For Use If Loop Not Detected

1. Iterate through Step 3 for each label mapping for FEC previously sent to Peer.
2. Install label received and label sent to Peer for forwarding/switching use.
3. End iteration from Step 1.
4. Goto LMp.31.

LMp.31: End iteration from LMp.17.  
Go to LMp.33.

LMp.32: Execute procedure `Send_Message` (MsgSource, Label Release, FEC, Label).

LMp.33: DONE.

Notes:

1. If the LSR is merging, there should be at most 1 received mapping for the FEC for the LSP in question. In the non-merging case, there could be multiple received mappings for the FEC for the LSP in question.
2. If the LSR has detected a loop and it has not previously received a label mapping from MsgSource for the FEC, it simply releases the label.

3. Does the Label received in the message match any of the 1 or more label mappings identified in the previous step (LMp.4 or LMp.9)?
4. An unsolicited mapping with a different label from the same peer would be an attempt to establish multipath label switching, which is not supported in this version of LDP.
5. If the Label is not in forwarding/switching use, LMp.7 has no effect.
6. If the received label mapping message matched an outstanding label request in LMp.1, then (by definition) the LSR has not previously received a label mapping for FEC for the LSP in question. If the LSR is merging upstream labels for the LSP in question, there should be at most 1 received mapping. In the non-merging case, there could be multiple received label mappings for the same FEC, one for each resulting LSP.
7. The LMp.17 iteration includes MsgSource in order to handle the case where the LSR is operating in Downstream Unsolicited Ordered Control mode. Ordered Control prevents the LSR from advertising a label for the FEC until it has received a label mapping from its next hop (MsgSource) for the FEC.
8. If the LSR is merging the LSP, it may have previously sent label mappings for the FEC LSP to one or more peers. If the LSR is not merging, it may have sent a label mapping for the LSP in question to at most one LSR.
9. The Loop Detection Path Vector attribute is considered in this check. If the received RAttributes include a Path Vector and no Path Vector had been previously sent to the Peer, or if the received Path Vector is inconsistent with the Path Vector previously sent to the Peer, then the attributes are considered to be inconsistent. Note that an LSR is not required to store a received Path Vector after it propagates the Path Vector in a mapping message. If an LSR does not store the Path Vector, it has no way to check the consistency of a newly received Path Vector. This means that whenever such an LSR receives a mapping message carrying a Path Vector it must always propagate the Path Vector.
10. LMp.22 through LMp.27 deal with a situation that can arise when the LSR is using independent control and it receives a mapping from the downstream peer after it has sent a mapping to an upstream peer. In this situation, the LSR needs to propagate any changed attributes, such as Hop Count, upstream. If Loop

Detection is configured on, the propagated attributes must include the Path Vector.

11. An LSR operating in Downstream Unsolicited mode MUST process any Label Request messages it receives. If there are pending label requests, fall through into the Downstream on Demand procedures in order to satisfy the pending requests.
12. As determined by step LMP.1.
13. An LSR operating in Ordered Control mode may choose to skip at this stage the peer from which it received the advertisement that caused it to generate the label-map message. Doing so will in effect provide a form of split-horizon.

#### A.1.3. Receive Label Abort Request

##### Summary:

When an LSR receives a Label Abort Request message from a peer, it checks whether it has already responded to the label request in question. If it has, it silently ignores the message. If it has not, it sends the peer a Label Request Aborted Notification. In addition, if it has a label request outstanding for the LSP in question to a downstream peer, it sends a Label Abort Request to the downstream peer to abort the LSP.

##### Context:

- LSR. The LSR handling the event.
- MsgSource. The LDP peer that sent the message.
- FEC. The FEC specified in the message.
- RequestMessageID. The message ID of the label request message to be aborted.
- Next Hop. The next hop for the FEC.

##### Algorithm:

- LABR.1 Does the message match a previously received Label Request message from MsgSource? (See Note 1.)  
If not, goto LABR.12.
- LABR.2 Has LSR responded to the previously received label request?  
If so, goto LABR.12.

- LABR.3    Execute procedure Send\_Message (MsgSource, Notification, Label Request Aborted, TLV), where TLV is the Label Request Message ID TLV received in the Label Abort Request message.
- LABR.4    Does LSR have a Label Request message outstanding for FEC?  
          If so, goto LABR.7.
- LABR.5    Does LSR have a label mapping for FEC?  
          If not, goto LABR.11.
- LABR.6    Generate Event: Received Label Release message for FEC from MsgSource. (See Note 2.)  
          Goto LABR.11.
- LABR.7    Is LSR merging the LSP for FEC?  
          If not, goto LABR.9.
- LABR.8    Are there outstanding label requests for this FEC?  
          If so, goto LABR.11.
- LABR.9    Execute procedure Send\_Message (Next Hop, Label Abort Request, FEC, TLV), where TLV is a Label Request message ID TLV containing the Message ID used by the LSR in the outstanding Label Request message.
- LABR.10   Record that a label abort request for FEC is pending.
- LABR.11   Delete record of label request for FEC from MsgSource.
- LABR.12   DONE.

Notes:

1. LSR uses FEC and the Label Request message ID TLV carried by the label abort request to locate its record (if any) for the previously received label request from MsgSource.
2. If LSR has received a label mapping from NextHop, it should behave as if it had advertised a label mapping to MsgSource and MsgSource has released it.

A.1.4. Receive Label Release

Summary:

When an LSR receives a Label Release message for a FEC from a peer, it checks whether other peers hold the released label. If none do, the LSR removes the label from forwarding/switching use,



if it has not already done so, and if the LSR holds a label mapping from the FEC next hop, it releases the label mapping.

Context:

- LSR. The LSR handling the event.
- MsgSource. The LDP peer that sent the message.
- Label. The label specified in the message.
- FEC. The FEC specified in the message.

Algorithm:

- LRl.1 Does FEC match a known FEC? If not, goto LRl.14.
- LRl.2 Remove MsgSource from record of peers that hold Label for FEC. (See Note 1.)
- LRl.3 Does message match an outstanding label withdraw for FEC previously sent to MsgSource?  
If not, goto LRl.5
- LRl.4 Delete record of outstanding label withdraw for FEC previously sent to MsgSource.
- LRl.5 Is LSR merging labels for this FEC? If not, goto LRl.7.  
(See Note 2.)
- LRl.6 Does LSR have outstanding label advertisements for this FEC?  
If so, goto LRl.11.
- LRl.7 Is LSR egress for the FEC?  
If so, goto LRl.11.
- LRl.8 Is there a Next Hop for FEC? AND Does LSR have a previously received label mapping for FEC from Next Hop?  
If not, goto LRl.11.
- LRl.9 Is LSR configured to propagate releases?  
If not, goto LRl.11. (See Note 3.)
- LRl.10 Execute procedure Send\_Message (Next Hop, Label Release, FEC, Label from Next Hop).
- LRl.11 Remove Label from forwarding/switching use for traffic from MsgSource.

LRl.12 Do any peers still hold Label for FEC?  
If so, goto LRl.14.

LRl.13 Free the Label.

LRl.14 DONE.

Notes:

1. If LSR is using Downstream Unsolicited label distribution, it SHOULD NOT re-advertise a label mapping for FEC to MsgSource until MsgSource requests it.
2. LRl.5 through LRl.9 deal with determining whether where the LSR should propagate the Label Release to a downstream peer (LRl.9).
3. If LRl.9 is reached, no upstream LSR holds a label for the FEC, and the LSR holds a label for the FEC from the FEC Next Hop. The LSR could propagate the Label Release to the Next Hop. By propagating the Label Release, the LSR releases a potentially scarce label resource. In doing so, it also increases the latency for re-establishing the LSP should MsgSource or some other upstream LSR send it a new Label Request for FEC. Whether or not to propagate the release is not a protocol issue. Label distribution will operate properly whether or not the release is propagated. The decision to propagate or not should take into consideration factors such as, whether labels are a scarce resource in the operating environment, the importance of keeping LSP setup latency low by keeping the amount of signaling required small, and whether LSP setup is ingress-controlled or egress-controlled in the operating environment.

A.1.5. Receive Label Withdraw

Summary:

When an LSR receives a Label Withdraw message for a FEC from an LDP peer, it responds with a Label Release message and it removes the label from any forwarding/switching use. If Ordered Control is in use, the LSR sends a Label Withdraw message to each LDP peer to which it had previously sent a label mapping for the FEC. If the LSR is using Downstream on Demand label advertisement with independent control, it then acts as if it had just recognized the FEC.

Context:

- LSR. The LSR handling the event.

- MsgSource. The LDP peer that sent the message.
- Label. The label specified in the message.
- FEC. The FEC specified in the message.

Algorithm:

- LWd.a Remove Label from forwarding/switching use. (See Note 1.)
- LWd.b Execute procedure Send\_Message (MsgSource, Label Release, FEC, Label).
- LWd.c Has LSR previously received and retained a matching label mapping for FEC from MsgSource?  
If not, goto LWd.13.
- LWd.d Delete matching label mapping for FEC previously received from MsgSource.
- LWd.e Is LSR using Ordered Control?  
If so, goto LWd.8.
- LWd.f Is MsgSource using Downstream On Demand label advertisement?  
If not, goto LWd.13.
- LWd.g Generate Event: Recognize New FEC for FEC.  
Goto LWd.13. (See Note 2.)
- LWd.h Iterate through LWd.12 for each Peer, other than MsgSource.
- LWd.i Has LSR previously sent a label mapping for FEC to Peer?  
If not, continue iteration for next Peer at LWd.8.
- LWd.j Does the label previously sent to Peer "map" to the withdrawn Label?  
If not, continue iteration for next Peer at LWd.8. (See Note 3.)
- LWd.k Execute procedure Send\_Label\_Withdraw (Peer, FEC, Label previously sent to Peer).
- LWd.l End iteration from LWd.8.
- LWd.m DONE.

Notes:

1. If the Label is not in forwarding/switching use, LwD.1 has no effect.
2. LwD.7 handles the case where the LSR is using Downstream On Demand label distribution with independent control. In this situation, the LSR should send a label request to the FEC next hop as if it had just recognized the FEC.
3. LwD.10 handles both label merging (one or more incoming labels map to the same outgoing label) and no label merging (one label maps to the outgoing label) cases.

#### A.1.6. Recognize New FEC

##### Summary:

The response by an LSR to learning a new FEC via the routing table may involve one or more of the following actions:

- Transmission of label mappings for the FEC to one or more LDP peers;
- Transmission of a label request for the FEC to the FEC next hop;
- Any of the actions that can occur when the LSR receives a label mapping for the FEC from the FEC next hop.

##### Context:

LSR. The LSR handling the event.

FEC. The newly recognized FEC.

Next Hop. The next hop for the FEC.

InitAttributes. Attributes to be associated with the new FEC.  
(See Note 1.)

SAttributes. Attributes to be included in Label Mapping or Label Request messages, if any, sent to peers.

StoredHopCount. Hop count associated with FEC label mapping, if any, previously received from Next Hop.

##### Algorithm:

## FEC.1 Perform LSR Label Distribution procedure:

For Downstream Unsolicited Independent Control

1. Iterate through 5 for each Peer.
2. Has LSR previously received and retained a label mapping for FEC from Next Hop?  
If so, set Propagating to IsPropagating.  
If not, set Propagating to NotPropagating.
3. Execute procedure Prepare\_Label\_Mapping\_Attributes (Peer, FEC, InitAttributes, SAttributes, Propagating, Unknown hop count(0)).
4. Execute procedure Send\_Label (Peer, FEC, SAttributes).
5. End iteration from 1.  
Goto FEC.2.

For Downstream Unsolicited Ordered Control

1. Iterate through 5 for each Peer.
2. Is LSR egress for the FEC? OR Has LSR previously received and retained a label mapping for FEC from Next Hop?  
If not, continue iteration for next Peer.
3. Execute procedure Prepare\_Label\_Mapping\_Attributes (Peer, FEC, InitAttributes, SAttributes, Propagating, StoredHopCount).
4. Execute procedure Send\_Label (Peer, FEC, SAttributes).
5. End iteration from 1.  
Goto FEC.2.

For Downstream On Demand Independent Control

OR

For Downstream On Demand Ordered Control

1. Goto FEC.2. (See Note 2.)

FEC.2 Has LSR previously received and retained a label mapping for FEC from Next Hop?  
If so, goto FEC.5

FEC.3 Is Next Hop an LDP peer?  
If not, Goto FEC.6

FEC.4 Perform LSR Label Request procedure:

For Request Never

1. Goto FEC.6

For Request When Needed

OR

For Request On Request

1. Execute procedure Prepare\_Label\_Request\_Attributes (Next Hop, FEC, InitAttributes, SAttributes);

2. Execute procedure Send\_Label\_Request (Next Hop, FEC, Goto FEC.6.

FEC.5 Generate Event: Received Label Mapping from Next Hop. (See Note 3.)

FEC.6 DONE.

Notes:

1. An example of an attribute that might be part of InitAttributes is one that specifies desired LSP characteristics, such as Class of Service (CoS). (Note that while the current version of LDP does not specify a CoS attribute, LDP extensions may.)

The means by which FEC InitAttributes, if any, are specified is beyond the scope of LDP. Note that the InitAttributes will not include a known Hop Count or a Path Vector.

2. An LSR using Downstream On Demand label distribution would send a label only if it had a previously received label request marked as pending. The LSR would have no such pending requests because it responds to any label request for an unknown FEC by sending the requesting LSR a No Route notification and discarding the label request; see LRq.3

3. If the LSR has a label for the FEC from the Next Hop, it should behave as if it had just received the label from the Next Hop. This occurs in the case of Liberal Label retention mode.

#### A.1.7. Detect Change in FEC Next Hop

##### Summary:

The response by an LSR to a change in the next hop for a FEC may involve one or more of the following actions:

- Removal of the label from the FEC's old next hop from forwarding/switching use;
- Transmission of label mapping messages for the FEC to one or more LDP peers;
- Transmission of a label request to the FEC's new next hop;
- Any of the actions that can occur when the LSR receives a label mapping from the FEC's new next hop.

##### Context:

- LSR. The LSR handling the event.
- FEC. The FEC whose next hop changed.
- New Next Hop. The current next hop for the FEC.
- Old Next Hop. The previous next hop for the FEC.
- OldLabel. Label, if any, previously received from Old Next Hop.
- CurAttributes. The attributes, if any, currently associated with the FEC.
- SAttributes. Attributes to be included in the Label Request message, if any, sent to New Next Hop.

##### Algorithm:

- NH.1 Has LSR previously received and retained a label mapping for FEC from Old Next Hop?  
If not, goto NH.6.

- NH.2 Remove label from forwarding/switching use. (See Note 1.)
- NH.3 Is LSR using Liberal Label retention?
- If so, goto NH.6.
- NH.4 Execute procedure Send\_Message (Old Next Hop, Label Release, OldLabel).
- NH.5 Delete label mapping for FEC previously received from Old Next Hop.
- NH.6 Does LSR have a label request pending with Old Next Hop?
- If not, goto NH.10.
- NH.7 Is LSR using Conservative Label retention?
- If not, goto NH.10.
- NH.8 Execute procedure Send\_Message (Old Next Hop, Label Abort Request, FEC, TLV), where TLV is a Label Request Message ID TLV that carries the message ID of the pending label request.
- NH.9 Record that a label abort request is pending for FEC with Old Next Hop.
- NH.10 Is there a New Next Hop for FEC?
- If not, goto NH.16.
- NH.11 Has LSR previously received and retained a label mapping for FEC from New Next Hop?
- If not, goto NH.13.
- NH.12 Generate Event: Received Label Mapping from New Next Hop. Goto NH.20. (See Note 2.)
- NH.13 Is LSR using Downstream on Demand advertisement? OR Is Next Hop using Downstream on Demand advertisement? OR Is LSR using Conservative Label retention? (See Note 3.)
- If so, goto NH.14.
- If not, goto NH.20.



NH.14 Execute procedure `Prepare_Label_Request_Attributes` (Next Hop, FEC, `CurAttributes`, `SAttributes`).

NH.15 Execute procedure `Send_Label_Request` (New Next Hop, FEC, `SAttributes`). (See Note 4.)

Goto NH.20.

NH.16 Iterate through NH.19 for each Peer.

NH.17 Has LSR previously sent a label mapping for FEC to Peer? If not, continue iteration for next Peer at NH.16

NH.18 Execute procedure `Send_Label_Withdraw` (Peer, FEC, Label previously sent to Peer).

NH.19 End iteration from NH.16.

NH.20 DONE.

Notes:

- If the Label is not in forwarding/switching use, NH.2 has no effect.
- If the LSR has a label for the FEC from the New Next Hop, it should behave as if it had just received the label from the New Next Hop.
- The purpose of the check on label retention mode is to avoid a race with steps LMP.12-LMP.13 of the procedure for handling a Label Mapping message where the LSR operating in Conservative Label retention mode may have released a label mapping received from the New Next Hop before it detected that the FEC next hop had changed.
- Regardless of the Label Request procedure in use by the LSR, it MUST send a label request if the conditions in NH.13 hold. Therefore, it executes the `Send_Label_Request` procedure directly rather than perform the LSR Label Request procedure.

A.1.8. Receive Notification / Label Request Aborted

Summary:

When an LSR receives a Label Request Aborted notification from an LDP peer, it records that the corresponding label request transaction, if any, has completed.

Context:

- LSR. The LSR handling the event.
- FEC. The FEC for which a label was requested.
- RequestMessageID. The message ID of the label request message to be aborted.
- MsgSource. The LDP peer that sent the Notification message.

Algorithm:

LRqA.a Does the notification correspond to an outstanding label request abort for FEC? (See Note 1.)

If not, goto LRqA.3.

LRqA.b Record that the label request for FEC has been aborted.

LRqA.c DONE.

Note:

1. The LSR uses the FEC and RequestMessageID to locate its record, if any, of the outstanding label request abort.

#### A.1.9. Receive Notification / No Label Resources

Summary:

When an LSR receives a No Label Resources notification from an LDP peer, it stops sending label request messages to the peer until it receives a Label Resources Available Notification from the peer.

Context:

LSR. The LSR handling the event.

FEC. The FEC for which a label was requested.

MsgSource. The LDP peer that sent the Notification message.

Algorithm:

NoRes.1 Delete record of outstanding label request for FEC sent to MsgSource.

- NoRes.2 Record that label mapping for FEC from MsgSource is needed but that no label resources are available.
- NoRes.3 Set status record indicating it is not OK to send label requests to MsgSource.
- NoRes.4 DONE.

#### A.1.10. Receive Notification / No Route

##### Summary:

When an LSR receives a No Route notification from an LDP peer in response to a Label Request message, the Label No Route procedure in use dictates its response. The LSR either will take no further action, or it will defer the label request by starting a timer and send another Label Request message to the peer when the timer later expires.

##### Context:

- LSR. The LSR handling the event.
- FEC. The FEC for which a label was requested.
- Attributes. The attributes associated with the label request.
- MsgSource. The LDP peer that sent the Notification message.

##### Algorithm:

- NoNH.1 Delete record of outstanding label request for FEC sent to MsgSource.
- NoNH.2 Perform LSR Label No Route procedure.
- For Request No Retry
1. Goto NoNH.3.
- For Request Retry
1. Record deferred label request for FEC and Attributes to be sent to MsgSource.
2. Start timeout. Goto NoNH.3.
- NoNH.3 DONE.

## A.1.11. Receive Notification / Loop Detected

## Summary:

When an LSR receives a Loop Detected Status Code from an LDP peer in response to a Label Request message or a Label Mapping message, it behaves as if it had received a No Route notification.

## Context:

See "Receive Notification / No Route".

## Algorithm:

See "Receive Notification / No Route".

## Note:

1. When the Loop Detected notification is in response to a Label Request message, it arrives in a Status Code TLV in a Notification message. When it is in response to a Label Mapping message, it arrives in a Status Code TLV in a Label Release message.

## A.1.12. Receive Notification / Label Resources Available

## Summary:

When an LSR receives a Label Resources Available notification from an LDP peer, it resumes sending label requests to the peer.

## Context:

- LSR. The LSR handling the event.
- MsgSource. The LDP peer that sent the Notification message.
- SAttributes. Attributes stored with the postponed Label Request message.

## Algorithm:

- Res.1 Set status record indicating it is OK to send label requests to MsgSource.
- Res.2 Iterate through Res.6 for each record of a FEC label mapping needed from MsgSource for which no label resources are available.

Res.3 Is MsgSource the next hop for FEC?

If not, goto Res.5.

Res.4 Execute procedure Send\_Label\_Request (MsgSource, FEC, SAttributes). If the procedure fails, terminate iteration.

Res.5 Delete record that no resources are available for a label mapping for FEC needed from MsgSource.

Res.6 Res.6 End iteration from Res.2.

Res.7 DONE.

#### A.1.13. Detect Local Label Resources Have Become Available

##### Summary:

After an LSR has sent a No Label Resources notification to an LDP peer, when label resources later become available it sends a Label Resources Available notification to each such peer.

##### Context:

- LSR. The LSR handling the event.
- Attributes. Attributes stored with the postponed Label Mapping message.

##### Algorithm:

ResA.1 Iterate through ResA.4 for each Peer to which LSR has previously sent a No Label Resources notification.

ResA.2 Execute procedure Send\_Notification (Peer, Label Resources Available).

ResA.3 Delete record that No Label Resources notification was previously sent to Peer.

ResA.4 End iteration from ResA.1.

ResA.5 Iterate through ResA.8 for each record of a label mapping needed for FEC for Peer but no-label-resources. (See Note 1.)

ResA.6 Execute procedure Send\_Label (Peer, FEC, Attributes). If the procedure fails, terminate iteration.

ResA.7 Clear record of FEC label mapping needed for peer but no-label-resources.

ResA.8 End iteration from ResA.5

ResA.9 DONE.

Note:

1. Iteration ResA.5 through ResA.8 handles the situation where the LSR is using Downstream Unsolicited label distribution and was previously unable to allocate a label for a FEC.

#### A.1.14. LSR Decides to No Longer Label Switch a FEC

Summary:

An LSR may unilaterally decide to no longer label switch a FEC for an LDP peer. An LSR that does so MUST send a Label Withdraw message for the FEC to the peer.

Context:

- Peer. The peer.
- FEC. The FEC.
- PrevAdvLabel. The label for the FEC previously advertised to the Peer.

Algorithm:

NoLS.1 Execute procedure Send\_Label\_Withdraw (Peer, FEC, PrevAdvLabel). (See Note 1.)

DONE.

Note:

1. The LSR may remove the label from forwarding/switching use as part of this event or as part of processing the Label Release from the peer in response to the label withdraw. If the LSR doesn't wait for the Label Release message from the peer, it SHOULD NOT reuse the label until it receives the Label Release.

## A.1.15. Timeout of Deferred Label Request

## Summary:

Label requests are deferred in response to No Route and Loop Detected notifications. When a deferred FEC label request for a peer times out, the LSR sends the label request.

## Context:

- LSR. The LSR handling the event.
- FEC. The FEC associated with the timeout event.
- Peer. The LDP peer associated with the timeout event.
- Attributes. Attributes stored with the deferred Label Request message.

## Algorithm:

TO.1 Retrieve the record of the deferred label request.

TO.2 Is Peer the next hop for FEC?

    If not, goto TO.4.

TO.3 Execute procedure Send\_Label\_Request (Peer, FEC).

TO.4 DONE.

## A.2. Common Label Distribution Procedures

This section specifies utility procedures used by the algorithms that handle label distribution events.

## A.2.1. Send\_Label

## Summary:

The Send\_Label procedure allocates a label for a FEC for an LDP peer, if possible, and sends a label mapping for the FEC to the peer. If the LSR is unable to allocate the label and if it has a pending label request from the peer, it sends the LDP peer a No Label Resources notification.

## Parameters:

- Peer. The LDP peer to which the label mapping is to be sent.
- FEC. The FEC for which a label mapping is to be sent.
- Attributes. Attributes to be included with the label mapping.

Additional Context:

- LSR. The LSR executing the procedure.
- Label. The label allocated and sent to Peer.

Algorithm:

- SL.1 Does LSR have a label to allocate?  
If not, goto SL.9.
- SL.2 Allocate Label and bind it to the FEC.
- SL.3 Install Label for forwarding/switching use.
- SL.4 Execute procedure Send\_Message (Peer, Label Mapping, FEC, Label, Attributes).
- SL.5 Record label mapping for FEC with Label and Attributes has been sent to Peer.
- SL.6 Does LSR have a record of a FEC label request from Peer marked as pending?  
If not, goto SL.8.
- SL.7 Delete record of pending label request for FEC from Peer.
- SL.8 Return success.
- SL.9 Does LSR have a label request for FEC from Peer marked as pending?  
If not, goto SL.13.
- SL.10 Execute procedure Send\_Notification (Peer, No Label Resources).
- SL.11 Delete record of pending label request for FEC from Peer.
- SL.12 Record No Label Resources notification has been sent to Peer.  
Goto SL.14.



SL.13 Record label mapping needed for FEC and Attributes for Peer, but no-label-resources. (See Note 1.)

SL.14 Return failure.

Note:

1. SL.13 handles the case of Downstream Unsolicited label distribution when the LSR is unable to allocate a label for a FEC to send to a Peer.

#### A.2.2. Send\_Label\_Request

Summary:

An LSR uses the Send\_Label\_Request procedure to send a request for a label for a FEC to an LDP peer if currently permitted to do so.

Parameters:

- Peer. The LDP peer to which the label request is to be sent.
- FEC. The FEC for which a label request is to be sent.
- Attributes. Attributes to be included in the label request, e.g., Hop Count, Path Vector.

Additional Context:

- LSR. The LSR executing the procedure.

Algorithm:

- SLRq.1 Has a label request for FEC previously been sent to Peer and is it marked as outstanding?  
If so, Return success. (See Note 1.)
- SLRq.2 Is status record indicating it is OK to send label requests to Peer set?  
If not, goto SLRq.6
- SLRq.3 Execute procedure Send\_Message (Peer, Label Request, FEC, Attributes).
- SLRq.4 Record that label request for FEC has been sent to Peer and mark it as outstanding.
- SLRq.5 Return success.

SLRq.6 Postpone the label request by recording that label mapping for FEC and Attributes from Peer is needed but that no label resources are available.

SLRq.7 Return failure.

Note:

1. If the LSR is a non-merging LSR, it must distinguish between attempts to send label requests for a FEC triggered by different upstream LDP peers from duplicate requests. This procedure will not send a duplicate label request.

#### A.2.3. Send\_Label\_Withdraw

Summary:

An LSR uses the Send\_Label\_Withdraw procedure to withdraw a label for a FEC from an LDP peer. To do this, the LSR sends a Label Withdraw message to the peer.

Parameters:

- Peer. The LDP peer to which the label withdraw is to be sent.
- FEC. The FEC for which a label is being withdrawn.
- Label. The label being withdrawn.

Additional Context:

- LSR. The LSR executing the procedure.

Algorithm:

SWd.1 Execute procedure Send\_Message (Peer, Label Withdraw, FEC, Label).

SWd.2 Record that label withdraw for FEC has been sent to Peer and mark it as outstanding.

#### A.2.4. Send\_Notification

Summary:

An LSR uses the Send\_Notification procedure to send an LDP peer a Notification message.

## Parameters

- Peer. The LDP peer to which the Notification message is to be sent.
- Status. Status code to be included in the Notification message.

## Additional Context:

None.

## Algorithm:

Snt.1 Execute procedure Send\_Message (Peer, Notification, Status)

## A.2.5. Send\_Message

## Summary:

An LSR uses the Send\_Message procedure to send an LDP peer an LDP message.

## Parameters:

Peer. The LDP peer to which the message is to be sent.

Message Type. The type of message to be sent.

Additional message contents . . . .

## Additional Context:

None.

## Algorithm:

This procedure is the means by which an LSR sends an LDP message of the specified type to the specified LDP peer.

## A.2.6. Check\_Received\_Attributes

## Summary:

Check the attributes received in a Label Mapping or Label Request message. If the attributes include a Hop Count or Path Vector, perform a Loop Detection check. If a loop is detected, cause a Loop Detected Notification message to be sent to MsgSource.

## Parameters:

- MsgSource. The LDP peer that sent the message.
- MsgType. The type of message received.
- RAttributes. The attributes in the message.

## Additional Context:

LSR Id. The unique LSR Id of this LSR.

Hop Count. The Hop Count, if any, in the received attributes.

Path Vector. The Path Vector, if any, in the received attributes.

## Algorithm:

- CRa.1 Do RAttributes include Hop Count?  
If not, goto CRa.5.
- CRa.2 Does Hop Count exceed Max allowable hop count?  
If so, goto CRa.6.
- CRa.3 Do RAttributes include Path Vector?  
If not, goto CRa.5.
- CRa.4 Does Path Vector include LSR Id? OR Does length of Path  
Vector exceed Max allowable length?  
If so, goto CRa.6
- CRa.5 Return No Loop Detected.
- CRa.6 Is MsgType LabelMapping?  
If so, goto CRa.8. (See Note 1.)
- CRa.7 Execute procedure Send\_Notification (MsgSource, Loop  
Detected).
- CRa.8 Return Loop Detected.
- CRa.9 DONE.

## Note:

1. When the attributes being checked were received in a Label  
Mapping message, the LSR sends the Loop Detected notification in

a Status Code TLV in a Label Release message. (See Section "Receive Label Mapping".)

#### A.2.7. Prepare\_Label\_Request\_Attributes

##### Summary:

This procedure is used whenever a Label Request is to be sent to a Peer to compute the Hop Count and Path Vector, if any, to include in the message.

##### Parameters:

Peer. The LDP peer to which the message is to be sent.

FEC. The FEC for which a label request is to be sent.

RAttributes. The attributes this LSR associates with the LSP for FEC.

SAttributes. The attributes to be included in the Label Request message.

##### Additional Context:

LSR Id. The unique LSR Id of this LSR.

##### Algorithm:

- PRqA.1 Is Hop Count required for this Peer? (See Note 1.) OR Do RAttributes include a Hop Count? OR Is Loop Detection configured on LSR?  
If not, goto PRqA.14.
- PRqA.2 Is LSR ingress for FEC?  
If not, goto PRqA.6.
- PRqA.3 Include Hop Count of 1 in SAttributes.
- PRqA.4 Is Loop Detection configured on LSR?  
If not, goto PRqA.14.
- PRqA.5 Is LSR merge-capable?  
If so, goto PRqA.14.  
If not, goto PRqA.13.
- PRqA.6 Do RAttributes include a Hop Count?  
If not, goto PRqA.8.

- PRqA.7    Increment RAttributes Hop Count and copy the resulting Hop Count to SAttributes. (See Note 2.)  
          Goto PRqA.9.
- PRqA.8    Include Hop Count of unknown (0) in SAttributes.
- PRqA.9    Is Loop Detection configured on LSR?  
          If not, goto PRqA.14.
- PRqA.10   Do RAttributes have a Path Vector?  
          If so, goto PRqA.12.
- PRqA.11   Is LSR merge-capable?  
          If so, goto PRqA.14.  
          If not, goto PRqA.13.
- PRqA.12   Add LSR Id to beginning of Path Vector from RAttributes and copy the resulting Path Vector into SAttributes.  
          Goto PRqA.14.
- PRqA.13   Include Path Vector of length 1 containing LSR Id in SAttributes.
- PRqA.14   DONE.

Notes:

1. The link with Peer may require that Hop Count be included in Label Request messages; for example, see RFC 3035 [RFC3034] and RFC 3034 [RFC3034].
2. For hop count arithmetic, unknown + 1 = unknown.

A.2.8. Prepare\_Label\_Mapping\_Attributes

Summary:

This procedure is used whenever a Label Mapping is to be sent to a Peer to compute the Hop Count and Path Vector, if any, to include in the message.

Parameters:

- Peer. The LDP peer to which the message is to be sent.
- FEC. The FEC for which a label request is to be sent.

- RAttributes. The attributes this LSR associates with the LSP for FEC.
- SAttributes. The attributes to be included in the Label Mapping message.
- IsPropagating. The LSR is sending the Label Mapping message to propagate one received from the FEC next hop.
- PrevHopCount. The Hop Count, if any, this LSR associates with the LSP for the FEC.

Additional Context:

LSR Id. The unique LSR Id of this LSR.

Algorithm:

- PMpA.1 Do the RAttributes include any unknown TLVs?  
If not, goto PMpA.4.
- PMpA.2 Do the settings of the U- and F-bits require forwarding of these TLVs?  
If not, goto PMpA.4.
- PMpA.3 Copy the unknown TLVs in SAttributes.
- PMpA.4 Is Hop Count required for this Peer? (see Note 1.) OR Do RAttributes include a Hop Count? OR Is Loop Detection configured on LSR?  
If not, goto PMpA.24.
- PMpA.5 Is LSR egress for FEC?  
If not, goto PMpA.7.
- PMpA.6 Include Hop Count of 1 in SAttributes.  
Goto PMpA.24.
- PMpA.7 Do RAttributes have a Hop Count?  
If not, goto PMpA.11.
- PMpA.8 Is LSR a member of the edge set for an LSR domain whose LSRs do not perform TTL decrement AND Is Peer in that domain? (See Note 2.)  
If not, goto PMpA.10.
- PMpA.9 Include Hop Count of 1 in SAttributes.  
Goto PMpA.12.

- PMpA.10 Increment RAttributes Hop Count and copy the resulting Hop Count to SAttributes. (See Note 2.)  
Goto PMpA.12.
- PMpA.11 Include Hop Count of unknown (0) in SAttributes.
- PMpA.12 Is Loop Detection configured on LSR?  
If not, goto PMpA.24.
- PMpA.13 Do RAttributes have a Path Vector?  
If so, goto PMpA.22.
- PMpA.14 Is LSR propagating a received Label Mapping?  
If not, goto PMpA.23.
- PMpA.15 Does LSR support merging?  
If not, goto PMpA.17.
- PMpA.16 Has LSR previously sent a Label Mapping for FEC to Peer?  
If not, goto PMpA.23.
- PMpA.17 Do RAttributes include a Hop Count?  
If not, goto PMpA.24.
- PMpA.18 Is Hop Count in RAttributes unknown(0)?  
If so, goto PMpA.23.
- PMpA.19 Has LSR previously sent a Label Mapping for FEC to Peer?  
If not, goto PMpA.24.
- PMpA.20 Is Hop Count in RAttributes different from PrevHopCount?  
If not, goto PMpA.24.
- PMpA.21 Is the Hop Count in RAttributes > PrevHopCount? OR Is PrevHopCount unknown(0)?  
If not, goto PMpA.24.
- PMpA.22 Add LSR Id to beginning of Path Vector from RAttributes and copy the resulting Path Vector into SAttributes.  
Goto PMpA.24.
- PMpA.23 Include Path Vector of length 1 containing LSR Id in SAttributes.
- PMpA.24 DONE.

Notes:



1. The link with Peer may require that Hop Count be included in Label Mapping messages; for example, see RFC 3035 [RFC3034] and RFC 3034 [RFC3034].
2. If the LSR is at the edge of a cloud of LSRs that do not perform TTL-decrement and it is propagating the Label Mapping message upstream into the cloud, it sets the Hop Count to 1 so that Hop Count across the cloud is calculated properly. This ensures proper TTL management for packets forwarded across the part of the LSP that passes through the cloud.
3. For hop count arithmetic, unknown + 1 = unknown.

#### Acknowledgments

The editors of this document relies heavily on, and would like to thank, everyone that contributed to the development and improvement of the LDP Specification.

This document is produced as part of advancing the LDP specification to Internet Standard status. The predecessor (RFC 5036) was published as Draft Standard October 2007. It was produced by the MPLS Working Group of the IETF and was jointly authored by Loa Andersson, Bob Thomas and Ina Minei.

Since the Draft Standard version was published IETF has abandoned the 3 steps standards ladder. Now there is only proposed standard (PS) and Internet Standard (IS). This is part of the motivation to make the effort to bring the LDP specification to Internet Standard.

The LDP specification was originally published as RFC 3036 in January 2001. It was produced by the MPLS Working Group of the IETF and was jointly authored by Loa Andersson, Paul Doolan, Nancy Feldman, Andre Fredette, and Bob Thomas.

The ideas and text in RFC 3036 were collected from a number of sources. We would like to thank Rick Boivie, Ross Callon, Alex Conta, Eric Gray, Yoshihiro Ohba, Eric Rosen, Bernard Suter, Yakov Rekhter, and Arun Viswanathan for their input for RFC 3036.

The authors would like to thank Eric Gray, David Black, and Sam Hartman for their input to and review of RFC 5036. That input has been of great help also for the current document.

In addition, the authors would like to thank the members of the MPLS Working Group for their ideas and the support they have given to this document, and in particular, to Eric Rosen, Luca Martini, Markus

Jork, Mark Duffy, Vach Kompella, Kishore Tiruveedhula, Rama Ramakrishnan, Nick Weeds, Adrian Farrel, and Andy Malis.

Editor note - this section is still work in progress.

#### Authors' Addresses

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

Loa Andersson  
Huawei Technologies  
Email: loa@pi.nu

Nic Leymann  
Deutsche Telekom  
Email: N.Leymann@telekom.de

Ina Minei  
Google  
Email: inaminei@google.com

Kamran Raza  
Cisco Systems, Inc.  
Email: skraza@cisco.com

MPLS WG  
Internet-Draft  
Intended status: Standards Track  
Expires: March 11, 2016

K. Kompella  
Juniper Networks, Inc.  
L. Contreras  
Telefonica I+D  
September 8, 2015

Resilient MPLS Rings  
draft-kompella-mpls-rmr-02

Abstract

This document describes the use of the MPLS control and data planes on ring topologies. It describes the special nature of rings, and proceeds to show how MPLS can be effectively used in such topologies. It describes how MPLS rings are configured, auto-discovered and signaled, as well as how the data plane works. Companion documents describe the details of discovery and signaling for specific protocols.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 11, 2016.

Copyright Notice

Copyright (c) 2015 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 (<http://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. Definitions . . . . .	3
2. Motivation . . . . .	4
3. Theory of Operation . . . . .	5
3.1. Provisioning . . . . .	5
3.2. Ring Nodes . . . . .	5
3.3. Ring Links and Directions . . . . .	6
3.3.1. Bypass Links . . . . .	6
3.4. Ring LSPs . . . . .	7
3.5. Installing Primary LFIB Entries . . . . .	7
3.6. Installing FRR LFIB Entries . . . . .	7
3.7. Protection . . . . .	8
4. Autodiscovery . . . . .	9
4.1. Overview . . . . .	9
4.2. Ring Announcement Phase . . . . .	10
4.3. Mastership Phase . . . . .	11
4.4. Ring Identification Phase . . . . .	11
4.5. Ring Changes . . . . .	11
5. Ring Signaling . . . . .	12
6. Ring OAM . . . . .	12
7. Security Considerations . . . . .	12
8. Acknowledgments . . . . .	12
9. IANA Considerations . . . . .	13
10. References . . . . .	13
10.1. Normative References . . . . .	13
10.2. Informative References . . . . .	13
Authors' Addresses . . . . .	13

## 1. Introduction

Rings are a very common topology in transport networks. A ring is the simplest topology offering link and node resilience. Rings are nearly ubiquitous in access and aggregation networks. As MPLS increases its presence in such networks, and takes on a greater role in transport, it is imperative that MPLS handles rings well; this is not the case today.

This document describes the special nature of rings, and the special needs of MPLS on rings. It then shows how these needs can be met in several ways, some of which involve extensions to protocols such as IS-IS [RFC5305], OSPF[RFC3630], RSVP-TE [RFC3209] and LDP [RFC5036].

The intent of this document is to handle rings that "occur naturally". Many access and aggregation networks in metros have their start as a simple ring. They may then grow into more complex topologies, for example, by adding parallel links to the ring, or by adding "bypass" links. The goal here is to discover these rings (with some guidance), and run MPLS over them efficiently. The intent is not to construct rings in a mesh network, and use those for protection.

### 1.1. Definitions

A (directed) graph  $G = (V, E)$  consists of a set of vertices (or nodes)  $V$  and a set of edges (or links)  $E$ . An edge is an ordered pair of nodes  $(a, b)$ , where  $a$  and  $b$  are in  $V$ . (In this document, the terms node and link will be used instead of vertex and edge.)

A ring is a subgraph of  $G$ . A ring consists of a subset of  $n$  nodes  $\{R_i, 0 \leq i < n\}$  of  $V$ . The directed edges  $\{(R_i, R_{i+1}) \text{ and } (R_{i+1}, R_i), 0 \leq i < n-1\}$  must be a subset of  $E$  (note that index arithmetic is done modulo  $n$ ). We define the direction from node  $R_i$  to  $R_{i+1}$  as "clockwise" (CW) and the reverse direction as "anticlockwise" (AC). As there may be several rings in a graph, we number each ring with a distinct ring ID RID.

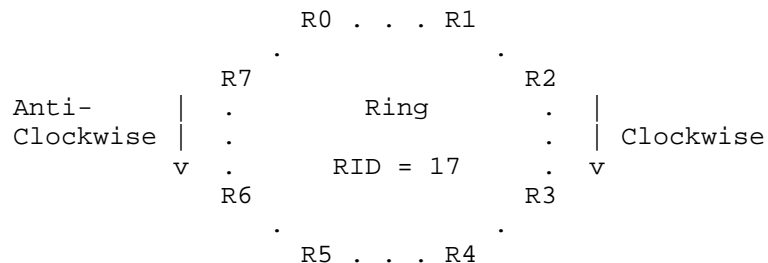


Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

Ring ID (RID): A non-zero number that identifies a ring; this is unique in some scope of a Service Provider's network. A node may belong to multiple rings.

Ring node: A member of a ring. Note that a device may belong to several rings.

Node index: A logical numbering of nodes in a ring, from zero upto one less than the ring size. Used purely for exposition in this document.

Ring master: The ring master initiates the ring identification process. Mastership is indicated in the IGP by a two-bit field.

Ring neighbors: Nodes whose indices differ by one (modulo ring size).

Ring links: Links that connect ring neighbors.

Bypass links: Links that connect non-neighboring ring nodes.

Ring direction: A two-bit field in the IGP indicating the direction of a link. The choices are:

UN: 00 undefined link

CW: 01 clockwise ring link

AC: 10 anticlockwise ring link

BY: 11 bypass link

Ring Identification: The process of discovering ring nodes, ring links, link directions, and bypass links.

The following notation is used for ring LSPs:

R<sub>k</sub>: A ring node with index k. R<sub>k</sub> has AC neighbor R<sub>(k-1)</sub> and CW neighbor R<sub>(k+1)</sub>.

RL<sub>k</sub>: A (unicast) Ring LSP anchored on node R<sub>k</sub>.

CL<sub>jk</sub> (AL<sub>jk</sub>): A label allocated by R<sub>j</sub> for RL<sub>k</sub> in the CW (AC) direction.

P<sub>jk</sub> (Q<sub>jk</sub>): A Path (Resv) message sent by R<sub>j</sub> for RL<sub>k</sub>.

## 2. Motivation

A ring is the simplest topology that offers resilience. This is perhaps the main reason to lay out fiber in a ring. Thus, effective mechanisms for fast failover on rings are needed. Furthermore, there

are large numbers of rings. Thus, configuration of rings needs to be as simple as possible. Finally, bandwidth management on access rings is very important, as bandwidth is generally quite constrained here.

The goals of this document are to present mechanisms for improved MPLS-based resilience in ring networks (using ideas that are reminiscent of Bidirectional Line Switched Rings), for automatic bring-up of LSPs, better bandwidth management and for auto-hierarchy. These goals can be achieved using extensions to existing IGP and MPLS signaling protocols, using central provisioning, or in other ways.

### 3. Theory of Operation

Say a ring has ring ID RID. The ring is provisioned by choosing one or more ring masters for the ring and assigning them the RID. Other nodes in the ring may also be assigned this RID, or may be configured as "promiscuous". Ring discovery then kicks in. When each ring node knows its CW and AC ring neighbors and its ring links, and all bypass links have been identified, ring identification is complete.

Once ring identification is complete, each node signals one or more ring LSPs  $RL_i$ .  $RL_i$ , anchored on node  $R_i$ , consists of two counter-rotating unicast LSPs that start and end at  $R_i$ . A ring LSP is "multipoint": any node  $R_j$  can use  $RL_i$  to send traffic to  $R_i$ ; this can be in either the CW or AC directions, or both (i.e., load balanced). Both of these counter-rotating LSPs are "active"; the choice of direction to send traffic to  $R_i$  is determined by policy at the node where traffic is injected into the ring. The default is to send traffic along the shortest path. Bidirectional connectivity between nodes  $R_i$  and  $R_j$  is achieved by using two different ring LSPs:  $R_i$  uses  $RL_j$  to reach  $R_j$ , and  $R_j$  uses  $RL_i$  to reach  $R_i$ .

#### 3.1. Provisioning

The goal here is to provision rings with the absolute minimum configuration. The exposition below aims to achieve that using auto-discovery via a link-state IGP (see Section 4). Of course, auto-discovery can be overridden by configuration. For example, a link that would otherwise be classified by auto-discovery as a ring link might be configured not to be used for ring LSPs.

#### 3.2. Ring Nodes

Ring nodes have a loopback address, and run a link-state IGP and an MPLS signaling protocol. To provision a node as a ring node for ring RID, the node is simply assigned that RID. A node may be part of several rings, and thus may be assigned several ring IDs.

To simplify ring provisioning even further, a node N may be made "promiscuous" by being assigned an RID of 0. A promiscuous node listens to RIDs in its IGP neighbors' link-state updates. If N hears a non-zero RID from a neighbor, it joins that ring by taking on that RID. However, if N hears more than one non-zero RID from its neighbors, N remains in promiscuous mode. In many situations, the use of promiscuous mode means that only one or two nodes in the ring needs to be provisioned; everything else is auto-discovered.

A ring node indicates in its IGP updates the ring LSP signaling protocols it supports. This can be LDP and/or RSVP-TE. Ideally, each node should support both.

### 3.3. Ring Links and Directions

Ring links must be MPLS-capable. They are by default unnumbered, point-to-point (from the IGP point of view) and "auto-bundled". The last attribute means that parallel links between ring neighbors are considered as a single link, without the need for explicit configuration for bundling (such as a Link Aggregation Group). Note that each component may be advertised separately in the IGP; however, signaling messages and labels across one component link apply to all components. Parallel links between a pair of ring nodes is often the result of having multiple lambdas or fibers between those nodes. RMR is primarily intended for operation at the packet layer; however, parallel links at the lambda or fiber layer result in parallel links at the packet layer.

A ring link is not provisioned as belonging to the ring; it is discovered to belong to ring RID if both its adjacent nodes belong to RID. A ring link's direction (CW or AC) is also discovered; this process is initiated by the ring's ring master. Note that the above two attributes can be overridden by provisioning if needed; it is then up to the provisioning system to maintain consistency across the ring.

#### 3.3.1. Bypass Links

Bypass links are discovered once ring nodes, ring links and directions have been established. As defined earlier, bypass links are links joining non-neighbor ring nodes; often, this may be the result of optically bypassing ring nodes. The use of bypass links will be described in a future version of this document.



### 3.4. Ring LSPs

Ring LSPs are not provisioned. Once a ring node  $R_i$  knows its RID, its ring links and directions, it kicks off ring LSP signaling automatically.  $R_i$  allocates CW and AC labels for each ring LSP  $RL_k$ .  $R_i$  also initiates the creation of  $RL_i$ . As the signaling propagates around the ring, CW and AC labels are exchanged. When  $R_i$  receives CW and AC labels for  $RL_k$  from its ring neighbors, primary and fast reroute (FRR) paths for  $RL_k$  are installed at  $R_i$ . More details are given in Section 5.

For RSVP-TE LSPs, bandwidths may be signaled in both directions. However, these are not provisioned either; rather, one does "reverse call admission control". When a service needs to use an LSP, the ring node where the traffic enters the ring attempts to increase the bandwidth on the LSP to the egress. If successful, the service is admitted to the ring.

### 3.5. Installing Primary LFIB Entries

In setting up  $RL_k$ , a node  $R_j$  sends out two labels:  $CL_{jk}$  to  $R_{j-1}$  and  $AL_{jk}$  to  $R_{j+1}$ .  $R_j$  also receives two labels:  $CL_{j+1,k}$  from  $R_{j+1}$ , and  $AL_{j-1,k}$  from  $R_{j-1}$ .  $R_j$  can now set up the forwarding entries for  $RL_k$ . In the CW direction,  $R_j$  swaps incoming label  $CL_{jk}$  with  $CL_{j+1,k}$  with next hop  $R_{j+1}$ ; these allow  $R_j$  to act as LSR for  $RL_k$ .  $R_j$  also installs an LFIB entry to push  $CL_{j+1,k}$  with next hop  $R_{j+1}$  to act as ingress for  $RL_k$ . Similarly, in the AC direction,  $R_j$  swaps incoming label  $AL_{jk}$  with  $AL_{j-1,k}$  with next hop  $R_{j-1}$  (as LSR), and an entry to push  $AL_{j-1,k}$  with next hop  $R_{j-1}$  (as ingress).

Clearly,  $R_k$  does not act as ingress for its own LSPs. However, if these LSPs use UHP, then  $R_k$  installs LFIB entries to pop  $CL_{k,k}$  for packets received from  $R_{k-1}$  and to pop  $AL_{k,k}$  for packets received from  $R_{k+1}$ .

### 3.6. Installing FRR LFIB Entries

At the same time that  $R_j$  sets up its primary CW and AC LFIB entries, it can also set up the protection forwarding entries for  $RL_k$ . In the CW direction,  $R_j$  sets up an FRR LFIB entry to swap incoming label  $CL_{jk}$  with  $AL_{j-1,k}$  with next hop  $R_{j-1}$ . In the AC direction,  $R_j$  sets up an FRR LFIB entry to swap incoming label  $AL_{jk}$  with  $CL_{j+1,k}$  with next hop  $R_{j+1}$ . Again,  $R_k$  does not install FRR LFIB entries in this manner.

### 3.7. Protection

In this scheme, there are no protection LSPs as such -- no node or link bypasses, no standby LSPs, no detours, and no LFA-type protection. Protection is via the "other" direction around the ring, which is why ring LSPs are in counter-rotating pairs. Protection works in the same way for link, node and ring LSP failures.

If a node  $R_j$  detects a failure from  $R_{j+1}$  -- either all links to  $R_{j+1}$  fail, or  $R_{j+1}$  itself fails,  $R_j$  switches traffic on all CW ring LSPs to the AC direction using the FRR LFIB entries. If the failure is specific to a single ring LSP,  $R_j$  switches traffic just for that LSP. In either case, this switchover can be very fast, as the FRR LFIB entries can be preprogrammed. Fast detection and fast switchover lead to minimal traffic loss.

$R_j$  then sends an indication to  $R_{j-1}$  that the CW direction is not working, so that  $R_{j-1}$  can similarly switch traffic to the AC direction. For RSVP-TE, this indication can be a PathErr or a Notify; other signaling protocols have similar indications. These indications propagate AC until each traffic source on the ring AC of the failure uses the AC direction. Thus, within a short period, traffic will be flowing in the optimal path, given that there is a failure on the ring. This contrasts with (say) bypass protection, where until the ingress recomputes a new path, traffic will be suboptimal.

Note that the failure of a node or a link will not necessarily affect all ring LSPs. Thus, it is important to identify the affected LSPs (and switch them), but to leave the rest alone.

One point to note is that when a ring node, say  $R_j$ , fails,  $RL_j$  is clearly unusable. However, the above protection scheme will cause a traffic loop:  $R_{j-1}$  detects a failure CW, and protects by sending CW traffic on  $RL_j$  back all the way to  $R_{j+1}$ , which in turn sends traffic to  $R_{j-1}$ , etc. There are three proposals to avoid this:

1. Each ring node acting as ingress sends traffic with a TTL of at most  $2*n$ , where  $n$  is the number of nodes in the ring.
2. A ring node sends protected traffic (i.e., traffic switched from CW to AC or vice versa) with TTL just large enough to reach the egress.
3. A ring node sends protected traffic with a special purpose label below the ring LSP label. A protecting node first checks for the presence of this label; if present, it means that the traffic is looping and MUST be dropped.

It is recommended that (2) be implemented. The other methods are optional.

#### 4. Autodiscovery

##### 4.1. Overview

Auto-discovery proceeds in three phases. The first phase is the announcement phase. The second phase is the mastership phase. The third phase is the ring identification phase.

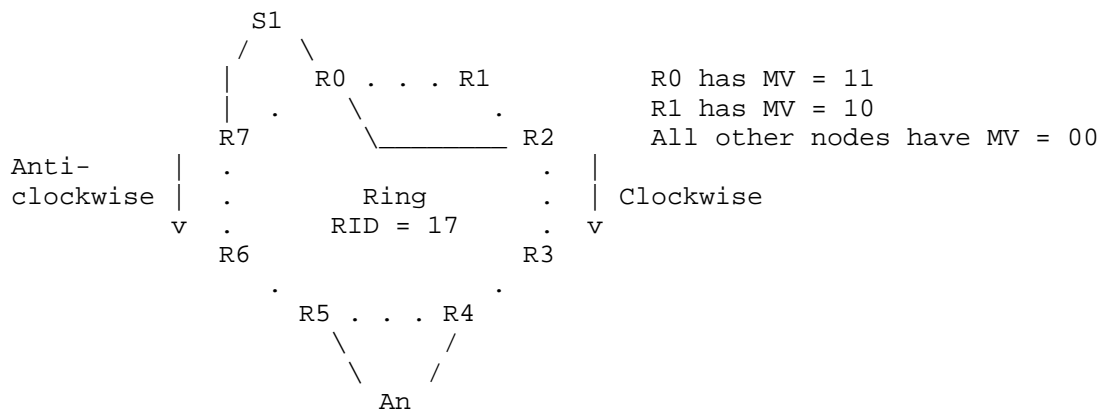
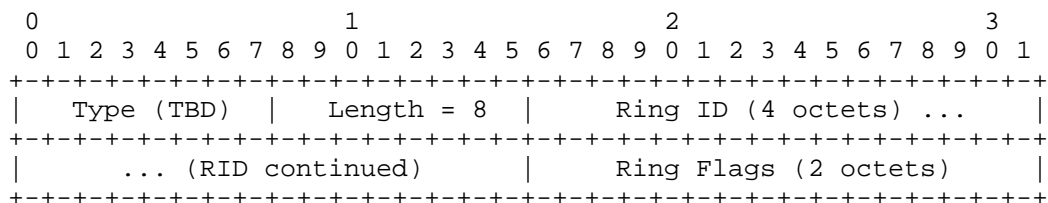


Figure 2: Ring with non-ring nodes and links

In what follows, we refer to a ring Type-Length-Value (TLV). This is a new TLV that contains an RID and associated flags. A ring link TLV is a ring TLV that appears as a sub-TLV of a traffic engineering TLV (TE TLV) of each link that is identified as a ring link or a bypass link. For IS-IS, the TE TLV is the extended reachability TLV; for OSPF, it is the Link TLV in the opaque TE LSA. A ring node TLV is a ring TLV that appears as a sub-TLV of a "node TLV" once for each ring this node is participating in. In IS-IS, the node TLV is the Router ID TLV; in OSPF, it is a new top-level TLV of the TE LSA. The ring direction field is ignored in ring node TLVs.



IS-IS Ring TLV Format

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
|          Type (TBD)                |          Length = 12              |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
|          Ring ID (4 octets)         |                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Ring Flags (2 octets)      |          Pad (2 octets)          |
+-----+-----+-----+-----+-----+-----+-----+-----+
Pad is set to zero when sending and ignored on receipt.

```

#### OSPF Ring TLV Format

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+-----+-----+-----+-----+-----+-----+-----+
|MV|RD|SP|OP|M|          MBZ          |
+-----+-----+-----+-----+-----+-----+-----+
MV: Mastership Value
RD: Ring Direction
SP: Signaling Protocols (10 = RSVP-TE; 01 = LDP)
OP: OAM Protocols (10 = BFD; 01 = EFM)
M : Elected Master (0 = no, 1 = yes)

```

#### Ring Flags Format

### 4.2. Ring Announcement Phase

Each node participating in an MPLS ring is assigned an RID; in the example, RID = 17. A node is also provisioned with a mastership value. Each node advertises a ring node TLV for each ring it is participating in, along with the associated flags. It then starts timer T1.

A node in promiscuous mode doesn't advertise any ring node TLVs. If it hears exactly one non-zero RID from its IGP neighbors, it joins that ring, and sends one ring node TLV with that RID. If it hears more than one RID from its IGP neighbors, it doesn't join any rings, and withdraws any ring node TLVs it may have advertised.

The announcement phase allows a ring node to discover other ring nodes in the same ring so that a ring master can be elected and ring links be identified.

#### 4.3. Mastership Phase

When timer T1 fires, a node enters the mastership phase. In this phase, each ring node N starts timer T2 and checks if it is master. If it is the node with the lowest loopback address of all nodes with the highest mastership values, N declares itself master by readvertising its ring node TLV with the M bit set.

When timer T2 fires, each node examines the ring node TLVs from all other nodes in the ring to identify the ring master. There should be exactly one; if not, each node restarts timer T2 and tries again. The nodes that set their M bit should be extra careful in advertising their M bit in subsequent tries.

#### 4.4. Ring Identification Phase

When there is exactly one ring master M, M enters the Ring Identification Phase. M indicates that it has successfully completed this phase by advertising ring link TLVs. This is the trigger for M's CW neighbor to enter the Ring Identification Phase. This phase passes CW until all ring nodes have completed ring identification.

In the Ring Identification Phase, a node X that has two or more IGP neighbors that belong to the ring picks one of them to be its CW ring neighbor. If X is the ring master, it also picks a node as its AC ring neighbor. If there are exactly two such nodes, this step is trivial. If not, X computes a ring that includes all nodes that have completed the Ring Identification Phase (as seen by their ring link TLVs) and further contains the maximal number of nodes that belong to the ring. Based on that, X picks a CW neighbor and inserts ring link TLVs with ring direction CW for each link to its CW neighbor; X also inserts a ring link TLV with direction AC for each link to its AC neighbor. Then, X determines its bypass links. These are links connected to ring nodes that are not ring neighbors. X advertises ring link TLVs for bypass links by setting the link direction to "bypass link".

#### 4.5. Ring Changes

The main changes to a ring are:

- ring link addition;
- ring link deletion;
- ring node addition; and
- ring node deletion.

The main goal of handling ring changes is (as much as possible) not to perturb existing ring operation. Thus, if the ring master hasn't changed, all of the above changes should be local to the point of change. Link adds just update the IGP; signaling should take advantage of the new capacity as soon as it learns. Link deletions in the case of parallel links also show up as a change in capacity (until the last link in the bundle is removed.)

The removal of the last ring link between two nodes, or the removal of a ring node is an event that triggers protection switching. In a simple ring, the result is a broken ring. However, if a ring has bypass links, then it may be able to converge to a smaller ring with protection. Details of this process will be given in a future version.

The addition of a new ring node can also be handled incrementally. Again, the details of this process will be given in a future version.

## 5. Ring Signaling

A future version of this document will specify protocol-independent details about ring LSP signaling.

## 6. Ring OAM

Each ring node should advertise in its ring node TLV the OAM protocols it supports. Each ring node is expected to run a link-level OAM over each ring and bypass link. This should be an OAM protocol that both neighbors agree on. The default hello time is 3.3 millisecond.

Each ring node also sends OAM messages over each direction of its ring LSP. This is a multi-hop OAM to check LSP liveness; typically, BFD would be used for this. The node chooses the hello interval; the default is once a second.

## 7. Security Considerations

It is not anticipated that either the notion of MPLS rings or the extensions to various protocols to support them will cause new security loopholes. As this document is updated, this section will also be updated.

## 8. Acknowledgments

Many thanks to Pierre Bichon whose exemplar of self-organizing networks and whose urging for ever simpler provisioning led to the notion of promiscuous nodes.

## 9. IANA Considerations

There are no requests as yet to IANA for this document.

## 10. References

### 10.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.

### 10.2. Informative References

- [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, <<http://www.rfc-editor.org/info/rfc3209>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<http://www.rfc-editor.org/info/rfc3630>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<http://www.rfc-editor.org/info/rfc5036>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<http://www.rfc-editor.org/info/rfc5305>>.

## Authors' Addresses

Kireeti Kompella  
Juniper Networks, Inc.  
1194 N. Mathilda Avenue  
Sunnyvale, CA 94089  
USA

Email: [kireeti.kompella@gmail.com](mailto:kireeti.kompella@gmail.com)

Luis M. Contreras  
Telefonica I+D  
Ronda de la Comunicacion  
Sur-3 building, 3rd floor  
Madrid 28050  
Spain

Email: [luismiguel.contrerasmurillo@telefonica.com](mailto:luismiguel.contrerasmurillo@telefonica.com)  
URI: <http://people.tid.es/LuisM.Contreras/>



MPLS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 8, 2017

M. Taillon  
T. Saad, Ed.  
R. Gandhi  
Cisco Systems, Inc.  
A. Deshmukh  
M. Jork  
V. Beeram  
Juniper Networks  
March 07, 2017

RSVP-TE Summary Fast Reroute Extensions for LSP Tunnels  
draft-mtaillon-mpls-summary-frr-rsvpte-05

Abstract

This document defines Resource Reservation Protocol (RSVP) Traffic-Engineering (TE) signaling extensions that reduce the amount of RSVP signaling required for Fast Reroute (FRR) procedures and subsequently improve the scalability of the RSVP-TE signaling when undergoing FRR convergence after a link or node failure. Such extensions allow the RSVP message exchange between the Point of Local Repair (PLR) and the Merge Point (MP) to be independent of the number of protected Label Switched Paths (LSPs) traversing between them when facility bypass FRR protection is used. The signaling extensions are fully backwards compatible with nodes that do not support them.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 8, 2017.

## 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 (<http://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. Key Word Definitions . . . . .	3
2.2. Terminology . . . . .	3
3. Summary FRR Signaling Procedures . . . . .	3
3.1. Signaling Procedures Prior to Failure . . . . .	4
3.1.1. Extended ASSOCIATION Object . . . . .	5
3.1.2. PLR Summary FRR Signaling Procedure . . . . .	9
3.1.3. MP Summary FRR Signaling Procedure . . . . .	9
3.2. Signaling Procedures Post Failure . . . . .	10
3.2.1. SUMMARY_FRR_BYPASS Object . . . . .	10
3.2.2. PLR Summary FRR Signaling Procedure . . . . .	12
3.2.3. MP Summary FRR Signaling Procedure . . . . .	12
3.3. Refreshing Summary FRR Active LSPs . . . . .	13
4. Compatibility . . . . .	13
5. Security Considerations . . . . .	13
6. IANA Considerations . . . . .	13
7. Acknowledgments . . . . .	14
8. Contributors . . . . .	14
9. Normative References . . . . .	14
Authors' Addresses . . . . .	15

## 1. Introduction

The Fast Reroute (FRR) procedures defined in [RFC4090] describe the mechanisms for the Point of Local Repair (PLR) to reroute traffic and signaling of a protected RSVP-TE LSP onto the bypass tunnel in the event of a TE link or node failure. Such signaling procedures are performed individually for each affected protected LSP. This may eventually lead to control plane scalability and latency issues under limited (memory and CPU processing) resources after a failure that

affects a large number of protected LSPs traversing the same PLR and Merge Point (MP) nodes.

For example, in a large RSVP-TE LSPs scale deployment, a single LSR acting as a PLR node may host tens of thousands of protected RSVP-TE LSPs egressing the same link, and also act as a MP node for similar number of LSPs ingressing the same link. In the event of the failure of the link or neighbor node, the RSVP-TE control plane of the node when acting as PLR becomes busy rerouting protected LSPs signaling over the bypass tunnel(s) in one direction, and when acting as an MP node becomes busy merging RSVP states from signaling received over bypass tunnels for LSP(s) in the reverse direction. Subsequently, the head-end LER(s) that are notified of the local repair at downstream LSR will attempt to (re)converge affected RSVP-TE LSPs onto newly computed paths - possibly traversing the same previously affected LSR(s). As a result, the RSVP-TE control plane at the PLR and MP becomes overwhelmed by the amount of FRR RSVP-TE processing overhead following the link or node failure, and the competing other control plane protocol(s) (e.g. the IGP) that undergo their convergence at the same time.

The extensions defined in this document enable a MP node to become aware of the PLR node's bypass tunnel assignment group and allow FRR procedures between PLR node and MP node to be signaled and processed on groups of LSPs. Further, the MESSAGE\_ID for the rerouted PATH and RESV states are exchanged a priori to the fault such that Summary Refresh procedures defined in [RFC2961] can continue to be used to refresh the rerouted state(s) after FRR has occurred.

## 2. Conventions Used in This Document

### 2.1. Key Word Definitions

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 BCP 14, RFC 2119 [RFC2119].

### 2.2. Terminology

The reader is assumed to be familiar with the terminology in [RFC3209] and [RFC4090].

## 3. Summary FRR Signaling Procedures

The RSVP ASSOCIATION object is defined in [RFC4872] as a means to associate LSPs with each other. For example, in the context of GMPLS-controlled LSP(s), the object is used to associate recovery

LSPs with the LSP they are protecting. The Extended ASSOCIATION object is introduced in [RFC6780] to expand on the possible usage of the ASSOCIATION object and generalize the definition of the Association ID field.

This document proposes the use of the Extended ASSOCIATION object to carry the Summary FRR information and associate the protected LSP(s) with the bypass tunnel that protects them. To this extent, a new Association Type for the Extended ASSOCIATION object, and a new Association ID are proposed in this draft to describe the Bypass Summary FRR (B-SFRR) association.

The PLR creates and manages the Summary FRR LSP groups (Bypass\_Group\_Identifiers) and shares them with the MP via signaling. Protected LSPs sharing the same egress link and bypass assignment are grouped together and are assigned the same group. The MP maintains the PLR group assignments learned via signaling, and acknowledges the group assignments via signaling. Once the PLR receives the acknowledgment, FRR signaling can proceed as group based.

The PLR node that supports Summary FRR procedures adds the Extended ASSOCIATION object with Bypass Summary FRR Association Type - referred to thereon in this document as B-SFRR Extended ASSOCIATION object- in the RSVP Path message of the protected LSP to inform the MP of the PLR's assigned bypass tunnel, Summary FRR Bypass\_Group\_Identifier, and the MESSAGE\_ID object that the PLR will use to refresh the protected LSP PATH state after FRR occurs.

The MP node that supports Summary FRR procedures adds the B-SFRR Extended ASSOCIATION object in a RSVP Resv message of the protected LSP to acknowledge the PLR's bypass tunnel assignment, and provide the MESSAGE\_ID object that the MP node will use to refresh the protected LSP RESV state after FRR occurs.

This document also defines a new RSVP FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object that is sent within the RSVP Path message of a bypass LSP to inform the MP node that one or more groups of protected LSPs that are being protected by the bypass tunnel are being rerouted i.e. signaling is rerouted over the bypass tunnel.

### 3.1. Signaling Procedures Prior to Failure

Before Summary FRR procedures can be used, a handshake MUST be completed between the PLR and MP. This handshake is performed using B-SFRR Extended ASSOCIATION object that is carried in both the RSVP Path and Resv messages of the protected LSP.

### 3.1.1.1. Extended ASSOCIATION Object

The B-SFRR Extended ASSOCIATION object is populated using the rules defined below to associate the Summary FRR enabled protected LSP with the bypass LSP that is protecting it.

The Association Type, Association ID, and Association Source MUST be set as defined in [RFC4872] for the ASSOCIATION Object. More specifically:

#### Association Source:

The Association Source is set to an address selected by the node that originates the association. For Bypass Summary FRR association it is set to an address of the PLR node.

#### Association Type:

The Association Type is set to indicate the Bypass Summary FRR association. A new Association Type is defined as follows:

Value	Type
-----	-----
(TBD-1)	Bypass Summary FRR Association (B-SFRR)

#### Extended Association ID:

The Extended Association ID is populated by the node originating the association -- i.e. the PLR for the Bypass Summary FRR association. The rules to populate the Extended Association ID in this case is described below.

### 3.1.1.1.1. IPv4 Extended Association ID

The IPv4 Extended Association ID for Summary FRR bypass assignment has the following format:

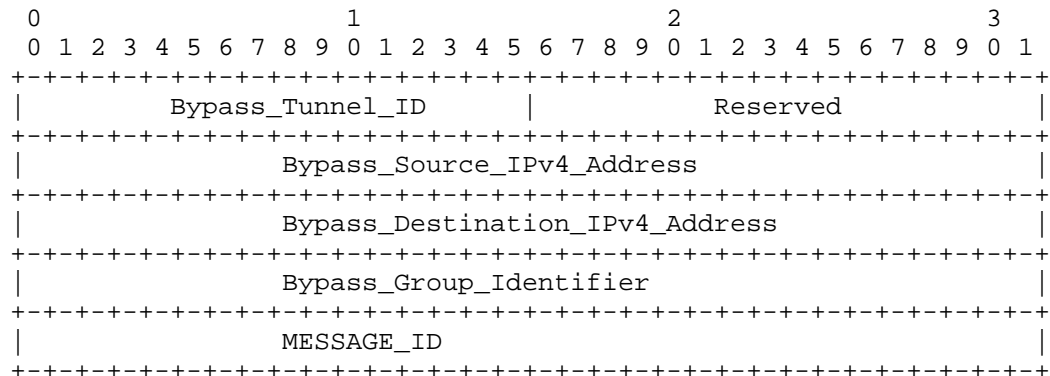


Figure 1: The IPv4 Extended Association ID field

Bypass\_Tunnel\_ID: 16 bits

The bypass tunnel identifier.

Reserved: 16 bits

Reserved for future use.

Bypass\_Source\_IPv4\_Address: 32 bits

The bypass tunnel source IPV4 address.

Bypass\_Destination\_IPv4\_Address: 32 bits

The bypass tunnel destination IPV4 address.

Bypass\_Group\_Identifier: 32 bits

The bypass tunnel group identifier.

MESSAGE\_ID

A MESSAGE\_ID object as defined by [RFC2961].

### 3.1.1.2. IPv6 Extended Association ID

The IPv6 Extended Association ID field for the Summary FRR information has the following format:

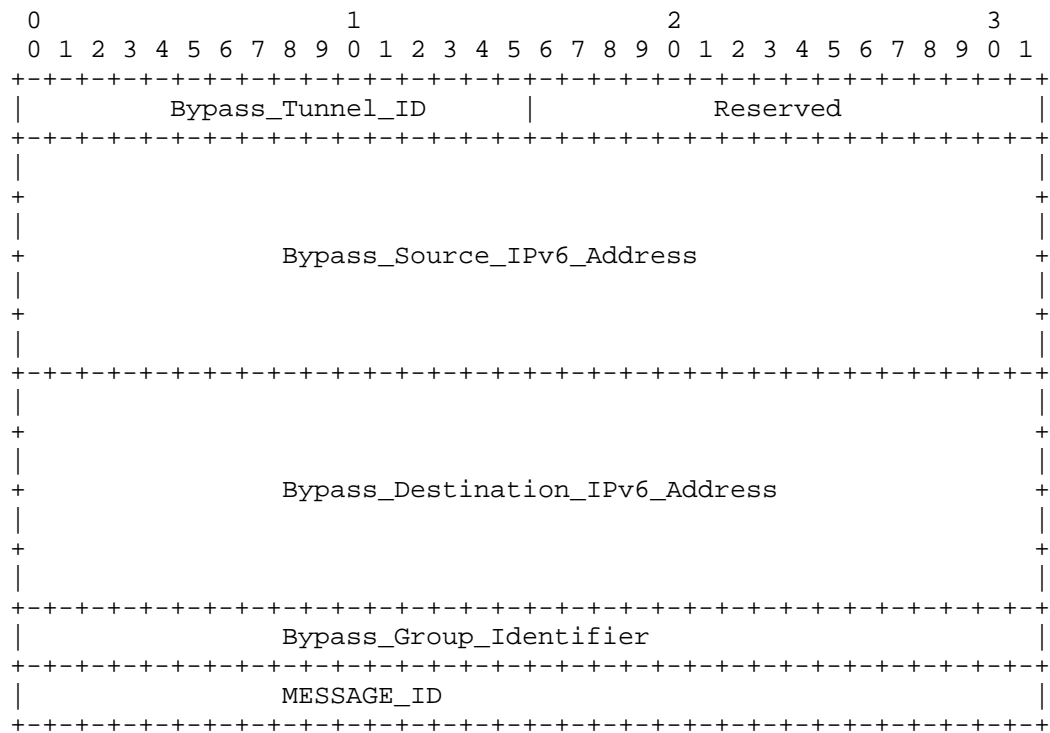


Figure 2: The IPv6 Extended Association ID field

Bypass\_Tunnel\_ID: 16 bits

The bypass tunnel identifier.

Reserved: 16 bits

Reserved for future use.

Bypass\_Source\_IPv6\_Address: 128 bits

The bypass tunnel source IPV6 address.

Bypass\_Destination\_IPv6\_Address: 128 bits

The bypass tunnel destination IPV6 address.

Bypass\_Group\_Identifier: 32 bits

The bypass tunnel group identifier.

MESSAGE\_ID

A MESSAGE\_ID object as defined by [RFC2961].

The PLR assigns a bypass tunnel and Bypass\_Group\_Identifier for each protected LSP. The same Bypass\_Group\_Identifier is used for the set of protected LSPs that share the same bypass tunnel and traverse the same egress link and are not already rerouted. The PLR also generates a MESSAGE\_ID object (flags SHOULD be clear, Epoch and Message\_Identifier MUST be set according to [RFC2961]).

The PLR MUST generate a new Message\_Identifier each time the contents of the B-SFRR Extended ASSOCIATION object change; for example, when PLR node changes the bypass tunnel assignment.

The PLR node notifies the MP node of the bypass tunnel assignment via adding a B-SFRR Extended ASSOCIATION object in the RSVP Path message for the protected LSP using procedures described in Section 3.2.

The MP node acknowledges the PLR node assignment by signaling the B-SFRR Extended Association object within the RSVP Resv message of the protected LSP. With exception of the MESSAGE\_ID objects, all other fields of the received B-SFRR Extended ASSOCIATION object in the RSVP Path message are copied into the B-SFRR Extended ASSOCIATION object to be added in the Resv message. The MESSAGE\_ID object is set according to [RFC2961] with the Flags being clear. A new Message\_Identifier MUST be used to acknowledge an updated PLR assignment.



The PLR considers the protected LSP as Summary FRR capable only if the B-SFRR Extended ASSOCIATION objects sent in the RSVP Path message and the one received in the RSVP Resv message (with exception of the MESSAGE\_ID) match. If it does not match, or if B-SFRR Extended Association object is absent in a subsequent refresh, the PLR node MUST consider the protected LSP as not Summary FRR capable.

### 3.1.2. PLR Summary FRR Signaling Procedure

The B-SFRR Extended ASSOCIATION object is added by each PLR in the RSVP Path message of the protected LSP to record the bypass tunnel assignment. This object is updated every time the PLR updates the bypass tunnel assignment (which triggers an RSVP Path change message).

Upon receiving an RSVP Resv message with B-SFRR Extended ASSOCIATION object, the PLR node checks if the expected subobjects in the B-SFRR Extended ASSOCIATION ID are present. If present, the PLR determines if the MP has acknowledged the current PLR assignment.

To be a valid acknowledgement, the received B-SFRR Extended ASSOCIATION object contents within the RSVP Resv message of the protected LSP MUST match the latest B-SFRR Extended ASSOCIATION object contents that the PLR node had sent within the RSVP Path message (with exception of the MESSAGE\_ID).

Note, when forwarding an RSVP Resv message upstream, the PLR node SHOULD remove any/all B-SFRR Extended ASSOCIATION objects whose Association Source matches the PLR node address.

### 3.1.3. MP Summary FRR Signaling Procedure

Upon receiving an RSVP Path message with an B-SFRR Extended ASSOCIATION object, the MP node processes all (there may be multiple PLRs for a single MP) B-SFRR Extended ASSOCIATION objects that have the MP node address as Bypass Destination address in the Association ID.

The MP node first ensures the existence of the bypass tunnel and that the Bypass\_Group\_Identifier is not already FRR active. That is, an LSP cannot join a group that is already FRR rerouted.

The MP node builds a mirrored Summary FRR Group database per PLR, which is determined using the Bypass\_Source\_Address field. The MESSAGE\_ID is extracted and recorded for the protected LSP PATH state. The MP node signals a B-SFRR Extended Association object within the RSVP Resv message of the protected LSP. With exception of the MESSAGE\_ID objects, all other fields of the received B-SFRR

Extended ASSOCIATION object in the RSVP Path message are copied into the B-SFRR Extended ASSOCIATION object to be added in the Resv message. The MESSAGE\_ID object is set according to [RFC2961] with the Flags being clear.

Note, an MP may receive more than one RSVP Path message with the B-SFRR Extended ASSOCIATION object from different upstream PLR node(s). In this case, the MP node is expected to save all the received MESSAGE\_IDs from the different upstream PLR node(s). After a failure, the MP node determines and activates the associated Summary Refresh ID to use once it receives and processes the RSVP Path message with FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object over the bypass LSP from the PLR.

When forwarding an RSVP Path message downstream, the MP SHOULD remove any/all B-SFRR Extended ASSOCIATION object(s) whose Association ID contains Bypass\_Destination\_Address matching the MP node address.

### 3.2. Signaling Procedures Post Failure

Upon detection of the fault (egress link or node failure) the PLR first performs the object modification procedures described by Section 6.4.3 of [RFC4090] for all affected protected LSPs. For Summary FRR LSPs assigned to the same bypass tunnel a common RSVP\_HOP and SENDER\_TEMPLATE MUST be used.

The PLR MUST signal non-Summary FRR enabled LSPs over the bypass tunnel before signaling the Summary FRR enabled LSPs. This is needed to allow for the case when the PLR node has recently changed a bypass assignment and the MP has not processed the change yet.

A new object FRR\_ACTIVE SUMMARY\_FRR\_BYPASS is defined in Section 3.2.1 and sent within the RSVP Path message of the bypass LSP to reroute RSVP state of Summary FRR enabled LSPs.

#### 3.2.1. SUMMARY\_FRR\_BYPASS Object

The SUMMARY\_FRR\_BYPASS Object with Type FRR\_ACTIVE is carried in the Path message of a bypass LSP. This object is added by the PLR node to indicate to the MP node (bypass tunnel destination) that one or more groups of protected LSPs that are being protected by the specified bypass tunnel are being rerouted over the bypass tunnel.

The FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object is assigned the C-Type (TBD-3). The FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object has the below format.

SUMMARY\_FRR\_BYPASS Class-Num = (TBD-2) (of the form 11bbbbbb) Class-Name = SUMMARY\_FRR\_BYPASS Class, FRR\_ACTIVE C-Type = (TBD-3)

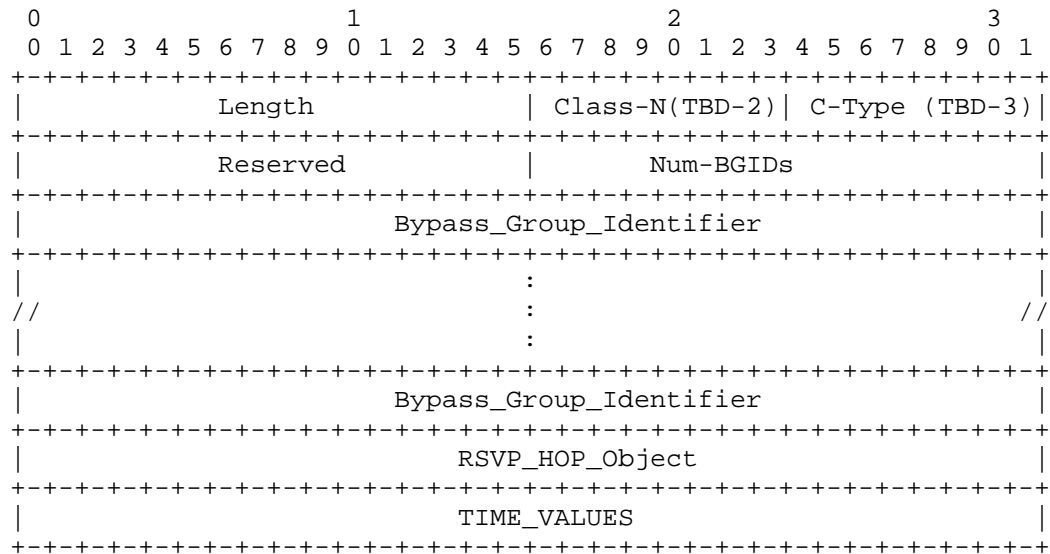


Figure 3: Summary FRR Bypass Object

Reserved: 16 bits

Reserved for future use.

Num-BGIDs: 16 bits

Number of Bypass\_Group\_Identifier fields.

Bypass\_Group\_Identifier: 32 bits

The Bypass\_Group\_Identifier that is previously advertised by the PLR using the Extended Association object. One or more Bypass\_Group\_Identifiers may be included.

RSVP\_HOP\_Object: Class 3, as defined by [RFC2205]

Replacement RSVP HOP object to be applied to all LSPs associated with each of the following Bypass\_Group\_Identifiers. This corresponds to C-Type = 1 for IPv4 RSVP HOP, or C-Type = 2 for IPv6 RSVP HOP depending on the IP address family carried within the object.

TIME\_VALUES object: Class 5, as defined by [RFC2205]

Replacement TIME\_VALUES object to be applied to all LSPs associated with each of the following Bypass\_Group\_Identifiers after receiving the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object.

### 3.2.2. PLR Summary FRR Signaling Procedure

After a failure event, when using the Summary FRR path signaling procedures, an individual RSVP Path message for each Summary FRR LSP is not signaled. Instead, to reroute Summary FRR LSPs via the bypass tunnel, the PLR adds the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object in the RSVP Path message of the RSVP session of the bypass tunnel.

The RSVP\_HOP\_Object field of the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object is set to the common RSVP\_HOP that was used by the PLR in Section 3.2 of this document.

The previously received MESSAGE\_ID from the MP is activated. As a result, the MP may refresh the protected rerouted RESV state using Summary Refresh procedures.

For each affected Summary FRR group, its Bypass\_Group\_Identifier is added to the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object.

### 3.2.3. MP Summary FRR Signaling Procedure

Upon receiving an RSVP Path message with a FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object, the MP performs normal merge point processing for each protected LSP associated with each Bypass\_Group\_Identifier, as if it received individual RSVP Path messages for the LSP.

For each Summary FRR LSP being merged, the MP first modifies the Path state as follows:

1. The RSVP\_HOP object is copied from the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS RSVP\_HOP\_Object field.
2. The TIME\_VALUES object is copied from the FRR\_ACTIVE SUMMARY\_FRR\_BYPASS TIMES\_VALUE field. The TIME\_VALUES object contains the refresh time of the PLR to generate refreshes and that would have exchanged in a Path message sent to the MP after the failure when no SFRR procedures are in effect.
3. The SENDER\_TEMPLATE object SrcAddress field is copied from the bypass tunnel SENDER\_TEMPLATE object. For the case where PLR is also the head-end, and SENDER\_TEMPLATE SrcAddress of the protected LSP and bypass tunnel are the same, the MP MUST use the modified HOP Address field instead.
4. The ERO object is modified as per Section 6.4.4. of [RFC4090]. Once the above modifications are completed, the MP then performs the merge processing as per [RFC4090].

5. The previously received MESSAGE\_ID from the PLR is activated, meaning that the PLR may now refresh the protected rerouted PATH state using Summary Refresh procedures.

A failure during merge processing of any individual rerouted LSP MUST result in an RSVP Path Error message.

An individual RSVP Resv message for each successfully merged Summary FRR LSP is not signaled. The MP node SHOULD immediately use Summary Refresh procedures to refresh the protected LSP RESV state.

### 3.3. Refreshing Summary FRR Active LSPs

Refreshing of Summary FRR active LSPs is performed using Summary Refresh as defined by [RFC2961].

## 4. Compatibility

The (Extended) ASSOCIATION object is defined in [RFC4872] with a class number in the form 1lbbbbbb, which ensures compatibility with non-supporting node(s). Such nodes will ignore the object and forward it without modification.

The new FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object is to be defined with a class number in the form 1lbbbbbb, which ensures compatibility with non-supporting nodes. Per [RFC2205], the nodes not supporting this extension will ignore the object but forward it, unexamined and unmodified, in all messages.

## 5. Security Considerations

This document updates an existing RSVP object, and introduces a new RSVP object. Thus, in the event of the interception of a signaling message, a slightly more information could be deduced about the state of the network than was previously the case. Existing mechanisms for maintaining the integrity and authenticity of RSVP protocol messages [RFC2747] can be applied.

## 6. IANA Considerations

IANA maintains the "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Parameters" registry (see <http://www.iana.org/assignments/gmpls-sig-parameters>). The "Association Type" subregistry is included in this registry.

This registry has been updated by new Association Type for Extended ASSOCIATION Object defined in this document as follows:

Value	Name	Reference
-----	-----	-----
TBD-1	Bypass Summary FRR Association (B-SFRR)	Section 2.1.1

IANA also maintains and assigns the values for the RSVP-TE protocol parameters "Resource Reservation Protocol (RSVP) Parameters" (see <http://www.iana.org/assignments/rsvp-parameters>).

From this registry, a new RSVP Class (TBD-2) and of the form 11bbbbbb and a new C-Type (TBD-3) are requested for the new FRR\_ACTIVE SUMMARY\_FRR\_BYPASS object defined in this document.

Class-Number = (TBD-2), Class-Name = SUMMARY\_FRR\_BYPASS

C-Type = (TBD-3) Name = FRR\_ACTIVE

## 7. Acknowledgments

The authors would like to thank Loa Andersson, Lou Berger, Eric Osborne, Gregory Mirsky, and Mach Chen for reviewing and providing valuable comments to this document.

## 8. Contributors

Nicholas Tan  
Arista Networks

Email: [ntan@arista.com](mailto:ntan@arista.com)

## 9. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<http://www.rfc-editor.org/info/rfc2205>>.
- [RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <<http://www.rfc-editor.org/info/rfc2747>>.

- [RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, DOI 10.17487/RFC2961, April 2001, <<http://www.rfc-editor.org/info/rfc2961>>.
- [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, <<http://www.rfc-editor.org/info/rfc3209>>.
- [RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, DOI 10.17487/RFC4090, May 2005, <<http://www.rfc-editor.org/info/rfc4090>>.
- [RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <<http://www.rfc-editor.org/info/rfc4872>>.
- [RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP ASSOCIATION Object Extensions", RFC 6780, DOI 10.17487/RFC6780, October 2012, <<http://www.rfc-editor.org/info/rfc6780>>.

#### Authors' Addresses

Mike Taillon  
Cisco Systems, Inc.

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

Tarek Saad (editor)  
Cisco Systems, Inc.

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

Rakesh Gandhi  
Cisco Systems, Inc.

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

Abhishek Deshmukh  
Juniper Networks

Email: [adeshmukh@juniper.net](mailto:adeshmukh@juniper.net)

Markus Jork  
Juniper Networks

Email: [mjork@juniper.net](mailto:mjork@juniper.net)

Vishnu Pavan Beeram  
Juniper Networks

Email: [vbeeram@juniper.net](mailto:vbeeram@juniper.net)



PCE Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 8, 2017

D. Dhody, Ed.  
Huawei Technologies  
J. Hardwick  
Metaswitch  
V. Beeram  
Juniper Networks  
J. Tantsura  
July 7, 2016

A YANG Data Model for Path Computation Element Communications Protocol  
(PCEP)  
draft-pkd-pce-pcep-yang-06

## Abstract

This document defines a YANG data model for the management of Path Computation Element communications Protocol (PCEP) for communications between a Path Computation Client (PCC) and a Path Computation Element (PCE), or between two PCEs. The data model includes configuration data and state data (status information and counters for the collection of statistics).

## Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

## Copyright Notice

Copyright (c) 2016 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 (<http://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. Requirements Language . . . . .	3
3. Terminology and Notation . . . . .	3
3.1. Tree Diagrams . . . . .	4
3.2. Prefixes in Data Node Names . . . . .	5
4. Objectives . . . . .	5
5. The Design of PCEP Data Model . . . . .	6
5.1. The Entity . . . . .	17
5.2. The Peer Lists . . . . .	18
5.3. The Session Lists . . . . .	18
5.4. Notifications . . . . .	19
6. Advanced PCE Features . . . . .	19
6.1. Stateful PCE's LSP-DB . . . . .	19
7. Open Issues and Next Step . . . . .	20
7.1. The PCE-Initiated LSP . . . . .	20
7.2. PCEP over TLS (PCEPS) . . . . .	20
8. PCEP YANG Module . . . . .	20
9. Security Considerations . . . . .	83
10. Manageability Considerations . . . . .	84
10.1. Control of Function and Policy . . . . .	84
10.2. Information and Data Models . . . . .	84
10.3. Liveness Detection and Monitoring . . . . .	84
10.4. Verify Correct Operations . . . . .	84
10.5. Requirements On Other Protocols . . . . .	84
10.6. Impact On Network Operations . . . . .	84
11. IANA Considerations . . . . .	84
12. Acknowledgements . . . . .	85
13. References . . . . .	85
13.1. Normative References . . . . .	85
13.2. Informative References . . . . .	86
Appendix A. Contributor Addresses . . . . .	88
Authors' Addresses . . . . .	89

## 1. Introduction

The Path Computation Element (PCE) defined in [RFC4655] is an entity that is capable of computing a network path or route based on a network graph, and applying computational constraints. A Path

Computation Client (PCC) may make requests to a PCE for paths to be computed.

PCEP is the communication protocol between a PCC and PCE and is defined in [RFC5440]. PCEP interactions include path computation requests and path computation replies as well as notifications of specific states related to the use of a PCE in the context of Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) Traffic Engineering (TE). [I-D.ietf-pce-stateful-pce] specifies extensions to PCEP to enable stateful control of MPLS TE LSPs.

This document defines a YANG [RFC6020] data model for the management of PCEP speakers. It is important to establish a common data model for how PCEP speakers are identified, configured, and monitored. The data model includes configuration data and state data (status information and counters for the collection of statistics).

This document contains a specification of the PCEP YANG module, "ietf-pcep" which provides the PCEP [RFC5440] data model.

## 2. 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].

## 3. Terminology and Notation

This document uses the terminology defined in [RFC4655] and [RFC5440]. In particular, it uses the following acronyms.

- o Path Computation Request message (PCReq).
- o Path Computation Reply message (PCRep).
- o Notification message (PCNtf).
- o Error message (PCErr).
- o Request Parameters object (RP).
- o Synchronization Vector object (SVEC).
- o Explicit Route object (ERO).

This document also uses the following terms defined in [RFC7420]:

- o PCEP entity: a local PCEP speaker.

- o PCEP peer: to refer to a remote PCEP speaker.
- o PCEP speaker: where it is not necessary to distinguish between local and remote.

Further, this document also uses the following terms defined in [I-D.ietf-pce-stateful-pce] :

- o Stateful PCE, Passive Stateful PCE, Active Stateful PCE
- o Delegation, Revocation, Redelegation
- o LSP State Report, Path Computation Report message (PCRpt).
- o LSP State Update, Path Computation Update message (PCUpd).

[I-D.ietf-pce-pce-initiated-lsp] :

- o PCE-initiated LSP, Path Computation LSP Initiate Message (PCInitiate).

[I-D.ietf-pce-lsp-setup-type] :

- o Path Setup Type (PST).

[I-D.ietf-pce-segment-routing] :

- o Segment Routing (SR).
- o Segment Identifier (SID).
- o Maximum SID Depth (MSD).

### 3.1. Tree Diagrams

A graphical representation of the complete data tree is presented in Section 5. The meaning of the symbols in these diagrams is as follows and as per [I-D.ietf-netmod-rfc6087bis]:

- o Brackets "[" and "]" enclose list keys.
- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- o Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).

- o Symbols after data node names: "?" means an optional node and "\*" denotes a "list" or "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

### 3.2. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

## 4. Objectives

This section describes some of the design objectives for the model:

- o In case of existing implementations, it needs to map the data model defined in this document to their proprietary native data model. To facilitate such mappings, the data model should be simple.
- o The data model should be suitable for new implementations to use as is.
- o Mapping to the PCEP MIB Module should be clear.
- o The data model should allow for static configurations of peers.
- o The data model should include read-only counters in order to gather statistics for sent and received PCEP messages, received messages with errors, and messages that could not be sent due to errors.

- o It should be fairly straightforward to augment the base data model for advanced PCE features.

## 5. The Design of PCEP Data Model

The module, "ietf-pcep", defines the basic components of a PCE speaker.

```

module: ietf-pcep
+--rw pcep!
|   +--rw entity
|   |   +--rw addr inet:ip-address
|   |   +--rw enabled? boolean
|   |   +--rw role pcep-role
|   |   +--rw description? string
|   |   +--rw domain
|   |   |   +--rw domain* [domain-type domain]
|   |   |   |   +--rw domain-type domain-type
|   |   |   |   +--rw domain domain
|   |   +--rw capability
|   |   |   +--rw gmpls? boolean {gmpls}?
|   |   |   +--rw bi-dir? boolean
|   |   |   +--rw diverse? boolean
|   |   |   +--rw load-balance? boolean
|   |   |   +--rw synchronize? boolean {svec}?
|   |   |   +--rw objective-function? boolean {obj-fn}?
|   |   |   +--rw add-path-constraint? boolean
|   |   |   +--rw prioritization? boolean
|   |   |   +--rw multi-request? boolean
|   |   |   +--rw gco? boolean {gco}?
|   |   |   +--rw p2mp? boolean {p2mp}?
|   |   |   +--rw stateful {stateful}?
|   |   |   |   +--rw enabled? boolean
|   |   |   |   +--rw active? boolean
|   |   |   |   +--rw pce-initiated? boolean {pce-initiated}?
|   |   +--rw sr {sr}?
|   |   |   +--rw enabled? boolean
|   |   |   +--rw msd? uint8
|   +--rw pce-info
|   |   +--rw scope
|   |   |   +--rw intra-area-scope? boolean
|   |   |   +--rw intra-area-pref? uint8
|   |   |   +--rw inter-area-scope? boolean
|   |   |   +--rw inter-area-scope-default? boolean
|   |   |   +--rw inter-area-pref? uint8
|   |   |   +--rw inter-as-scope? boolean
|   |   |   +--rw inter-as-scope-default? boolean
|   |   |   +--rw inter-as-pref? uint8

```

```

| | | +--rw inter-layer-scope?          boolean
| | | +--rw inter-layer-pref?          uint8
+--rw neigh-domains
| | | +--rw domain* [domain-type domain]
| | | | +--rw domain-type      domain-type
| | | | +--rw domain          domain
+--rw (auth-type-selection)?
| | | +---:(auth-key-chain)
| | | | +--rw key-chain?        key-chain:key-chain-ref
+---:(auth-key)
| | | | +--rw key?              string
| | | | +--rw crypto-algorithm
| | | | | +--rw (algorithm)?
| | | | | | +---:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
| | | | | | | +--rw hmac-sha1-12?          empty
| | | | | | +---:(aes-cmac-prf-128) {aes-cmac-prf-128}?
| | | | | | | +--rw aes-cmac-prf-128?      empty
| | | | | | +---:(md5)
| | | | | | | +--rw md5?                    empty
| | | | | | +---:(sha-1)
| | | | | | | +--rw sha-1?                  empty
| | | | | | +---:(hmac-sha-1)
| | | | | | | +--rw hmac-sha-1?            empty
| | | | | | +---:(hmac-sha-256)
| | | | | | | +--rw hmac-sha-256?          empty
| | | | | | +---:(hmac-sha-384)
| | | | | | | +--rw hmac-sha-384?          empty
| | | | | | +---:(hmac-sha-512)
| | | | | | | +--rw hmac-sha-512?          empty
| | | | | | +---:(clear-text) {clear-text}?
| | | | | | | +--rw clear-text?            empty
| | | | | | +---:(replay-protection-only) {replay-protection-only}?
| | | | | | | +--rw replay-protection-only? empty
+---:(auth-tls) {tls}?
| | | | +--rw tls
+--rw connect-timer?          uint32
+--rw connect-max-retry?      uint32
+--rw init-backoff-timer?     uint32
+--rw max-backoff-timer?      uint32
+--rw open-wait-timer?        uint32
+--rw keep-wait-timer?        uint32
+--rw keep-alive-timer?       uint32
+--rw dead-timer?             uint32
+--rw allow-negotiation?      boolean
+--rw max-keep-alive-timer?    uint32
+--rw max-dead-timer?         uint32
+--rw min-keep-alive-timer?    uint32
+--rw min-dead-timer?         uint32

```

```

+--rw sync-timer?                uint32 {svec}?
+--rw request-timer?             uint32
+--rw max-sessions?              uint32
+--rw max-unknown-reqs?          uint32
+--rw max-unknown-msgs?          uint32
+--rw pcep-notification-max-rate uint32
+--rw stateful-parameter {stateful}?
|   +--rw state-timeout?          uint32
|   +--rw redelegation-timeout?   uint32
|   +--rw rpt-non-pcep-lsp?       boolean
+--rw peers
|   +--rw peer* [addr]
|   |   +--rw addr                inet:ip-address
|   |   +--rw description?        string
|   |   +--rw domain
|   |   |   +--rw domain* [domain-type domain]
|   |   |   |   +--rw domain-type domain-type
|   |   |   |   +--rw domain      domain
|   |   +--rw capability
|   |   |   +--rw gmpls?           boolean {gmpls}?
|   |   |   +--rw bi-dir?          boolean
|   |   |   +--rw diverse?         boolean
|   |   |   +--rw load-balance?    boolean
|   |   |   +--rw synchronize?     boolean {svec}?
|   |   |   +--rw objective-function? boolean {obj-fn}?
|   |   |   +--rw add-path-constraint? boolean
|   |   |   +--rw prioritization?  boolean
|   |   |   +--rw multi-request?   boolean
|   |   |   +--rw gco?             boolean {gco}?
|   |   |   +--rw p2mp?            boolean {p2mp}?
|   |   +--rw stateful {stateful}?
|   |   |   +--rw enabled?         boolean
|   |   |   +--rw active?          boolean
|   |   |   +--rw pce-initiated?   boolean {pce-initiated}?
|   |   +--rw sr {sr}?
|   |   |   +--rw enabled?         boolean
|   |   |   +--rw msd?             uint8
|   +--rw scope
|   |   +--rw intra-area-scope?    boolean
|   |   +--rw intra-area-pref?     uint8
|   |   +--rw inter-area-scope?    boolean
|   |   +--rw inter-area-scope-default? boolean
|   |   +--rw inter-area-pref?     uint8
|   |   +--rw inter-as-scope?      boolean
|   |   +--rw inter-as-scope-default? boolean
|   |   +--rw inter-as-pref?       uint8
|   |   +--rw inter-layer-scope?   boolean
|   |   +--rw inter-layer-pref?    uint8

```



```

|      +---rw neigh-domains
|      |      +---rw domain* [domain-type domain]
|      |      |      +---rw domain-type      domain-type
|      |      |      +---rw domain          domain
+---rw delegation-pref?      uint8 {stateful}?
+---rw (auth-type-selection)?
|      +---:(auth-key-chain)
|      |      +---rw key-chain?              key-chain:key-chain-ref
+---:(auth-key)
|      +---rw key?                          string
+---rw crypto-algorithm
|      +---rw (algorithm)?
|      |      +---:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
|      |      |      +---rw hmac-sha1-12?      empty
+---:(aes-cmac-prf-128) {aes-cmac-prf-128}?
|      |      +---rw aes-cmac-prf-128?      empty
+---:(md5)
|      |      +---rw md5?                    empty
+---:(sha-1)
|      |      +---rw sha-1?                  empty
+---:(hmac-sha-1)
|      |      +---rw hmac-sha-1?            empty
+---:(hmac-sha-256)
|      |      +---rw hmac-sha-256?          empty
+---:(hmac-sha-384)
|      |      +---rw hmac-sha-384?          empty
+---:(hmac-sha-512)
|      |      +---rw hmac-sha-512?          empty
+---:(clear-text) {clear-text}?
|      |      +---rw clear-text?            empty
+---:(replay-protection-only) {replay-protection-only}
? |      |      +---rw replay-protection-only?  empty
|      |      +---:(auth-tls) {tls}?
|      |      +---rw tls
+---ro pcep-state
|      +---ro entity
|      |      +---ro addr?                  inet:ip-address
|      |      +---ro index?                uint32
|      |      +---ro admin-status?          pcep-admin-status
|      |      +---ro oper-status?          pcep-admin-status
|      |      +---ro role?                 pcep-role
+---ro domain
|      |      +---ro domain* [domain-type domain]
|      |      |      +---ro domain-type      domain-type
|      |      |      +---ro domain          domain
+---ro capability
|      |      +---ro gmpls?                 boolean {gmpls}?
|      |      +---ro bi-dir?               boolean

```

```

| +--ro diverse?                boolean
| +--ro load-balance?           boolean
| +--ro synchronize?            boolean {svec}?
| +--ro objective-function?     boolean {obj-fn}?
| +--ro add-path-constraint?    boolean
| +--ro prioritization?         boolean
| +--ro multi-request?          boolean
| +--ro gco?                    boolean {gco}?
| +--ro p2mp?                   boolean {p2mp}?
| +--ro stateful {stateful}?
| | +--ro enabled?              boolean
| | +--ro active?               boolean
| | +--ro pce-initiated?       boolean {pce-initiated}?
| +--ro sr {sr}?
| | +--ro enabled?              boolean
| | +--ro msd?                  uint8
+--ro pce-info
| +--ro scope
| | +--ro intra-area-scope?     boolean
| | +--ro intra-area-pref?      uint8
| | +--ro inter-area-scope?     boolean
| | +--ro inter-area-scope-default? boolean
| | +--ro inter-area-pref?      uint8
| | +--ro inter-as-scope?       boolean
| | +--ro inter-as-scope-default? boolean
| | +--ro inter-as-pref?        uint8
| | +--ro inter-layer-scope?    boolean
| | +--ro inter-layer-pref?     uint8
| +--ro neigh-domains
| | +--ro domain* [domain-type domain]
| | | +--ro domain-type        domain-type
| | | +--ro domain              domain
| +--ro (auth-type-selection)?
| | +--:(auth-key-chain)
| | | +--ro key-chain?          key-chain:key-chain-ref
| | +--:(auth-key)
| | | +--ro key?                 string
| | | +--ro crypto-algorithm
| | | | +--ro (algorithm)?
| | | | | +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
| | | | | | +--ro hmac-sha1-12?          empty
| | | | | +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
| | | | | | +--ro aes-cmac-prf-128?      empty
| | | | +--:(md5)
| | | | | +--ro md5?                    empty
| | | | +--:(sha-1)
| | | | | +--ro sha-1?                  empty
| | | | +--:(hmac-sha-1)

```

```

|         |         |   +--ro hmac-sha-1?           empty
|         |         |   +---:(hmac-sha-256)
|         |         |   |   +--ro hmac-sha-256?       empty
|         |         |   |   +---:(hmac-sha-384)
|         |         |   |   |   +--ro hmac-sha-384?    empty
|         |         |   |   |   +---:(hmac-sha-512)
|         |         |   |   |   |   +--ro hmac-sha-512? empty
|         |         |   |   |   |   +---:(clear-text) {clear-text}?
|         |         |   |   |   |   |   +--ro clear-text? empty
|         |         |   |   |   |   |   +---:(replay-protection-only) {replay-protection-only}?
|         |         |   |   |   |   |   |   +--ro replay-protection-only? empty
|         |         |   +---:(auth-tls) {tls}?
|         |         |   +--ro tls
+--ro connect-timer?          uint32
+--ro connect-max-retry?      uint32
+--ro init-backoff-timer?     uint32
+--ro max-backoff-timer?      uint32
+--ro open-wait-timer?        uint32
+--ro keep-wait-timer?        uint32
+--ro keep-alive-timer?       uint32
+--ro dead-timer?            uint32
+--ro allow-negotiation?      boolean
+--ro max-keep-alive-timer?    uint32
+--ro max-dead-timer?         uint32
+--ro min-keep-alive-timer?    uint32
+--ro min-dead-timer?         uint32
+--ro sync-timer?            uint32 {svec}?
+--ro request-timer?          uint32
+--ro max-sessions?           uint32
+--ro max-unknown-reqs?       uint32
+--ro max-unknown-msgs?       uint32
+--ro stateful-parameter {stateful}?
|   +--ro state-timeout?      uint32
|   +--ro redelegation-timeout? uint32
|   +--ro rpt-non-pcep-lsp?    boolean
+--ro lsp-db {stateful}?
|   +--ro association-list* [id source global-source extended-id]
|   |   +--ro type?           assoc-type
|   |   +--ro id              uint16
|   |   +--ro source          inet:ip-address
|   |   +--ro global-source    uint32
|   |   +--ro extended-id     string
|   |   +--ro lsp* [plsp-id pcc-id]
|   |   |   +--ro plsp-id      -> /pcep-state/entity/lsp-db/lsp/plsp-id
|   |   |   +--ro pcc-id      -> /pcep-state/entity/lsp-db/lsp/pcc-id
+--ro lsp* [plsp-id pcc-id]
|   +--ro plsp-id             uint32
|   +--ro pcc-id              inet:ip-address

```

```

on
|
|   +---ro lsp-ref
|   |   +---ro source?          -> /te:te/lsp-state/lsp/source
|   |   +---ro destination?     -> /te:te/lsp-state/lsp/destinati
|
|   +---ro tunnel-id?          -> /te:te/lsp-state/lsp/tunnel-id
|   +---ro lsp-id?            -> /te:te/lsp-state/lsp/lsp-id
|   +---ro extended-tunnel-id? -> /te:te/lsp-state/lsp/extended-
tunnel-id
|
|   +---ro type?                -> /te:te/lsp-state/lsp/type
+---ro admin-state?            boolean
+---ro operational-state?      operational-state
+---ro delegated
|   +---ro enabled?            boolean
|   +---ro pce?                -> /pcep-state/entity/peers/peer/addr
|   +---ro srp-id?            uint32
+---ro initiation {pce-initiated}?
|   +---ro enabled?            boolean
|   +---ro pce?                -> /pcep-state/entity/peers/peer/addr
+---ro symbolic-path-name?     string
+---ro last-error?             lsp-error
+---ro pst?                    pst
+---ro association-list* [id source global-source extended-id]
|   +---ro id                  -> /pcep-state/entity/lsp-db/associatio
n-list/id
|   +---ro source              -> /pcep-state/entity/lsp-db/associatio
n-list/source
|   +---ro global-source       -> /pcep-state/entity/lsp-db/associatio
n-list/global-source
|   +---ro extended-id        -> /pcep-state/entity/lsp-db/associatio
n-list/extended-id
+---ro peers
+---ro peer* [addr]
|   +---ro addr                inet:ip-address
|   +---ro role?               pcep-role
|   +---ro domain
|   |   +---ro domain* [domain-type domain]
|   |   |   +---ro domain-type  domain-type
|   |   |   +---ro domain      domain
+---ro capability
|   +---ro gmpls?              boolean {gmpls}?
|   +---ro bi-dir?             boolean
|   +---ro diverse?            boolean
|   +---ro load-balance?       boolean
|   +---ro synchronize?        boolean {svec}?
|   +---ro objective-function?  boolean {obj-fn}?
|   +---ro add-path-constraint? boolean
|   +---ro prioritization?      boolean
|   +---ro multi-request?       boolean
|   +---ro gco?                boolean {gco}?
|   +---ro p2mp?               boolean {p2mp}?
+---ro stateful {stateful}?
|   +---ro enabled?            boolean
|   +---ro active?             boolean
|   +---ro pce-initiated?      boolean {pce-initiated}?

```

```

|   +--ro sr {sr}?
|   |   +--ro enabled?    boolean
|   |   +--ro msd?       uint8
+--ro pce-info
|   +--ro scope
|   |   +--ro intra-area-scope?    boolean
|   |   +--ro intra-area-pref?    uint8
|   |   +--ro inter-area-scope?    boolean
|   |   +--ro inter-area-scope-default?    boolean
|   |   +--ro inter-area-pref?    uint8
|   |   +--ro inter-as-scope?    boolean
|   |   +--ro inter-as-scope-default?    boolean
|   |   +--ro inter-as-pref?    uint8
|   |   +--ro inter-layer-scope?    boolean
|   |   +--ro inter-layer-pref?    uint8
+--ro neigh-domains
|   +--ro domain* [domain-type domain]
|   |   +--ro domain-type    domain-type
|   |   +--ro domain        domain
+--ro delegation-pref?    uint8 {stateful}?
+--ro (auth-type-selection)?
|   +--:(auth-key-chain)
|   |   +--ro key-chain?    key-chain:key-chain-ref
+--:(auth-key)
|   +--ro key?    string
+--ro crypto-algorithm
|   +--ro (algorithm)?
|   |   +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
|   |   |   +--ro hmac-sha1-12?    empty
|   |   +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
|   |   |   +--ro aes-cmac-prf-128?    empty
|   |   +--:(md5)
|   |   |   +--ro md5?    empty
|   |   +--:(sha-1)
|   |   |   +--ro sha-1?    empty
|   |   +--:(hmac-sha-1)
|   |   |   +--ro hmac-sha-1?    empty
|   |   +--:(hmac-sha-256)
|   |   |   +--ro hmac-sha-256?    empty
|   |   +--:(hmac-sha-384)
|   |   |   +--ro hmac-sha-384?    empty
|   |   +--:(hmac-sha-512)
|   |   |   +--ro hmac-sha-512?    empty
|   |   +--:(clear-text) {clear-text}?
|   |   |   +--ro clear-text?    empty
|   |   +--:(replay-protection-only) {replay-protection-only}
?
|   |   +--ro replay-protection-only?    empty
+--:(auth-tls) {tls}?

```

```

|      +--ro tls
+--ro discontinuity-time?      yang:timestamp
+--ro initiate-session?       boolean
+--ro session-exists?         boolean
+--ro num-sess-setup-ok?      yang:counter32
+--ro num-sess-setup-fail?    yang:counter32
+--ro session-up-time?        yang:timestamp
+--ro session-fail-time?      yang:timestamp
+--ro session-fail-up-time?   yang:timestamp
+--ro pcep-stats
|   +--ro avg-rsp-time?       uint32
|   +--ro lwm-rsp-time?       uint32
|   +--ro hwm-rsp-time?       uint32
|   +--ro num-pcreq-sent?     yang:counter32
|   +--ro num-pcreq-rcvd?     yang:counter32
|   +--ro num-pcrep-sent?     yang:counter32
|   +--ro num-pcrep-rcvd?     yang:counter32
|   +--ro num-pcerr-sent?     yang:counter32
|   +--ro num-pcerr-rcvd?     yang:counter32
|   +--ro num-pcntf-sent?     yang:counter32
|   +--ro num-pcntf-rcvd?     yang:counter32
|   +--ro num-keepalive-sent? yang:counter32
|   +--ro num-keepalive-rcvd? yang:counter32
|   +--ro num-unknown-rcvd?   yang:counter32
|   +--ro num-corrupt-rcvd?   yang:counter32
|   +--ro num-req-sent?       yang:counter32
|   +--ro num-req-sent-pend-rep? yang:counter32
|   +--ro num-req-sent-ero-rcvd? yang:counter32
|   +--ro num-req-sent-nopath-rcvd? yang:counter32
|   +--ro num-req-sent-cancel-rcvd? yang:counter32
|   +--ro num-req-sent-error-rcvd? yang:counter32
|   +--ro num-req-sent-timeout? yang:counter32
|   +--ro num-req-sent-cancel-sent? yang:counter32
|   +--ro num-req-rcvd?       yang:counter32
|   +--ro num-req-rcvd-pend-rep? yang:counter32
|   +--ro num-req-rcvd-ero-sent? yang:counter32
|   +--ro num-req-rcvd-nopath-sent? yang:counter32
|   +--ro num-req-rcvd-cancel-sent? yang:counter32
|   +--ro num-req-rcvd-error-sent? yang:counter32
|   +--ro num-req-rcvd-cancel-rcvd? yang:counter32
|   +--ro num-rep-rcvd-unknown? yang:counter32
|   +--ro num-req-rcvd-unknown? yang:counter32
|   +--ro svec {svec}?
|   |   +--ro num-svec-sent?   yang:counter32
|   |   +--ro num-svec-req-sent? yang:counter32
|   |   +--ro num-svec-rcvd?   yang:counter32
|   |   +--ro num-svec-req-rcvd? yang:counter32
+--ro stateful {stateful}?

```

```

+--ro num-pcrpt-sent?                yang:counter32
+--ro num-pcrpt-rcvd?                yang:counter32
+--ro num-pcupd-sent?                yang:counter32
+--ro num-pcupd-rcvd?                yang:counter32
+--ro num-rpt-sent?                  yang:counter32
+--ro num-rpt-rcvd?                  yang:counter32
+--ro num-rpt-rcvd-error-sent?       yang:counter32
+--ro num-upd-sent?                  yang:counter32
+--ro num-upd-rcvd?                  yang:counter32
+--ro num-upd-rcvd-unknown?          yang:counter32
+--ro num-upd-rcvd-undelegated?      yang:counter32
+--ro num-upd-rcvd-error-sent?       yang:counter32
+--ro initiation {pce-initiated}?
    +--ro num-pcinitiate-sent?        yang:counter32
    +--ro num-pcinitiate-rcvd?        yang:counter32
    +--ro num-initiate-sent?          yang:counter32
    +--ro num-initiate-rcvd?          yang:counter32
    +--ro num-initiate-rcvd-error-sent? yang:counter32
+--ro num-req-sent-closed?            yang:counter32
+--ro num-req-rcvd-closed?            yang:counter32
+--ro sessions
    +--ro session* [initiator]
        +--ro initiator                pcep-initiator
        +--ro state-last-change?        yang:timestamp
        +--ro state?                    pcep-sess-state
        +--ro session-creation?         yang:timestamp
        +--ro connect-retry?            yang:counter32
        +--ro local-id?                  uint32
        +--ro remote-id?                 uint32
        +--ro keepalive-timer?           uint32
        +--ro peer-keepalive-timer?      uint32
        +--ro dead-timer?                uint32
        +--ro peer-dead-timer?           uint32
        +--ro ka-hold-time-rem?          uint32
        +--ro overloaded?                boolean
        +--ro overload-time?             uint32
        +--ro peer-overloaded?           boolean
        +--ro peer-overload-time?        uint32
        +--ro lspdb-sync?                sync-state {stateful}?
        +--ro discontinuity-time?        yang:timestamp
    +--ro pcep-stats
        +--ro avg-rsp-time?              uint32
        +--ro lwm-rsp-time?              uint32
        +--ro hwm-rsp-time?              uint32
        +--ro num-pcreq-sent?            yang:counter32
        +--ro num-pcreq-rcvd?            yang:counter32
        +--ro num-pcrep-sent?            yang:counter32
        +--ro num-pcrep-rcvd?            yang:counter32

```

```

+--ro num-pcerr-sent?                yang:counter32
+--ro num-pcerr-rcvd?                yang:counter32
+--ro num-pcntf-sent?                yang:counter32
+--ro num-pcntf-rcvd?                yang:counter32
+--ro num-keepalive-sent?            yang:counter32
+--ro num-keepalive-rcvd?            yang:counter32
+--ro num-unknown-rcvd?              yang:counter32
+--ro num-corrupt-rcvd?              yang:counter32
+--ro num-req-sent?                  yang:counter32
+--ro num-req-sent-pend-rep?          yang:counter32
+--ro num-req-sent-ero-rcvd?          yang:counter32
+--ro num-req-sent-nopath-rcvd?       yang:counter32
+--ro num-req-sent-cancel-rcvd?       yang:counter32
+--ro num-req-sent-error-rcvd?        yang:counter32
+--ro num-req-sent-timeout?           yang:counter32
+--ro num-req-sent-cancel-sent?       yang:counter32
+--ro num-req-rcvd?                  yang:counter32
+--ro num-req-rcvd-pend-rep?          yang:counter32
+--ro num-req-rcvd-ero-sent?          yang:counter32
+--ro num-req-rcvd-nopath-sent?       yang:counter32
+--ro num-req-rcvd-cancel-sent?       yang:counter32
+--ro num-req-rcvd-error-sent?        yang:counter32
+--ro num-req-rcvd-cancel-rcvd?       yang:counter32
+--ro num-rep-rcvd-unknown?           yang:counter32
+--ro num-req-rcvd-unknown?           yang:counter32
+--ro svec {svec}?
|   +--ro num-svec-sent?              yang:counter32
|   +--ro num-svec-req-sent?          yang:counter32
|   +--ro num-svec-rcvd?              yang:counter32
|   +--ro num-svec-req-rcvd?          yang:counter32
+--ro stateful {stateful}?
    +--ro num-pcrpt-sent?              yang:counter32
    +--ro num-pcrpt-rcvd?              yang:counter32
    +--ro num-pcupd-sent?              yang:counter32
    +--ro num-pcupd-rcvd?              yang:counter32
    +--ro num-rpt-sent?                yang:counter32
    +--ro num-rpt-rcvd?                yang:counter32
    +--ro num-rpt-rcvd-error-sent?     yang:counter32
    +--ro num-upd-sent?                yang:counter32
    +--ro num-upd-rcvd?                yang:counter32
    +--ro num-upd-rcvd-unknown?        yang:counter32
    +--ro num-upd-rcvd-undelegated?    yang:counter32
    +--ro num-upd-rcvd-error-sent?     yang:counter32
+--ro initiation {pce-initiated}?
    +--ro num-pcinitiate-sent?         yang:counter
    +--ro num-pcinitiate-rcvd?         yang:counter
    +--ro num-initiate-sent?           yang:counter
    +--ro num-initiate-rcvd?           yang:counter

```



```

32                                     +--ro num-initiate-rcvd-error-sent?   yang:counter
notifications:
  +---n pcep-session-up
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro session-initiator?       -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
  |   +--ro state-last-change?       yang:timestamp
  |   +--ro state?                   pcep-sess-state
  +---n pcep-session-down
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro session-initiator?       pcep-initiator
  |   +--ro state-last-change?       yang:timestamp
  |   +--ro state?                   pcep-sess-state
  +---n pcep-session-local-overload
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro session-initiator?       -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
  |   +--ro overloaded?               boolean
  |   +--ro overload-time?            uint32
  +---n pcep-session-local-overload-clear
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro overloaded?               boolean
  +---n pcep-session-peer-overload
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro session-initiator?       -> /pcep-state/entity/peers/peer/sessions/sess
ion/initiator
  |   +--ro peer-overloaded?           boolean
  |   +--ro peer-overload-time?       uint32
  +---n pcep-session-peer-overload-clear
  |   +--ro peer-addr?                -> /pcep-state/entity/peers/peer/addr
  |   +--ro peer-overloaded?           boolean

```

### 5.1. The Entity

The PCEP yang module may contain status information for the local PCEP entity.

The entity has an IP address (using ietf-inet-types [RFC6991]) and a "role" leaf (the local entity PCEP role) as mandatory.

Note that, the PCEP MIB module [RFC7420] uses an entity list and a system generated entity index as a primary index to the read only entity table. If the device implements the PCEP MIB, the "index" leaf MUST contain the value of the corresponding pcepPcepEntityIndex and only one entity is assumed.

## 5.2. The Peer Lists

The peer list contains peer(s) that the local PCEP entity knows about. A PCEP speaker is identified by its IP address. If there is a PCEP speaker in the network that uses multiple IP addresses then it looks like multiple distinct peers to the other PCEP speakers in the network.

Since PCEP sessions can be ephemeral, the peer list tracks a peer even when no PCEP session currently exists to that peer. The statistics contained are an aggregate of the statistics for all successive sessions to that peer.

To limit the quantity of information that is stored, an implementation MAY choose to discard this information if and only if no PCEP session exists to the corresponding peer.

The data model for PCEP peer presented in this document uses a flat list of peers. Each peer in the list is identified by its IP address (addr-type, addr).

There is one list for static peer configuration ("/pcep/entity/peers"), and a separate list for the operational state of all peers (i.e. static as well as discovered)("/pcep-state/entity/peers"). The former is used to enable remote PCE configuration at PCC (or PCE) while the latter has the operational state of these peers as well as the remote PCE peer which were discovered and PCC peers that have initiated session.

## 5.3. The Session Lists

The session list contains PCEP session that the PCEP entity (PCE or PCC) is currently participating in. The statistics in session are semantically different from those in peer since the former applies to the current session only, whereas the latter is the aggregate for all sessions that have existed to that peer.

Although [RFC5440] forbids more than one active PCEP session between a given pair of PCEP entities at any given time, there is a window during session establishment where two sessions may exist for a given pair, one representing a session initiated by the local PCEP entity and the other representing a session initiated by the peer. If either of these sessions reaches active state first, then the other is discarded.

The data model for PCEP session presented in this document uses a flat list of sessions. Each session in the list is identified by its

initiator. This index allows two sessions to exist transiently for a given peer, as discussed above.

There is only one list for the operational state of all sessions ("/pcep-state/entity/peers/peer/sessions/session").

#### 5.4. Notifications

This YANG model defines a list of notifications to inform client of important events detected during the protocol operation. The notifications defined cover the PCEP MIB notifications.

#### 6. Advanced PCE Features

This document contains a specification of the base PCEP YANG module, "ietf-pcep" which provides the basic PCEP [RFC5440] data model.

This document further handles advanced PCE features like -

- o Capability and Scope
- o Domain information (local/neighbour)
- o Path-Key
- o OF
- o GCO
- o P2MP
- o GMPLS
- o Inter-Layer
- o Stateful PCE
- o Segment Routing
- o Authentication including PCEPS (TLS)

[Editor's Note - Some of them would be added in a future revision.]

##### 6.1. Stateful PCE's LSP-DB

In the operational state of PCEP which supports stateful PCE mode, the list of LSP state are maintained in LSP-DB. The key is the PLSP-ID and the PCC IP address.

The PCEP data model contains the operational state of LSPs (/pcep-state/entity/lsp-db/lsp/) with PCEP specific attributes. The generic TE attributes of the LSP are defined in [I-D.ietf-teas-yang-te]. A reference to LSP state in TE model is maintained.

## 7. Open Issues and Next Step

This section is added so that open issues can be tracked. This section would be removed when the document is ready for publication.

### 7.1. The PCE-Initiated LSP

The TE Model at [I-D.ietf-teas-yang-te] should support creationg of tunnels at the controller (PCE) and marking them as PCE-Initiated. The LSP-DB in the PCEP Yang (/pcep-state/entity/lsp-db/lsp/initiation) also marks the LSPs which are PCE-initiated.

### 7.2. PCEP over TLS (PCEPS)

A future version of this document would add TLS related configurations.

## 8. PCEP YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

```
<CODE BEGINS> file "ietf-pcep@2016-07-07.yang"
module ietf-pcep {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pcep";
  prefix pcep;

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

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

  import ietf-te {
    prefix "te";
  }

  import ietf-key-chain {
    prefix "key-chain";
  }
}
```

```
}

organization
  "IETF PCE (Path Computation Element) Working Group";

contact
  "WG Web:    <http://tools.ietf.org/wg/pce/>
  WG List:    <mailto:pce@ietf.org>
  WG Chair:   JP Vasseur
              <mailto:jpv@cisco.com>
  WG Chair:   Julien Meuric
              <mailto:julien.meuric@orange.com>
  WG Chair:   Jonathan Hardwick
              <mailto:Jonathan.Hardwick@metaswitch.com>
  Editor:     Dhruv Dhody
              <mailto:dhruv.ietf@gmail.com>";

description
  "The YANG module defines a generic configuration and
  operational model for PCEP common across all of the
  vendor implementations.";

revision 2016-07-07 {
  description "Initial revision.";
  reference
    "RFC XXXX:  A YANG Data Model for Path Computation
    Element Communications Protocol
    (PCEP)";
}

/*
 * Identities
 */

identity pcep {
  description "Identity for the PCEP protocol.";
}

/*
 * Typedefs
 */
typedef pcep-role {
  type enumeration {
    enum unknown {
      value "0";
      description
```

```
        "An unknown role";
    }
    enum pcc {
        value "1";
        description
            "The role of a Path Computation Client";
    }
    enum pce {
        value "2";
        description
            "The role of Path Computation Element";
    }
    enum pcc-and-pce {
        value "3";
        description
            "The role of both Path Computation Client and
            Path Computation Element";
    }
}

description
    "The role of a PCEP speaker.
    Takes one of the following values
    - unknown(0): the role is not known.
    - pcc(1): the role is of a Path Computation
      Client (PCC).
    - pce(2): the role is of a Path Computation
      Server (PCE).
    - pccAndPce(3): the role is of both a PCC and
      a PCE.";
}

typedef pcep-admin-status {
    type enumeration {
        enum admin-status-up {
            value "1";
            description
                "Admin Status is Up";
        }
        enum admin-status-down {
            value "2";
            description
                "Admin Status is Down";
        }
    }
}

description
```

```
"The Admin Status of the PCEP entity.
  Takes one of the following values
    - admin-status-up(1): Admin Status is Up.
    - admin-status-down(2): Admin Status is Down";
}

typedef pcep-oper-status {
  type enumeration {
    enum oper-status-up {
      value "1";
      description
        "The PCEP entity is active";
    }
    enum oper-status-down {
      value "2";
      description
        "The PCEP entity is inactive";
    }
    enum oper-status-going-up {
      value "3";
      description
        "The PCEP entity is activating";
    }
    enum oper-status-going-down {
      value "4";
      description
        "The PCEP entity is deactivating";
    }
    enum oper-status-failed {
      value "5";
      description
        "The PCEP entity has failed and will recover
        when possible.";
    }
    enum oper-status-failed-perm {
      value "6";
      description
        "The PCEP entity has failed and will not recover
        without operator intervention";
    }
  }
}
description
  "The operational status of the PCEP entity.
  Takes one of the following values
    - oper-status-up(1): Active
    - oper-status-down(2): Inactive
    - oper-status-going-up(3): Activating
    - oper-status-going-down(4): Deactivating
```

```
        - oper-status-failed(5): Failed
        - oper-status-failed-perm(6): Failed Permanantly";
    }

    typedef pcep-initiator {
        type enumeration {
            enum local {
                value "1";
                description
                    "The local PCEP entity initiated the session";
            }

            enum remote {
                value "2";
                description
                    "The remote PCEP peer initiated the session";
            }
        }
        description
            "The initiator of the session, that is, whether the TCP
            connection was initiated by the local PCEP entity or
            the remote peer.
            Takes one of the following values
            - local(1): Initiated locally
            - remote(2): Initiated remotely";
    }

    typedef pcep-sess-state {
        type enumeration {
            enum tcp-pending {
                value "1";
                description
                    "The tcp-pending state of PCEP session.";
            }

            enum open-wait {
                value "2";
                description
                    "The open-wait state of PCEP session.";
            }

            enum keep-wait {
                value "3";
                description
                    "The keep-wait state of PCEP session.";
            }

            enum session-up {
```



```
        value "4";
        description
            "The session-up state of PCEP session.";
    }
}
description
    "The current state of the session.
    The set of possible states excludes the idle state
    since entries do not exist in the idle state.
    Takes one of the following values
    - tcp-pending(1): PCEP TCP Pending state
    - open-wait(2): PCEP Open Wait state
    - keep-wait(3): PCEP Keep Wait state
    - session-up(4): PCEP Session Up state";
}

typedef domain-type {
    type enumeration {
        enum ospf-area {
            value "1";
            description
                "The OSPF area.";
        }
        enum isis-area {
            value "2";
            description
                "The IS-IS area.";
        }
        enum as {
            value "3";
            description
                "The Autonomous System (AS).";
        }
    }
    description
        "The PCE Domain Type";
}

typedef domain-ospf-area {
    type union {
        type uint32;
        type yang:dotted-quad;
    }
    description
        "OSPF Area ID.";
}

typedef domain-isis-area {
```

```
    type string {
      pattern '[0-9A-Fa-f]{2}\.([0-9A-Fa-f]{4}\.){0,3}';
    }
    description
      "IS-IS Area ID.";
  }

  typedef domain-as {
    type uint32;
    description
      "Autonomous System number.";
  }

  typedef domain {
    type union {
      type domain-ospf-area;
      type domain-isis-area;
      type domain-as;
    }
    description
      "The Domain Information";
  }

  typedef operational-state {
    type enumeration {
      enum down {
        value "0";
        description
          "not active.";
      }
      enum up {
        value "1";
        description
          "signalled.";
      }
      enum active {
        value "2";
        description
          "up and carrying traffic.";
      }
      enum going-down {
        value "3";
        description
          "LSP is being torn down, resources are
            being released.";
      }
      enum going-up {
```

```
        value "4";
        description
            "LSP is being signalled.";
    }
}
description
    "The operational status of the LSP";
}

typedef lsp-error {
    type enumeration {
        enum no-error {
            value "0";
            description
                "No error, LSP is fine.";
        }
        enum unknown {
            value "1";
            description
                "Unknown reason.";
        }
        enum limit {
            value "2";
            description
                "Limit reached for PCE-controlled LSPs.";
        }
        enum pending {
            value "3";
            description
                "Too many pending LSP update requests.";
        }
        enum unacceptable {
            value "4";
            description
                "Unacceptable parameters.";
        }
        enum internal {
            value "5";
            description
                "Internal error.";
        }
        enum admin {
            value "6";
            description
                "LSP administratively brought down.";
        }
        enum preempted {
            value "7";
```

```
        description
            "LSP preempted.";
    }
    enum rsvp {
        value "8";
        description
            "RSVP signaling error.";
    }
}
description
    "The LSP Error Codes.";
}

typedef sync-state {
    type enumeration {
        enum pending {
            value "0";
            description
                "The state synchronization
                 has not started.";
        }
        enum ongoing {
            value "1";
            description
                "The state synchronization
                 is ongoing.";
        }
        enum finished {
            value "2";
            description
                "The state synchronization
                 is finished.";
        }
    }
}
description
    "The LSP-DB state synchronization operational status.";
}

typedef pst{
    type enumeration{
        enum rsvp-te{
            value "0";
            description
                "RSVP-TE signaling protocol";
        }
        enum sr{
            value "1";
            description
```

```

                                "Segment Routing Traffic Engineering";
                                }
                                }
                                description
                                    "The Path Setup Type";
                                }

typedef assoc-type{
    type enumeration{
        enum protection{
            value "1";
            description
                "Path Protection Association Type";
        }
    }
    description
        "The PCEP Association Type";
}

/*
 * Features
 */

feature svec {
    description
        "Support synchronized path computation.";
}

feature gmpls {
    description
        "Support GMPLS.";
}

feature obj-fn {
    description
        "Support OF as per RFC 5541.";
}

feature gco {
    description
        "Support GCO as per RFC 5557.";
}

feature pathkey {
    description
        "Support pathkey as per RFC 5520.";
}
```

```
feature p2mp {
    description
        "Support P2MP as per RFC 6006.";
}

feature stateful {
    description
        "Support stateful PCE.";
}

feature pce-initiated {
    description
        "Support PCE-Initiated LSP.";
}

feature tls {
    description
        "Support PCEP over TLS.";
}

feature sr {
    description
        "Support Segement Routing for PCE.";
}

/*
 * Groupings
 */

grouping pcep-entity-info{
    description
        "This grouping defines the attributes for PCEP entity.";
    leaf connect-timer {
        type uint32 {
            range "1..65535";
        }
        units "seconds";
        default 60;
        description
            "The time in seconds that the PCEP entity will wait
            to establish a TCP connection with a peer.  If a
            TCP connection is not established within this time
            then PCEP aborts the session setup attempt.";
        reference
            "RFC 5440: Path Computation Element (PCE)
            Communication Protocol (PCEP)";
    }
}
```

```
}

leaf connect-max-retry {
  type uint32;
  default 5;
  description
    "The maximum number of times the system tries to
    establish a TCP connection to a peer before the
    session with the peer transitions to the idle
    state.";
  reference
    "RFC 5440: Path Computation Element (PCE)
    Communication Protocol (PCEP)";
}

leaf init-backoff-timer {
  type uint32 {
    range "1..65535";
  }
  units "seconds";
  description
    "The initial back-off time in seconds for retrying
    a failed session setup attempt to a peer.
    The back-off time increases for each failed
    session setup attempt, until a maximum back-off
    time is reached. The maximum back-off time is
    max-backoff-timer.";
}

leaf max-backoff-timer {
  type uint32;
  units "seconds";
  description
    "The maximum back-off time in seconds for retrying
    a failed session setup attempt to a peer.
    The back-off time increases for each failed session
    setup attempt, until this maximum value is reached.
    Session setup attempts then repeat periodically
    without any further increase in back-off time.";
}

leaf open-wait-timer {
  type uint32 {
    range "1..65535";
  }
  units "seconds";
  default 60;
  description
```

```
        "The time in seconds that the PCEP entity will wait
        to receive an Open message from a peer after the
        TCP connection has come up.
        If no Open message is received within this time then
        PCEP terminates the TCP connection and deletes the
        associated sessions.";
    reference
        "RFC 5440: Path Computation Element (PCE)
        Communication Protocol (PCEP)";
}

leaf keep-wait-timer {
    type uint32 {
        range "1..65535";
    }
    units "seconds";
    default 60;
    description
        "The time in seconds that the PCEP entity will wait
        to receive a Keepalive or PCErr message from a peer
        during session initialization after receiving an
        Open message. If no Keepalive or PCErr message is
        received within this time then PCEP terminates the
        TCP connection and deletes the associated
        sessions.";
    reference
        "RFC 5440: Path Computation Element (PCE)
        Communication Protocol (PCEP)";
}

leaf keep-alive-timer {
    type uint32 {
        range "0..255";
    }
    units "seconds";
    default 30;
    description
        "The keep alive transmission timer that this PCEP
        entity will propose in the initial OPEN message of
        each session it is involved in. This is the
        maximum time between two consecutive messages sent
        to a peer. Zero means that the PCEP entity prefers
        not to send Keepalives at all.
        Note that the actual Keepalive transmission
        intervals, in either direction of an active PCEP
        session, are determined by negotiation between the
        peers as specified by RFC 5440, and so may differ
        from this configured value.";
```



```
        reference
            "RFC 5440: Path Computation Element (PCE)
              Communication Protocol (PCEP)";
    }

    leaf dead-timer {
        type uint32 {
            range "0..255";
        }
        units "seconds";
        must ". >= ../keep-alive-timer" {
            error-message "The dead timer must be "
                + "larger than the keep alive timer";
            description
                "This value MUST be greater than
                 keep-alive-timer.";
        }
        default 120;
        description
            "The dead timer that this PCEP entity will propose
             in the initial OPEN message of each session it is
             involved in. This is the time after which a peer
             should declare a session down if it does not
             receive any PCEP messages. Zero suggests that the
             peer does not run a dead timer at all." ;
        reference
            "RFC 5440: Path Computation Element (PCE)
              Communication Protocol (PCEP)";
    }

    leaf allow-negotiation{
        type boolean;
        description
            "Whether the PCEP entity will permit negotiation of
             session parameters.";
    }

    leaf max-keep-alive-timer{
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "In PCEP session parameter negotiation in seconds,
             the maximum value that this PCEP entity will
             accept from a peer for the interval between
             Keepalive transmissions. Zero means that the PCEP
```

```
        entity will allow no Keepalive transmission at
        all." ;
    }

    leaf max-dead-timer{
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "In PCEP session parameter negotiation in seconds,
            the maximum value that this PCEP entity will accept
            from a peer for the Dead timer.  Zero means that
            the PCEP entity will allow not running a Dead
            timer.";
    }

    leaf min-keep-alive-timer{
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "In PCEP session parameter negotiation in seconds,
            the minimum value that this PCEP entity will
            accept for the interval between Keepalive
            transmissions. Zero means that the PCEP entity
            insists on no Keepalive transmission at all.";
    }

    leaf min-dead-timer{
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "In PCEP session parameter negotiation in
            seconds, the minimum value that this PCEP entity
            will accept for the Dead timer.  Zero means that
            the PCEP entity insists on not running a Dead
            timer.";
    }

    leaf sync-timer{
        if-feature svec;
        type uint32 {
            range "0..65535";
        }
    }
```

```
    units "seconds";
    default 60;
    description
        "The value of SyncTimer in seconds is used in the
        case of synchronized path computation request
        using the SVEC object. Consider the case where a
        PCReq message is received by a PCE that contains
        the SVEC object referring to M synchronized path
        computation requests. If after the expiration of
        the SyncTimer all the M path computation requests
        have not been, received a protocol error is
        triggered and the PCE MUST cancel the whole set
        of path computation requests.
        The aim of the SyncTimer is to avoid the storage
        of unused synchronized requests should one of
        them get lost for some reasons (for example, a
        misbehaving PCC).
        Zero means that the PCEP entity does not use the
        SyncTimer.";
    reference
        "RFC 5440: Path Computation Element (PCE)
        Communication Protocol (PCEP)";
}

leaf request-timer{
    type uint32 {
        range "1..65535";
    }
    units "seconds";
    description
        "The maximum time that the PCEP entity will wait
        for a response to a PCReq message.";
}

leaf max-sessions{
    type uint32;
    description
        "Maximum number of sessions involving this PCEP
        entity that can exist at any time.";
}

leaf max-unknown-reqs{
    type uint32;
    default 5;
    description
        "The maximum number of unrecognized requests and
        replies that any session on this PCEP entity is
```

```
        willing to accept per minute before terminating
        the session.
        A PCRep message contains an unrecognized reply
        if it contains an RP object whose request ID
        does not correspond to any in-progress request
        sent by this PCEP entity.
        A PCReq message contains an unrecognized request
        if it contains an RP object whose request ID is
        zero.";
    reference
        "RFC 5440: Path Computation Element (PCE)
        Communication Protocol (PCEP)";
}

leaf max-unknown-msgs{
    type uint32;
    default 5;
    description
        "The maximum number of unknown messages that any
        session on this PCEP entity is willing to accept
        per minute before terminating the session.";
    reference
        "RFC 5440: Path Computation Element (PCE)
        Communication Protocol (PCEP)";
}

} // pcep-entity-info

grouping pce-scope{
    description
        "This grouping defines PCE path computation scope
        information which maybe relevant to PCE selection.
        This information corresponds to PCE auto-discovery
        information.";
    reference
        "RFC 5088: OSPF Protocol Extensions for Path
        Computation Element (PCE)
        Discovery
        RFC 5089: IS-IS Protocol Extensions for Path
        Computation Element (PCE)
        Discovery";
    leaf intra-area-scope{
        type boolean;
        default true;
        description
            "PCE can compute intra-area paths.";
    }
    leaf intra-area-pref{
```

```
        type uint8{
            range "0..7";
        }
        description
            "The PCE's preference for intra-area TE LSP
            computation.";
    }
    leaf inter-area-scope{
        type boolean;
        default false;
        description
            "PCE can compute inter-area paths.";
    }
    leaf inter-area-scope-default{
        type boolean;
        default false;
        description
            "PCE can act as a default PCE for inter-area
            path computation.";
    }
    leaf inter-area-pref{
        type uint8{
            range "0..7";
        }
        description
            "The PCE's preference for inter-area TE LSP
            computation.";
    }
    leaf inter-as-scope{
        type boolean;
        default false;
        description
            "PCE can compute inter-AS paths.";
    }
    leaf inter-as-scope-default{
        type boolean;
        default false;
        description
            "PCE can act as a default PCE for inter-AS
            path computation.";
    }
    leaf inter-as-pref{
        type uint8{
            range "0..7";
        }
        description
            "The PCE's preference for inter-AS TE LSP
            computation.";
```

```
    }
    leaf inter-layer-scope{
        type boolean;
        default false;
        description
            "PCE can compute inter-layer paths.";
    }
    leaf inter-layer-pref{
        type uint8{
            range "0..7";
        }
        description
            "The PCE's preference for inter-layer TE LSP
            computation.";
    }
} //pce-scope

grouping domain{
    description
        "This grouping specifies a Domain where the
        PCEP speaker has topology visibility.";
    leaf domain-type{
        type domain-type;
        description
            "The domain type.";
    }
    leaf domain{
        type domain;
        description
            "The domain Information.";
    }
} //domain

grouping capability{
    description
        "This grouping specifies a capability
        information of local PCEP entity. This maybe
        relevant to PCE selection as well. This
        information corresponds to PCE auto-discovery
        information.";
    reference
        "RFC 5088: OSPF Protocol Extensions for Path
        Computation Element (PCE)
        Discovery
        RFC 5089: IS-IS Protocol Extensions for Path
        Computation Element (PCE)
        Discovery";
    leaf gmpls{
```

```
        if-feature gmpls;
        type boolean;
        description
            "Path computation with GMPLS link
            constraints.";
    }
    leaf bi-dir{
        type boolean;
        description
            "Bidirectional path computation.";
    }
    leaf diverse{
        type boolean;
        description
            "Diverse path computation.";
    }
    leaf load-balance{
        type boolean;
        description
            "Load-balanced path computation.";
    }
    leaf synchronize{
        if-feature svec;
        type boolean;
        description
            "Synchronized paths computation.";
    }
    leaf objective-function{
        if-feature obj-fn;
        type boolean;
        description
            "Support for multiple objective functions.";
    }
    leaf add-path-constraint{
        type boolean;
        description
            "Support for additive path constraints (max
            hop count, etc.).";
    }
    leaf prioritization{
        type boolean;
        description
            "Support for request prioritization.";
    }
    leaf multi-request{
        type boolean;
        description
            "Support for multiple requests per message.";
```

```
    }
    leaf gco{
        if-feature gco;
        type boolean;
        description
            "Support for Global Concurrent Optimization
            (GCO).";
    }
    leaf p2mp{
        if-feature p2mp;
        type boolean;
        description
            "Support for P2MP path computation.";
    }
}

container stateful{
    if-feature stateful;
    description
        "If stateful PCE feature is present";
    leaf enabled{
        type boolean;
        description
            "Enabled or Disabled";
    }
    leaf active{
        type boolean;
        description
            "Support for active stateful PCE.";
    }
    leaf pce-initiated{
        if-feature pce-initiated;
        type boolean;
        description
            "Support for PCE-initiated LSP.";
    }
}

container sr{
    if-feature sr;
    description
        "If segment routing is supported";
    leaf enabled{
        type boolean;
        description
            "Enabled or Disabled";
    }
    leaf msd{ /*should be in MPLS yang model (?)*/
        type uint8;
        must "((../../role == 'pcc')" +
```



```

        " or " +
        "(../../role == 'pcc-and-pce'))))"
    {
        error-message
            "The PCEP entity must be PCC";
        description
            "When PCEP entity is PCC for
            MSD to be applicable";
    }
        description
            "Maximum SID Depth";
    }
}
} //capability

grouping info{
    description
        "This grouping specifies all information which
        maybe relevant to both PCC and PCE.
        This information corresponds to PCE auto-discovery
        information.";
    container domain{
        description
            "The local domain for the PCEP entity";
        list domain{
            key "domain-type domain";
            description
                "The local domain.";
            uses domain{
                description
                    "The local domain for the PCEP entity.";
            }
        }
    }
    container capability{
        description
            "The PCEP entity capability";
        uses capability{
            description
                "The PCEP entity supported
                capabilities.";
        }
    }
} //info

grouping pce-info{
    description
        "This grouping specifies all PCE information

```

```
        which maybe relevant to the PCE selection.
        This information corresponds to PCE auto-discovery
        information.";
    container scope{
        description
            "The path computation scope";
        uses pce-scope;
    }

    container neigh-domains{
        description
            "The list of neighbour PCE-Domain
            toward which a PCE can compute
            paths";
        list domain{
            key "domain-type domain";

            description
                "The neighbour domain.";
            uses domain{
                description
                    "The PCE neighbour domain.";
            }
        }
    }
} //pce-info

grouping pcep-stats{
    description
        "This grouping defines statistics for PCEP. It is used
        for both peer and current session.";
    leaf avg-rsp-time{
        type uint32;
        units "milliseconds";
        must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
            " or " +
            "(/pcep-state/entity/peers/peer/role = 'pcc'" +
            " and avg-rsp-time = 0))" {
            error-message
                "Invalid average response time";
            description
                "If role is pcc then this leaf is meaningless
                and is set to zero.";
        }
    }
    description
        "The average response time.
        If an average response time has not been
        calculated then this leaf has the value zero.";
```

```
}

leaf lwm-rsp-time{
  type uint32;
  units "milliseconds";
  must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
    " or " +
    "(/pcep-state/entity/peers/peer/role = 'pcc'" +
    " and lwm-rsp-time = 0))" {
    error-message
      "Invalid smallest (low-water mark)
      response time";
    description
      "If role is pcc then this leaf is meaningless
      and is set to zero.";
  }
  description
    "The smallest (low-water mark) response time seen.
    If no responses have been received then this
    leaf has the value zero.";
}

leaf hwm-rsp-time{
  type uint32;
  units "milliseconds";
  must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
    " or " +
    "(/pcep-state/entity/peers/peer/role = 'pcc'" +
    " and hwm-rsp-time = 0))" {
    error-message
      "Invalid greatest (high-water mark)
      response time seen";
    description
      "If role is pcc then this field is
      meaningless and is set to zero.";
  }
  description
    "The greatest (high-water mark) response time seen.
    If no responses have been received then this object
    has the value zero.";
}

leaf num-pcreq-sent{
  type yang:counter32;
  description
    "The number of PCReq messages sent.";
}
```

```
leaf num-pcreq-rcvd{
  type yang:counter32;
  description
    "The number of PCReq messages received.";
}

leaf num-pcrep-sent{
  type yang:counter32;
  description
    "The number of PCRep messages sent.";
}

leaf num-pcrep-rcvd{
  type yang:counter32;
  description
    "The number of PCRep messages received.";
}

leaf num-pcerr-sent{
  type yang:counter32;
  description
    "The number of PCErr messages sent.";
}

leaf num-pcerr-rcvd{
  type yang:counter32;
  description
    "The number of PCErr messages received.";
}

leaf num-pcntf-sent{
  type yang:counter32;
  description
    "The number of PCNtf messages sent.";
}

leaf num-pcntf-rcvd{
  type yang:counter32;
  description
    "The number of PCNtf messages received.";
}

leaf num-keepalive-sent{
  type yang:counter32;
  description
    "The number of Keepalive messages sent.";
}
```

```
leaf num-keepalive-rcvd{
  type yang:counter32;
  description
    "The number of Keepalive messages received.";
}

leaf num-unknown-rcvd{
  type yang:counter32;
  description
    "The number of unknown messages received.";
}

leaf num-corrupt-rcvd{
  type yang:counter32;
  description
    "The number of corrupted PCEP message received.";
}

leaf num-req-sent{
  type yang:counter32;
  description
    "The number of requests sent. A request corresponds
    1:1 with an RP object in a PCReq message. This might
    be greater than num-pcreq-sent because multiple
    requests can be batched into a single PCReq
    message.";
}

leaf num-req-sent-pend-rep{
  type yang:counter32;
  description
    "The number of requests that have been sent for
    which a response is still pending.";
}

leaf num-req-sent-ero-rcvd{
  type yang:counter32;
  description
    "The number of requests that have been sent for
    which a response with an ERO object was received.
    Such responses indicate that a path was
    successfully computed by the peer.";
}

leaf num-req-sent-nopath-rcvd{
  type yang:counter32;
  description
    "The number of requests that have been sent for
```

```
        which a response with a NO-PATH object was
        received. Such responses indicate that the peer
        could not find a path to satisfy the
        request.";
    }

    leaf num-req-sent-cancel-rcvd{
        type yang:counter32;
        description
            "The number of requests that were cancelled with
            a PCNtf message.
            This might be different than num-pcntf-rcvd because
            not all PCNtf messages are used to cancel requests,
            and a single PCNtf message can cancel multiple
            requests.";
    }

    leaf num-req-sent-error-rcvd{
        type yang:counter32;
        description
            "The number of requests that were rejected with a
            PCErr message.
            This might be different than num-pcerr-rcvd because
            not all PCErr messages are used to reject requests,
            and a single PCErr message can reject multiple
            requests.";
    }

    leaf num-req-sent-timeout{
        type yang:counter32;
        description
            "The number of requests that have been sent to a peer
            and have been abandoned because the peer has taken too
            long to respond to them.";
    }

    leaf num-req-sent-cancel-sent{
        type yang:counter32;
        description
            "The number of requests that were sent to the peer and
            explicitly cancelled by the local PCEP entity sending
            a PCNtf.";
    }

    leaf num-req-rcvd{
        type yang:counter32;
        description
            "The number of requests received. A request
```

```
        corresponds 1:1 with an RP object in a PCReq
        message.
        This might be greater than num-pcreq-rcvd because
        multiple requests can be batched into a single
        PCReq message.";
    }

    leaf num-req-rcvd-pend-rep{
        type yang:counter32;
        description
            "The number of requests that have been received for
            which a response is still pending.";
    }

    leaf num-req-rcvd-ero-sent{
        type yang:counter32;
        description
            "The number of requests that have been received for
            which a response with an ERO object was sent. Such
            responses indicate that a path was successfully
            computed by the local PCEP entity.";
    }

    leaf num-req-rcvd-nopath-sent{
        type yang:counter32;
        description
            "The number of requests that have been received for
            which a response with a NO-PATH object was sent. Such
            responses indicate that the local PCEP entity could
            not find a path to satisfy the request.";
    }

    leaf num-req-rcvd-cancel-sent{
        type yang:counter32;
        description
            "The number of requests received that were cancelled
            by the local PCEP entity sending a PCNtf message.
            This might be different than num-pcntf-sent because
            not all PCNtf messages are used to cancel requests,
            and a single PCNtf message can cancel multiple
            requests.";
    }

    leaf num-req-rcvd-error-sent{
        type yang:counter32;
        description
            "The number of requests received that were cancelled
            by the local PCEP entity sending a PCErr message.
```

```
        This might be different than num-pcerr-sent because
        not all PCErr messages are used to cancel requests,
        and a single PCErr message can cancel multiple
        requests.";
    }

    leaf num-req-rcvd-cancel-rcvd{
        type yang:counter32;
        description
            "The number of requests that were received from the
            peer and explicitly cancelled by the peer sending
            a PCNtf.";
    }

    leaf num-rep-rcvd-unknown{
        type yang:counter32;
        description
            "The number of responses to unknown requests
            received. A response to an unknown request is a
            response whose RP object does not contain the
            request ID of any request that is currently
            outstanding on the session.";
    }

    leaf num-req-rcvd-unknown{
        type yang:counter32;
        description
            "The number of unknown requests that have been
            received. An unknown request is a request
            whose RP object contains a request ID of
            zero.";
    }

    container svec{
        if-feature svec;
        description
            "If synchronized path computation is supported";
        leaf num-svec-sent{
            type yang:counter32;
            description
                "The number of SVEC objects sent in PCReq messages.
                An SVEC object represents a set of synchronized
                requests.";
        }

        leaf num-svec-req-sent{
            type yang:counter32;
            description
```



```
        "The number of requests sent that appeared in one
        or more SVEC objects.";
    }

    leaf num-svec-rcvd{
        type yang:counter32;
        description
            "The number of SVEC objects received in PCReq
            messages. An SVEC object represents a set of
            synchronized requests.";
    }

    leaf num-svec-req-rcvd{
        type yang:counter32;
        description
            "The number of requests received that appeared
            in one or more SVEC objects.";
    }
}
container stateful{
    if-feature stateful;
    description
        "Stateful PCE related statistics";
    leaf num-pcrpt-sent{
        type yang:counter32;
        description
            "The number of PCRpt messages sent.";
    }

    leaf num-pcrpt-rcvd{
        type yang:counter32;
        description
            "The number of PCRpt messages received.";
    }

    leaf num-pcupd-sent{
        type yang:counter32;
        description
            "The number of PCUpd messages sent.";
    }

    leaf num-pcupd-rcvd{
        type yang:counter32;
        description
            "The number of PCUpd messages received.";
    }

    leaf num-rpt-sent{
```

```
    type yang:counter32;
    description
      "The number of LSP Reports sent.  A LSP report
       corresponds 1:1 with an LSP object in a PCRpt
       message. This might be greater than
       num-pcrpt-sent because multiple reports can
       be batched into a single PCRpt message.";
  }

  leaf num-rpt-rcvd{
    type yang:counter32;
    description
      "The number of LSP Reports received.  A LSP report
       corresponds 1:1 with an LSP object in a PCRpt
       message.
       This might be greater than num-pcrpt-rcvd because
       multiple reports can be batched into a single
       PCRpt message.";
  }

  leaf num-rpt-rcvd-error-sent{
    type yang:counter32;
    description
      "The number of reports of LSPs received that were
       responded by the local PCEP entity by sending a
       PCErr message.";
  }

  leaf num-upd-sent{
    type yang:counter32;
    description
      "The number of LSP updates sent.  A LSP update
       corresponds 1:1 with an LSP object in a PCUpd
       message. This might be greater than
       num-pcupd-sent because multiple updates can
       be batched into a single PCUpd message.";
  }

  leaf num-upd-rcvd{
    type yang:counter32;
    description
      "The number of LSP Updates received.  A LSP update
       corresponds 1:1 with an LSP object in a PCUpd
       message.
       This might be greater than num-pcupd-rcvd because
       multiple updates can be batched into a single
       PCUpd message.";
  }
```

```
leaf num-upd-rcvd-unknown{
  type yang:counter32;
  description
    "The number of updates to unknown LSPs
    received. An update to an unknown LSP is a
    update whose LSP object does not contain the
    PLSP-ID of any LSP that is currently
    present.";
}

leaf num-upd-rcvd-undelegated{
  type yang:counter32;
  description
    "The number of updates to not delegated LSPs
    received. An update to an undelegated LSP is a
    update whose LSP object does not contain the
    PLSP-ID of any LSP that is currently
    delegated to current PCEP session.";
}

leaf num-upd-rcvd-error-sent{
  type yang:counter32;
  description
    "The number of updates to LSPs received that were
    responded by the local PCEP entity by sending a
    PCErr message.";
}

container initiation {
  if-feature pce-initiated;
  description
    "PCE-Initiated related statistics";
  leaf num-pcinitiate-sent{
    type yang:counter32;
    description
      "The number of PCInitiate messages sent.";
  }

  leaf num-pcinitiate-rcvd{
    type yang:counter32;
    description
      "The number of PCInitiate messages received.";
  }

  leaf num-initiate-sent{
    type yang:counter32;
    description
      "The number of LSP Initiation sent via PCE.
      A LSP initiation corresponds 1:1 with an LSP
```

```
        object in a PCInitiate message. This might be
        greater than num-pcinitiate-sent because
        multiple initiations can be batched into a
        single PCInitiate message.";
    }

    leaf num-initiate-rcvd{
        type yang:counter32;
        description
            "The number of LSP Initiation received from
            PCE. A LSP initiation corresponds 1:1 with
            an LSP object in a PCInitiate message. This
            might be greater than num-pcinitiate-rcvd
            because multiple initiations can be batched
            into a single PCInitiate message.";
    }

    leaf num-initiate-rcvd-error-sent{
        type yang:counter32;
        description
            "The number of initiations of LSPs received
            that were responded by the local PCEP entity
            by sending a PCErr message.";
    }
}

} //pcep-stats

grouping lsp-state{
    description
        "This grouping defines the attributes for LSP in LSP-DB.
        These are the attributes specifically from the PCEP
        perspective";
    leaf plsp-id{
        type uint32{
            range "1..1048575";
        }
        description
            "A PCEP-specific identifier for the LSP. A PCC
            creates a unique PLSP-ID for each LSP that is
            constant for the lifetime of a PCEP session.
            PLSP-ID is 20 bits with 0 and 0xFFFFF are
            reserved";
    }
    leaf pcc-id{
        type inet:ip-address;
        description
            "The local internet address of the PCC, that
```

```
        generated the PLSP-ID.";
    }

    container lsp-ref {
        description
            "reference to ietf-te lsp state";

        leaf source {
            type leafref {
                path "/te:te/te:lsps-state/te:lsp/te:source";
            }
            description
                "Tunnel sender address extracted from
                 SENDER_TEMPLATE object";
            reference "RFC3209";
        }
        leaf destination {
            type leafref {
                path "/te:te/te:lsps-state/te:lsp/te:"
                    + "destination";
            }
            description
                "Tunnel endpoint address extracted from
                 SESSION object";
            reference "RFC3209";
        }
    }
    leaf tunnel-id {
        type leafref {
            path "/te:te/te:lsps-state/te:lsp/te:tunnel-id";
        }
        description
            "Tunnel identifier used in the SESSION
             that remains constant over the life
             of the tunnel.";
        reference "RFC3209";
    }
    leaf lsp-id {
        type leafref {
            path "/te:te/te:lsps-state/te:lsp/te:lsp-id";
        }
        description
            "Identifier used in the SENDER_TEMPLATE
             and the FILTER_SPEC that can be changed
             to allow a sender to share resources with
             itself.";
        reference "RFC3209";
    }
    leaf extended-tunnel-id {
```

```
        type leafref {
            path "/te:te/te:lsps-state/te:lsp/te:"
              + "extended-tunnel-id";
        }
        description
            "Extended Tunnel ID of the LSP.";
        reference "RFC3209";
    }
    leaf type {
        type leafref {
            path "/te:te/te:lsps-state/te:lsp/te:type";
        }
        description "LSP type P2P or P2MP";
    }
}

leaf admin-state{
    type boolean;
    description
        "The desired operational state";
}
leaf operational-state{
    type operational-state;
    description
        "The operational status of the LSP";
}
container delegated{
    description
        "The delegation related parameters";
    leaf enabled{
        type boolean;
        description
            "LSP is delegated or not";
    }
    leaf pce{
        type leafref {
            path "/pcep-state/entity/peers/peer/addr";
        }
        must "((../enabled == true)" +
            " and " +
            "((../role == 'pcc'))" +
            " or " +
            "((../role == 'pcc-and-pce')))"
        {
            error-message
                "The PCEP entity must be PCC
                and the LSP be delegated";
            description

```

```
        "When PCEP entity is PCC for
        delegated LSP";
    }
    description
        "The reference to the PCE peer to
        which LSP is delegated";
    }
    leaf srp-id{
        type uint32;
        description
            "The last SRP-ID-number associated with this
            LSP.";
    }
}
container initiation {
    if-feature pce-initiated;
    description
        "The PCE initiation related parameters";
    leaf enabled{
        type boolean;
        description
            "LSP is PCE-initiated or not";
    }
    leaf pce{
        type leafref {
            path "/pcep-state/entity/peers/peer/addr";
        }
        must "(../enabled == true)"
        {
            error-message
                "The LSP must be PCE-Initiated";
            description
                "When the LSP must be PCE-Initiated";
        }
        description
            "The reference to the PCE
            that initiated this LSP";
    }
}
leaf symbolic-path-name{
    type string;
    description
        "The symbolic path name associated with the LSP.";
}
leaf last-error{
    type lsp-error;
    description
        "The last error for the LSP.";
```

```
    }
    leaf pst{
        type pst;
        default "rsvp-te";
        description
            "The Path Setup Type";
    }
} //lsp-state

grouping notification-instance-hdr {
    description
        "This group describes common instance specific data
        for notifications.";

    leaf peer-addr {
        type leafref {
            path "/pcep-state/entity/peers/peer/addr";
        }
        description
            "Reference to peer address";
    }
} // notification-instance-hdr

grouping notification-session-hdr {
    description
        "This group describes common session instance specific
        data for notifications.";

    leaf session-initiator {
        type leafref {
            path "/pcep-state/entity/peers/peer/sessions/" +
                "session/initiator";
        }
        description
            "Reference to pcep session initiator leaf";
    }
} // notification-session-hdr

grouping stateful-pce-parameter {
    description
        "This group describes stateful PCE specific
        parameters.";
    leaf state-timeout{
        type uint32;
        units "seconds";
    }
}
```



```
        description
            "When a PCEP session is terminated, a PCC
            waits for this time period before flushing
            LSP state associated with that PCEP session
            and reverting to operator-defined default
            parameters or behaviours.";
    }
    leaf redelegation-timeout{
        type uint32;
        units "seconds";
        must "((../role == 'pcc')" +
            " or " +
            "(../role == 'pcc-and-pce'))"
        {
            error-message "The PCEP entity must be PCC";
            description
                "When PCEP entity is PCC";
        }
        description
            "When a PCEP session is terminated, a PCC
            waits for this time period before revoking
            LSP delegation to a PCE and attempting to
            redelegate LSPs associated with the
            terminated PCEP session to an alternate
            PCE.";
    }
    leaf rpt-non-pcep-lsp{
        type boolean;
        must "((../role == 'pcc')" +
            " or " +
            "(../role == 'pcc-and-pce'))"
        {
            error-message "The PCEP entity must be PCC";
            description
                "When PCEP entity is PCC";
        }
        description
            "If set, a PCC reports LSPs that are not
            controlled by any PCE (for example, LSPs
            that are statically configured at the
            PCC). ";
    }
}

grouping authentication {
    description "Authentication Information";
    choice auth-type-selection {
```

```
description
  "Options for expressing authentication setting.";
case auth-key-chain {
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
      "key-chain name.";
  }
}
case auth-key {
  leaf key {
    type string;
    description
      "Key string in ASCII format.";
  }
  container crypto-algorithm {
    uses key-chain:crypto-algorithm-types;
    description
      "Cryptographic algorithm associated
       with key.";
  }
}
case auth-tls {
  if-feature tls;
  container tls {
    description
      "TLS related information - TBD";
  }
}
}

grouping association {
  description
    "Generic Association parameters";
  leaf type {
    type "assoc-type";
    description
      "The PCEP association type";
  }
  leaf id {
    type uint16;
    description
      "PCEP Association ID";
  }
  leaf source {
    type inet:ip-address;
    description
```

```
        "PCEP Association Source.";
    }
    leaf global-source {
        type uint32;
        description
            "PCEP Association Global
             Source.";
    }
    leaf extended-id{
        type string;
        description
            "Additional information to
             support unique identification.";
    }
}
grouping association-ref {
    description
        "Generic Association parameters";
    leaf id {
        type leafref {
            path "/pcep-state/entity/lsp-db/"
                + "association-list/id";
        }
        description
            "PCEP Association ID";
    }
    leaf source {
        type leafref {
            path "/pcep-state/entity/lsp-db/"
                + "association-list/source";
        }
        description
            "PCEP Association Source.";
    }
    leaf global-source {
        type leafref {
            path "/pcep-state/entity/lsp-db/"
                + "association-list/global-source";
        }
        description
            "PCEP Association Global
             Source.";
    }
    leaf extended-id{
        type leafref {
            path "/pcep-state/entity/lsp-db/"
                + "association-list/extended-id";
        }
    }
}
```

```
        description
            "Additional information to
            support unique identification.";
    }
}
/*
 * Configuration data nodes
 */
container pcep{

    presence
        "The PCEP is enabled";

    description
        "Parameters for list of configured PCEP entities
        on the device.";

    container entity {

        description
            "The configured PCEP entity on the device.";

        leaf addr {
            type inet:ip-address;
            mandatory true;
            description
                "The local Internet address of this PCEP
                entity.
                If operating as a PCE server, the PCEP
                entity listens on this address.
                If operating as a PCC, the PCEP entity
                binds outgoing TCP connections to this
                address.
                It is possible for the PCEP entity to
                operate both as a PCC and a PCE Server, in
                which case it uses this address both to
                listen for incoming TCP connections and to
                bind outgoing TCP connections.";
        }

        leaf enabled {
            type boolean;
            default true;
            description
                "The administrative status of this PCEP
                Entity.";
        }
    }
}
```

```

leaf role {
  type pcep-role;
  mandatory true;
  description
    "The role that this entity can play.
    Takes one of the following values.
    - unknown(0): this PCEP Entity role is not
      known.
    - pcc(1): this PCEP Entity is a PCC.
    - pce(2): this PCEP Entity is a PCE.
    - pcc-and-pce(3): this PCEP Entity is both
      a PCC and a PCE.";
}

leaf description {
  type string;
  description
    "Description of the PCEP entity configured
    by the user";
}

uses info {
  description
    "Local PCEP entity information";
}

container pce-info {
  must "((../role == 'pce')" +
    " or " +
    "(../role == 'pcc-and-pce'))"
  {
    error-message "The PCEP entity must be PCE";
    description
      "When PCEP entity is PCE";
  }
  uses pce-info {
    description
      "Local PCE information";
  }
  uses authentication {
    description
      "Local PCE authentication inform
ation";
  }
}

description
  "The Local PCE Entity PCE information";

```

```
    }

    uses pcep-entity-info {
        description
            "The configuration related to the PCEP
            entity.";
    }

    leaf pcep-notification-max-rate {
        type uint32;
        mandatory true;
        description
            "This variable indicates the maximum number of
            notifications issued per second. If events occur
            more rapidly, the implementation may simply fail
            to emit these notifications during that period,
            or may queue them until an appropriate time. A
            value of 0 means no notifications are emitted
            and all should be discarded (that is, not
            queued).";
    }

    container stateful-parameter{
        if-feature stateful;
        must "(../info/capability/stateful/active == true)"
        {
            error-message
                "The Active Stateful PCE must be enabled";
            description
                "When PCEP entity is active stateful
                enabled";
        }
        uses stateful-pce-parameter;

        description
            "The configured stateful parameters";
    }

    container peers{
        must "((../role == 'pcc') +
            " or " +
            "(../role == 'pcc-and-pce'))"
        {
            error-message
                "The PCEP entity must be PCC";
        }
    }
```

```
        description
            "When PCEP entity is PCC, as remote
            PCE peers are configured.";
    }
    description
        "The list of configured peers for the
        entity (remote PCE)";
    list peer{
        key "addr";

        description
            "The peer configured for the entity.
            (remote PCE)";

        leaf addr {
            type inet:ip-address;
            description
                "The local Internet address of this
                PCEP peer.";
        }

        leaf description {
            type string;
            description
                "Description of the PCEP peer
                configured by the user";
        }
        uses info {
            description
                "PCE Peer information";
        }
        uses pce-info {
            description
                "PCE Peer information";
        }
    }

    leaf delegation-pref{
        if-feature stateful;
        type uint8{
            range "0..7";
        }
        must "(../../info/capability/stateful/active"
            + " == true)"
        {
            error-message
                "The Active Stateful PCE must be
                enabled";
            description

```

```

        "When PCEP entity is active stateful
        enabled";
    }
    description
        "The PCE peer delegation preference.";
    }
    uses authentication {
        description
            "PCE Peer authentication";
    }
    } //peer
} //peers
} //entity
} //pcep

/*
 * Operational data nodes
 */

container pcep-state{
    config false;
    description
        "The list of operational PCEP entities on the
        device.";

    container entity{
        description
            "The operational PCEP entity on the device.";

        leaf addr {
            type inet:ip-address;
            description
                "The local Internet address of this PCEP
                entity.
                If operating as a PCE server, the PCEP
                entity listens on this address.
                If operating as a PCC, the PCEP entity
                binds outgoing TCP connections to this
                address.
                It is possible for the PCEP entity to
                operate both as a PCC and a PCE Server, in
                which case it uses this address both to
                listen for incoming TCP connections and to
                bind outgoing TCP connections.";
        }

        leaf index{
            type uint32;

```



```
        description
            "The index of the operational PCEP
            entity";
    }

    leaf admin-status {
        type pcep-admin-status;
        description
            "The administrative status of this PCEP Entity.
            This is the desired operational status as
            currently set by an operator or by default in
            the implementation. The value of enabled
            represents the current status of an attempt
            to reach this desired status.";
    }

    leaf oper-status {
        type pcep-admin-status;
        description
            "The operational status of the PCEP entity.
            Takes one of the following values.
            - oper-status-up(1): the PCEP entity is
              active.
            - oper-status-down(2): the PCEP entity is
              inactive.
            - oper-status-going-up(3): the PCEP entity is
              activating.
            - oper-status-going-down(4): the PCEP entity is
              deactivating.
            - oper-status-failed(5): the PCEP entity has
              failed and will recover when possible.
            - oper-status-failed-perm(6): the PCEP entity
              has failed and will not recover without
              operator intervention.";
    }

    leaf role {
        type pcep-role;
        description
            "The role that this entity can play.
            Takes one of the following values.
            - unknown(0): this PCEP entity role is
              not known.
            - pcc(1): this PCEP entity is a PCC.
            - pce(2): this PCEP entity is a PCE.
            - pcc-and-pce(3): this PCEP entity is
              both a PCC and a PCE.";
```

```

    }
    uses info {
        description
            "Local PCEP entity information";
    }
    container pce-info {
        when "((../role == 'pce') +
            " or " +
            "(../role == 'pcc-and-pce'))"
        {
            description
                "When PCEP entity is PCE";
        }
        uses pce-info {
            description
                "Local PCE information";
        }
        uses authentication {
            description
                "Local PCE authentication inform
ation";
        }
        description
            "The Local PCE Entity PCE information";
    }
    uses pcep-entity-info{
        description
            "The operational information related to the
            PCEP entity.";
    }
    container stateful-parameter{
        if-feature stateful;
        must "(../info/capability/stateful/active == true)"
        {
            error-message
                "The Active Stateful PCE must be enabled";
            description
                "When PCEP entity is active stateful
                enabled";
        }
        uses stateful-pce-parameter;
        description
            "The operational stateful parameters";
    }

```

```
container lsp-db{
  if-feature stateful;
  description
    "The LSP-DB";
  list association-list {
    key "id source global-source extended-id";
    description
      "List of all PCEP associations";
    uses association {
      description
        "The Association attributes";
    }
    list lsp {
      key "plsp-id pcc-id";
      description
        "List of all LSP in this association";
      leaf plsp-id {
        type leafref {
          path "/pcep-state/entity/lsp-db/"
            + "lsp/plsp-id";
        }
        description
          "Reference to PLSP-ID in LSP-DB";
      }
      leaf pcc-id {
        type leafref {
          path "/pcep-state/entity/lsp-db/"
            + "lsp/pcc-id";
        }
        description
          "Reference to PCC-ID in LSP-DB";
      }
    }
  }
}
list lsp{
  key "plsp-id pcc-id";
  description
    "List of all LSPs in LSP-DB";
  uses lsp-state{
    description
      "The PCEP specific attributes for
        LSP-DB.";
  }
  list association-list {
    key "id source global-source extended-id";
    description
      "List of all PCEP associations";
    uses association-ref {
```

```
        description
            "Reference to the Association
            attributes";
    }
}

}
}
container peers{
    description
        "The list of peers for the entity";

    list peer{
        key "addr";

        description
            "The peer for the entity.";

        leaf addr {
            type inet:ip-address;
            description
                "The local Internet address of this PCEP
                peer.";
        }

        leaf role {
            type pcep-role;
            description
                "The role of the PCEP Peer.
                Takes one of the following values.
                - unknown(0): this PCEP peer role
                  is not known.
                - pcc(1): this PCEP peer is a PCC.
                - pce(2): this PCEP peer is a PCE.
                - pcc-and-pce(3): this PCEP peer
                  is both a PCC and a PCE.";
        }

        uses info {
            description
                "PCEP peer information";
        }

        container pce-info {
            when "((../role == 'pce')" +
            " or " +
```

```
    "(../role == 'pcc-and-pce'))"
  {
    description
      "When PCEP entity is PCE";
  }
  uses pce-info {
    description
      "PCE Peer information";
  }
  description
    "The PCE Peer information";
}

leaf delegation-pref{
  if-feature stateful;
  type uint8{
    range "0..7";
  }
  must "((../role == 'pcc')" +
    " or " +
    "(../role == 'pcc-and-pce'))"
  {
    error-message
      "The PCEP entity must be PCC";
    description
      "When PCEP entity is PCC";
  }
  must "(../info/capability/stateful/active"
    + " == true)"
  {
    error-message
      "The Active Stateful PCE must be
        enabled";
    description
      "When PCEP entity is active stateful
        enabled";
  }
  description
    "The PCE peer delegation preference.";
}

uses authentication {
  description
    "PCE Peer authentication";
}

leaf discontinuity-time {
  type yang:timestamp;
```

```
        description
            "The timestamp of the time when the
             information and statistics were
             last reset.";
    }

    leaf initiate-session {
        type boolean;
        description
            "Indicates whether the local PCEP
             entity initiates sessions to this peer,
             or waits for the peer to initiate a
             session.";
    }

    leaf session-exists{
        type boolean;
        description
            "Indicates whether a session with
             this peer currently exists.";
    }

    leaf num-sess-setup-ok{
        type yang:counter32;
        description
            "The number of PCEP sessions successfully
             successfully established with the peer,
             including any current session. This
             counter is incremented each time a
             session with this peer is successfully
             established.";
    }

    leaf num-sess-setup-fail{
        type yang:counter32;
        description
            "The number of PCEP sessions with the peer
             that have been attempted but failed
             before being fully established. This
             counter is incremented each time a
             session retry to this peer fails.";
    }

    leaf session-up-time{
        type yang:timestamp;
        must "(../num-sess-setup-ok != 0 or " +
            "(../num-sess-setup-ok = 0 and " +
            "session-up-time = 0))" {
```

```
        error-message
            "Invalid Session Up timestamp";
        description
            "If num-sess-setup-ok is zero,
             then this leaf contains zero.";
    }
    description
        "The timestamp value of the last time a
         session with this peer was successfully
         established.";
}

leaf session-fail-time{
    type yang:timestamp;
    must "(../num-sess-setup-fail != 0 or " +
        "(../num-sess-setup-fail = 0 and " +
        "session-fail-time = 0))" {
        error-message
            "Invalid Session Fail timestamp";
        description
            "If num-sess-setup-fail is zero,
             then this leaf contains zero.";
    }
    description
        "The timestamp value of the last time a
         session with this peer failed to be
         established.";
}

leaf session-fail-up-time{
    type yang:timestamp;
    must "(../num-sess-setup-ok != 0 or " +
        "(../num-sess-setup-ok = 0 and " +
        "session-fail-up-time = 0))" {
        error-message
            "Invalid Session Fail from
             Up timestamp";
        description
            "If num-sess-setup-ok is zero,
             then this leaf contains zero.";
    }
    description
        "The timestamp value of the last time a
         session with this peer failed from
         active.";
}

container pcep-stats {
```

```
description
  "The container for all statistics at peer
  level.";
uses pcep-stats{
  description
    "Since PCEP sessions can be
    ephemeral, the peer statistics tracks
    a peer even when no PCEP session
    currently exists to that peer. The
    statistics contained are an aggregate
    of the statistics for all successive
    sessions to that peer.";
}

leaf num-req-sent-closed{
  type yang:counter32;
  description
    "The number of requests that were
    sent to the peer and implicitly
    cancelled when the session they were
    sent over was closed.";
}

leaf num-req-rcvd-closed{
  type yang:counter32;
  description
    "The number of requests that were
    received from the peer and
    implicitly cancelled when the
    session they were received over
    was closed.";
}
} //pcep-stats
```

```
container sessions {
  description
    "This entry represents a single PCEP
    session in which the local PCEP entity
    participates.
    This entry exists only if the
    corresponding PCEP session has been
    initialized by some event, such as
    manual user configuration, auto-
    discovery of a peer, or an incoming
    TCP connection.";
```



```
list session {
  key "initiator";

  description
    "The list of sessions, note that
    for a time being two sessions
    may exist for a peer";

  leaf initiator {
    type pcep-initiator;
    description
      "The initiator of the session,
      that is, whether the TCP
      connection was initiated by
      the local PCEP entity or the
      peer.
      There is a window during
      session initialization where
      two sessions can exist between
      a pair of PCEP speakers, each
      initiated by one of the
      speakers. One of these
      sessions is always discarded
      before it leaves OpenWait state.
      However, before it is discarded,
      two sessions to the given peer
      appear transiently in this MIB
      module. The sessions are
      distinguished by who initiated
      them, and so this field is the
      key.";
  }

  leaf state-last-change {
    type yang:timestamp;
    description
      "The timestamp value at the
      time this session entered its
      current state as denoted by
      the state leaf.";
  }

  leaf state {
    type pcep-sess-state;
    description
      "The current state of the
      session.
      The set of possible states
```

```
        excludes the idle state since
        entries do not exist in the
        idle state.";
    }

    leaf session-creation {
        type yang:timestamp;
        description
            "The timestamp value at the
            time this session was
            created.";
    }

    leaf connect-retry {
        type yang:counter32;
        description
            "The number of times that the
            local PCEP entity has
            attempted to establish a TCP
            connection for this session
            without success. The PCEP
            entity gives up when this
            reaches connect-max-retry.";
    }

    leaf local-id {
        type uint32 {
            range "0..255";
        }
        description
            "The value of the PCEP session
            ID used by the local PCEP
            entity in the Open message
            for this session.
            If state is tcp-pending then
            this is the session ID that
            will be used in the Open
            message. Otherwise, this is
            the session ID that was sent
            in the Open message.";
    }

    leaf remote-id {
        type uint32 {
            range "0..255";
        }
        must "((../state != 'tcp-pending'" +
            "and " +
```

```
        "../state != 'open-wait' )" +
        "or " +
        "((../state = 'tcp-pending'" +
        " or " +
        "../state = 'open-wait' )" +
        "and remote-id = 0))" {
            error-message
                "Invalid remote-id";
            description
                "If state is tcp-pending
                 or open-wait then this
                 leaf is not used and
                 MUST be set to zero.";
        }
    description
        "The value of the PCEP session
         ID used by the peer in its
         Open message for this
         session.";
}

leaf keepalive-timer {
    type uint32 {
        range "0..255";
    }
    units "seconds";
    must "(../state = 'session-up'" +
        "or " +
        "(../state != 'session-up'" +
        "and keepalive-timer = 0))" {
        error-message
            "Invalid keepalive
             timer";
        description
            "This field is used if
             and only if state is
             session-up. Otherwise,
             it is not used and
             MUST be set to
             zero.";
    }
    description
        "The agreed maximum interval at
         which the local PCEP entity
         transmits PCEP messages on this
         PCEP session. Zero means that
         the local PCEP entity never
         sends Keepalives on this
```

```
        session.";
    }

    leaf peer-keepalive-timer {
        type uint32 {
            range "0..255";
        }
        units "seconds";
        must "(../state = 'session-up' " +
            "or " +
            "(../state != 'session-up' " +
            "and " +
            "peer-keepalive-timer = 0))" {
            error-message
                "Invalid Peer keepalive
                timer";
            description
                "This field is used if
                and only if state is
                session-up. Otherwise,
                it is not used and MUST
                be set to zero.";
        }
        description
            "The agreed maximum interval at
            which the peer transmits PCEP
            messages on this PCEP session.
            Zero means that the peer never
            sends Keepalives on this
            session.";
    }

    leaf dead-timer {
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "The dead timer interval for
            this PCEP session.";
    }

    leaf peer-dead-timer {
        type uint32 {
            range "0..255";
        }
        units "seconds";
        must "(../state != 'tcp-pending' " +
```

```

        "and " +
        "../state != 'open-wait' )" +
        "or " +
        "((../state = 'tcp-pending'" +
        " or " +
        "../state = 'open-wait' )" +
        "and " +
        "peer-dead-timer = 0)))" {
            error-message
                "Invalid Peer Dead
                timer";
            description
                "If state is tcp-
                pending or open-wait
                then this leaf is not
                used and MUST be set to
                zero.";
        }
    description
        "The peer's dead-timer interval
        for this PCEP session.";
}

leaf ka-hold-time-rem {
    type uint32 {
        range "0..255";
    }
    units "seconds";
    must "((../state != 'tcp-pending'" +
        "and " +
        "../state != 'open-wait' )" +
        "or " +
        "((../state = 'tcp-pending'" +
        "or " +
        "../state = 'open-wait' )" +
        "and " +
        "ka-hold-time-rem = 0)))" {
        error-message
            "Invalid Keepalive hold
            time remaining";
        description
            "If state is tcp-pending
            or open-wait then this
            field is not used and
            MUST be set to zero.";
    }
}
description
    "The keep alive hold time

```

```
        remaining for this session.";
    }

    leaf overloaded {
        type boolean;
        description
            "If the local PCEP entity has
            informed the peer that it is
            currently overloaded, then this
            is set to true. Otherwise, it
            is set to false.";
    }

    leaf overload-time {
        type uint32;
        units "seconds";
        must "(../overloaded = true or" +
            "(../overloaded != true and" +
            " overload-time = 0))" {
            error-message
                "Invalid overload-time";
            description
                "This field is only used
                if overloaded is set to
                true. Otherwise, it is
                not used and MUST be set
                to zero.";
        }
        description
            "The interval of time that is
            remaining until the local PCEP
            entity will cease to be
            overloaded on this session.";
    }

    leaf peer-overloaded {
        type boolean;
        description
            "If the peer has informed the
            local PCEP entity that it is
            currently overloaded, then this
            is set to true. Otherwise, it
            is set to false.";
    }

    leaf peer-overload-time {
        type uint32;
        units "seconds";
```

```
must "(../peer-overloaded = true" +
    " or " +
    "(../peer-overloaded != true" +
    " and " +
    "peer-overload-time = 0))" {
    error-message
        "Invalid peer overload
        time";
    description
        "This field is only used
        if peer-overloaded is
        set to true. Otherwise,
        it is not used and MUST
        be set to zero.";
}
description
    "The interval of time that is
    remaining until the peer will
    cease to be overloaded. If it
    is not known how long the peer
    will stay in overloaded state,
    this leaf is set to zero.";
}
leaf lspdb-sync {
    if-feature stateful;
    type sync-state;
    description
        "The LSP-DB state synchronization
        status.";
}
leaf discontinuity-time {
    type yang:timestamp;
    description
        "The timestamp value of the time
        when the statistics were last
        reset.";
}
}
container pcep-stats {
    description
        "The container for all statistics
        at session level.";
    uses pcep-stats{
        description
            "The statistics contained are
            for the current sessions to
            that peer. These are lost
            when the session goes down.
```

```

";
    }
  } // pcep-stats

  } // session
} // sessions
} // peer
} // peers
} // entity
} // pcep-state

/*
 * Notifications
 */
notification pcep-session-up {
  description
    "This notification is sent when the value of
    '/pcep/pcep-state/peers/peer/sessions/session/state'
    enters the 'session-up' state.";

  uses notification-instance-hdr;

  uses notification-session-hdr;

  leaf state-last-change {
    type yang:timestamp;
    description
      "The timestamp value at the time this session entered
      its current state as denoted by the state leaf.";
  }

  leaf state {
    type pcep-sess-state;
    description
      "The current state of the session.
      The set of possible states excludes the idle state
      since entries do not exist in the idle state.";
  }
} // notification

notification pcep-session-down {
  description
    "This notification is sent when the value of
    '/pcep/pcep-state/peers/peer/sessions/session/state'
    leaves the 'session-up' state.";

  uses notification-instance-hdr;

```



```
    leaf session-initiator {
        type pcep-initiator;
        description
            "The initiator of the session.";
    }

    leaf state-last-change {
        type yang:timestamp;
        description
            "The timestamp value at the time this session entered
            its current state as denoted by the state leaf.";
    }

    leaf state {
        type pcep-sess-state;
        description
            "The current state of the session.
            The set of possible states excludes the idle state
            since entries do not exist in the idle state.";
    }
} //notification

notification pcep-session-local-overload {
    description
        "This notification is sent when the local PCEP entity
        enters overload state for a peer.";

    uses notification-instance-hdr;

    uses notification-session-hdr;

    leaf overloaded {
        type boolean;
        description
            "If the local PCEP entity has informed the peer that
            it is currently overloaded, then this is set to
            true. Otherwise, it is set to false.";
    }

    leaf overload-time {
        type uint32;
        units "seconds";
        must "(../overloaded = true or " +
            "(../overloaded != true and " +
            "overload-time = 0))" {
            error-message
                "Invalid overload-time";
        }
        description

```

```
        "This field is only used if overloaded is
        set to true. Otherwise, it is not used
        and MUST be set to zero.";
    }
    description
        "The interval of time that is remaining until the
        local PCEP entity will cease to be overloaded on
        this session.";
}
} //notification

notification pcep-session-local-overload-clear {
    description
        "This notification is sent when the local PCEP entity
        leaves overload state for a peer.";

    uses notification-instance-hdr;

    leaf overloaded {
        type boolean;
        description
            "If the local PCEP entity has informed the peer
            that it is currently overloaded, then this is set
            to true. Otherwise, it is set to false.";
    }
} //notification

notification pcep-session-peer-overload {
    description
        "This notification is sent when a peer enters overload
        state.";

    uses notification-instance-hdr;

    uses notification-session-hdr;

    leaf peer-overloaded {
        type boolean;
        description
            "If the peer has informed the local PCEP entity that
            it is currently overloaded, then this is set to true.
            Otherwise, it is set to false.";
    }

    leaf peer-overload-time {
        type uint32;
        units "seconds";
        must "(../peer-overloaded = true or " +
```

```
        "(../peer-overloaded != true and " +
        "peer-overload-time = 0))" {
            error-message
                "Invalid peer-overload-time";
            description
                "This field is only used if
                peer-overloaded is set to true.
                Otherwise, it is not used and MUST
                be set to zero.";
        }
    }
    description
        "The interval of time that is remaining until the
        peer will cease to be overloaded. If it is not known
        how long the peer will stay in overloaded state, this
        leaf is set to zero.";
}
} //notification

notification pcep-session-peer-overload-clear {
    description
        "This notification is sent when a peer leaves overload
        state.";

    uses notification-instance-hdr;

    leaf peer-overloaded {
        type boolean;
        description
            "If the peer has informed the local PCEP entity that
            it is currently overloaded, then this is set to true.
            Otherwise, it is set to false.";
    }
} //notification
} //module

<CODE ENDS>
```

## 9. Security Considerations

The YANG module defined in this memo 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 NETCONF access control model [RFC6536] provides the means 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 data nodes defined in the YANG module which 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.

TBD: List specific Subtrees and data nodes and their sensitivity/vulnerability.

## 10. Manageability Considerations

### 10.1. Control of Function and Policy

### 10.2. Information and Data Models

### 10.3. Liveness Detection and Monitoring

### 10.4. Verify Correct Operations

### 10.5. Requirements On Other Protocols

### 10.6. Impact On Network Operations

## 11. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

URI: urn:ietf:params:xml:ns:yang:ietf-pcep

Registrant Contact: The PCE WG of the IETF.

XML: N/A; the requested URI is an XML namespace.

This document registers a YANG module in the "YANG Module Names" registry [RFC6020].

Name:	ietf-pcep
Namespace:	urn:ietf:params:xml:ns:yang:ietf-pcep
Prefix:	pcep
Reference:	This I-D

## 12. Acknowledgements

The initial document is based on the PCEP MIB [RFC7420]. Further this document structure is based on Routing Yang Module [I-D.ietf-netmod-routing-cfg]. We would like to thank the authors of aforementioned documents.

## 13. References

### 13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<http://www.rfc-editor.org/info/rfc5440>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [I-D.ietf-pce-stateful-pce] Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-14 (work in progress), March 2016.
- [I-D.ietf-pce-pce-initiated-lsp] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model", draft-ietf-pce-pce-initiated-lsp-05 (work in progress), October 2015.

[I-D.ietf-pce-lsp-setup-type]

Sivabalan, S., Medved, J., Minei, I., Crabbe, E., Varga, R., Tantsura, J., and J. Hardwick, "Conveying path setup type in PCEP messages", draft-ietf-pce-lsp-setup-type-03 (work in progress), June 2015.

[I-D.ietf-pce-segment-routing]

Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Lopez, V., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-07 (work in progress), March 2016.

### 13.2. Informative References

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<http://www.rfc-editor.org/info/rfc4655>>.

[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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6536>>.

[RFC7420] Koushik, A., Stephan, E., Zhao, Q., King, D., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Management Information Base (MIB) Module", RFC 7420, DOI 10.17487/RFC7420, December 2014, <<http://www.rfc-editor.org/info/rfc7420>>.

[I-D.ietf-netmod-routing-cfg]

Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.

[I-D.ietf-netmod-rfc6087bis]

Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-06 (work in progress), March 2016.

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

Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-03 (work in progress), March 2016.

## Appendix A. Contributor Addresses

Rohit Pobbathi  
Huawei Technologies  
Divyashree Techno Park, Whitefield  
Bangalore, Karnataka 560066  
India

EMail: rohit.pobbathi@huawei.com

Vinod KumarS  
Huawei Technologies  
Divyashree Techno Park, Whitefield  
Bangalore, Karnataka 560066  
India

EMail: vinods.kumar@huawei.com

Zafar Ali  
Cisco Systems  
Canada

EMail: zali@cisco.com

Xufeng Liu  
Ericsson  
1595 Spring Hill Road, Suite 500  
Vienna, VA 22182  
USA

EMail: xufeng.liu@ericsson.com

Young Lee  
Huawei Technologies  
5340 Legacy Drive, Building 3  
Plano, TX 75023, USA

Phone: (469) 277-5838  
EMail: leeyoung@huawei.com

Udayasree Palle  
Huawei Technologies  
Divyashree Techno Park, Whitefield  
Bangalore, Karnataka 560066  
India

EMail: udayasree.palle@huawei.com



Xian Zhang  
Huawei Technologies  
Bantian, Longgang District  
Shenzhen 518129  
P.R.China

EMail: zhang.xian@huawei.com

Avantika  
Huawei Technologies  
Divyashree Techno Park, Whitefield  
Bangalore, Karnataka 560066  
India

EMail: avantika.sushilkumar@huawei.com

#### Authors' Addresses

Dhruv Dhody (editor)  
Huawei Technologies  
Divyashree Techno Park, Whitefield  
Bangalore, Karnataka 560066  
India

EMail: dhruv.ietf@gmail.com

Jonathan Hardwick  
Metaswitch  
100 Church Street  
Enfield EN2 6BQ  
UK

EMail: jonathan.hardwick@metaswitch.com

Vishnu Pavan Beeram  
Juniper Networks  
USA

EMail: vbeeram@juniper.net

Jeff Tantsura  
USA

EMail: jefftant@gmail.com

MPLS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 9, 2017

K. Raza  
R. Asati  
Cisco Systems, Inc.

X. Liu  
Ericsson

S. Esale  
Juniper Networks

X. Chen  
Huawei Technologies

H. Shah  
Ciena Corporation

July 8, 2016

YANG Data Model for MPLS LDP and mLDP  
draft-raza-mpls-ldp-mldp-yang-04

Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP) and Multipoint LDP (mLDP).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

## Copyright Notice

Copyright (c) 2016 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 (<http://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. Specification of Requirements . . . . .	3
3. LDP YANG Model . . . . .	3
3.1. Overview . . . . .	4
3.2. Configuration . . . . .	7
3.2.1. Configuration Hierarchy . . . . .	11
3.2.2. All-VRFs Configuration . . . . .	14
3.3. Operational State . . . . .	14
3.3.1. Derived States . . . . .	21
3.4. Notifications . . . . .	26
3.5. Actions . . . . .	26
4. mLDP YANG Model . . . . .	27
4.1. Overview . . . . .	27
4.2. Configuration . . . . .	28
4.2.1. Configuration Hierarchy . . . . .	28
4.2.2. mldp container . . . . .	30
4.2.3. Leveraging LDP containers . . . . .	31
4.2.4. YANG tree . . . . .	31
4.3. Operational State . . . . .	33
4.3.1. Derived states . . . . .	38
4.4. Notifications . . . . .	42
4.5. Actions . . . . .	43
5. Open Items . . . . .	43
6. YANG Specification . . . . .	43
7. Security Considerations . . . . .	110
8. IANA Considerations . . . . .	110
9. Acknowledgments . . . . .	110
10. References . . . . .	110
10.1. Normative References . . . . .	110
10.2. Informative References . . . . .	113
Appendix A. Additional Contributors . . . . .	113

Authors' Addresses	113
--------------------	-----

## 1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] 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] and Multipoint LDP (mLDP) [RFC6388]. For LDP, it also covers LDP IPv6 [RFC7552] and LDP capabilities [RFC5561].

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

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

This document is organized to define the data model for each of the above constructs (configuration, state, action, and notifications) in the sequence as listed earlier. Given that mLDP is tightly coupled with LDP, mLDP data model is defined under LDP tree and in the same sequence as listed above.

## 2. Specification of Requirements

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

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

## 3. LDP YANG Model

### 3.1. Overview

This document defines a new module named "ietf-mpls-ldp" for LDP/mLDP data model where this module augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg].

There are four main containers in "ietf-mpls-ldp" module as follows:

- o Read-Write parameters for configuration (Discussed in Section 3.2)
- o Read-only parameters for operational state (Discussed in Section 3.3)
- o Notifications for events (Discussed in Section 3.4)
- o RPCs for executing commands to perform some action (Discussed in Section 3.5)

For the configuration and state data, this model follows the similar approach described in [I-D.openconfig-netmod-opstate] to represent the configuration (intended state) and operational (applied and derived) state. This means that for every configuration (rw) item, there is an associated (ro) item under "state" container to represent the applied state. Furthermore, protocol derived state is also kept under "state" tree corresponding to the protocol area (discovery, peer etc.). [Ed note: This document will be (re-)aligned with [I-D.openconfig-netmod-opstate] once that specification is adopted as a WG document]

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

```
module: ietf-mpls-ldp
  +-- rw routing
    +-- rw control-plane-protocols
      +-- rw mpls-ldp
        +-- rw global
          +-- rw config
            +-- rw ...
          +-- ro state
            +-- ro ...
          .
        +-- rw ...
      +-- rw ...
    +-- rw ...
  ...

rpcs:
  +-- x mpls-ldp-rpc
  +-- x . . . . .

notifications:
  +--- n mpls-ldp-notif
  +--- n ...
```

Figure 1

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

- o This module aims to address only the core LDP/mLDP parameters as per RFC specification, as well as some widely used and deployed non-RFC features (such as label policies, session authentication etc). Any vendor specific feature should be defined in a vendor-specific augmentation of this model.
- o Multi-topology LDP [RFC7307] and Multi-topology mLDP [I-D.iwijnand-mpls-mldp-multi-topology] are beyond the scope of this document.
- o This module does not cover any applications running on top of LDP and mLDP, nor does it cover any OAM procedures for LDP and mLDP.
- o This model is a VPN Forwarding and Routing (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.

- o A "network-instance" as defined in [I-D.rtgyangdt-rtgwg-ni-model] refers to a VRF instance (both default and non-default) within the scope of this model.
- o This model supports two address-families, namely "ipv4" and "ipv6".
- o 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].
- o The label and peer policies (including filters) are defined using a prefix-list. When used for a peer policy, the prefix refers to the LSR Id of the peer. The prefix-list is referenced from routing-policy model as defined in [I-D.ietf-rtgwg-policy-model].
- o The use of grouping (templates) for bundling and grouping the configuration items is not employed in current revision, and is a subject for consideration in future.
- o 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 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 -- i.e. keeping peer bindings without established or recovered peering -- a "stale" peer. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A graphical representation of LDP YANG data model is presented in Figure 3, Figure 5, Figure 11, and Figure 12. Whereas, the actual model definition in YANG is captured in Section 6.

While presenting the YANG tree view and actual .yang specification, this document assumes the reader is familiar with the concepts of YANG modeling, its presentation and its compilation.

### 3.2. 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 specification supports VRF-centric configuration. For implementations that support protocol-centric configuration, with provision for inheritance and items that apply to all vrfs, we recommend an augmentation of this model such that any protocol-centric or all-vrf configuration is defined under their designated containers within the standard network-instance (please see Section 3.2.2)

This model augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg]. For LDP interfaces, this model refers the MPLS interface as defined under MPLS base specification [I-D.saad-mpls-base-yang]. Furthermore, as mentioned earlier, the configuration tree presents read-write intended configuration leave/items as well as read-only state of the applied configuration. The former is listed under "config" container and latter under "state" container.

Following is high-level configuration organization for LDP/mLDP:



```

module: ietf-mpls-ldp
  +-- routing
    +-- control-plane-protocols
      +-- mpls-ldp
        +-- global
          +-- ...
          +-- ...
          +-- address-family* [afi]
            +-- . . .
            +-- . . .
          +-- discovery
            +-- . . .
        +-- peers
          +-- ...
          +-- ...

```

Figure 2

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 parent. 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.

Following is a simplified graphical representation of the data model for LDP configuration

```

+--rw mpls-ldp!
  +--rw global
    |
    | +--rw config
    | |
    | | +--rw capability
    | | |
    | | | +--rw end-of-lib {capability-end-of-lib}?
    | | | | +--rw enable?    boolean
    | | | +--rw typed-wildcard-fec {capability-typed-wildcard-fec}?
    | | | | +--rw enable?    boolean
    | | | +--rw upstream-label-assignment {capability-upstream-label-assign
ment}?
    | | | | +--rw enable?    boolean
    | | | +--rw graceful-restart
    | | | | +--rw enable?          boolean
    | | | | +--rw helper-enable?   boolean {graceful-restart-helper-mod
e}?
    | | | +--rw reconnect-time?    uint16
    | | | +--rw recovery-time?     uint16
    | | | +--rw forwarding-holdtime? uint16
    | | +--rw igp-synchronization-delay? uint16
    | | +--rw lsr-id?              yang:dotted-quad

```

```

|   +--rw address-family* [afi]
|   |   +--rw afi          ldp-address-family
|   |   +--rw config
|   |   |   +--rw enable?          boolean
|   |   |   +--rw label-policy
|   |   |   |   +--rw independent-mode
|   |   |   |   |   +--rw assign {policy-label-assignment-config}?
|   |   |   |   |   |   +--rw (prefix-option)?
|   |   |   |   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   |   |   |   |   |   +--rw host-routes-only?      boolean
|   |   |   |   +--rw advertise
|   |   |   |   |   +--rw explicit-null
|   |   |   |   |   |   +--rw enable?          boolean
|   |   |   |   |   |   +--rw prefix-list?      prefix-list-ref
|   |   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   |   +--rw accept
|   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   +--rw ordered-mode {policy-ordered-label-config}?
|   |   |   +--rw egress-lsr
|   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   |   +--rw advertise
|   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   |   +--rw accept
|   |   |   |   +--rw prefix-list?          prefix-list-ref
|   |   +--rw ipv4
|   |   |   +--rw transport-address?      inet:ipv4-address
|   |   +--rw ipv6
|   |   |   +--rw transport-address?      inet:ipv6-address
|   +--rw discovery
|   |   +--rw interfaces
|   |   |   +--rw config
|   |   |   |   +--rw hello-holdtime?      uint16
|   |   |   |   +--rw hello-interval?      uint16
|   |   |   +--rw interface* [interface]
|   |   |   |   +--rw interface          mpls-interface-ref
|   |   |   |   +--rw config
|   |   |   |   |   +--rw hello-holdtime?          uint16
|   |   |   |   |   +--rw hello-interval?          uint16
|   |   |   |   |   +--rw igp-synchronization-delay? uint16 {per-interface-ti
mer-config}?
|   |   |   +--rw address-family* [afi]
|   |   |   |   +--rw afi          ldp-address-family
|   |   |   |   +--rw config
|   |   |   |   |   +--rw enable?      boolean
|   |   |   |   |   +--rw ipv4
|   |   |   |   |   |   +--rw transport-address?      union
|   |   |   |   |   +--rw ipv6
|   |   |   |   |   |   +--rw transport-address?      union
|   |   +--rw targeted

```

```

+--rw config
|   +--rw hello-holdtime?    uint16
|   +--rw hello-interval?    uint16
|   +--rw hello-accept {policy-extended-discovery-config}?
|       +--rw enable?        boolean
|       +--rw neighbor-list?  neighbor-list-ref
+--rw address-family* [afi]
|   +--rw afi                ldp-address-family
|   +--rw ipv4
|       +--rw target* [adjacent-address]
|           +--rw adjacent-address    inet:ipv4-address
|           +--rw config
|               +--rw enable?        boolean
|               +--rw local-address?  inet:ipv4-address
|   +--rw ipv6
|       +--rw target* [adjacent-address]
|           +--rw adjacent-address    inet:ipv6-address
|           +--rw config
|               +--rw enable?        boolean
|               +--rw local-address?  inet:ipv6-address
+--rw forwarding-nexthop {forwarding-nexthop-config}?
|   +--rw interfaces
|       +--rw interface* [interface]
|           +--rw interface          mpls-interface-ref
|           +--rw address-family* [afi]
|               +--rw afi            ldp-address-family
|               +--rw config
|                   +--rw ldp-disable?  boolean
+--rw label-policy
|   +--rw independent-mode
|       +--rw assign {policy-label-assignment-config}?
|           +--rw (prefix-option)?
|               +--rw prefix-list?    prefix-list-ref
|               +--rw host-routes-only?  boolean
|       +--rw advertise
|           +--rw explicit-null
|               +--rw enable?        boolean
|               +--rw prefix-list?    prefix-list-ref
|           +--rw prefix-list?        prefix-list-ref
|       +--rw accept
|           +--rw prefix-list?        prefix-list-ref
+--rw ordered-mode {policy-ordered-label-config}?
|   +--rw egress-lsr
|       +--rw prefix-list?    prefix-list-ref
|   +--rw advertise
|       +--rw prefix-list?    prefix-list-ref
|   +--rw accept
|       +--rw prefix-list?    prefix-list-ref

```

```

+--rw peers
  +--rw config
    |   +--rw session-authentication-md5-password?  string
    |   +--rw session-ka-holdtime?                  uint16
    |   +--rw session-ka-interval?                   uint16
    |   +--rw session-downstream-on-demand {session-downstream-on-demand-con
fig}?
    |   |   +--rw enable?          boolean
    |   |   +--rw peer-list?       peer-list-ref
    +--rw peer* [lsr-id]
      +--rw lsr-id      yang:dotted-quad
      +--rw config
        +--rw admin-down?          boolean
        +--rw capability
        +--rw label-policy
          |   +--rw advertise
          |   |   +--rw prefix-list?  prefix-list-ref
          |   +--rw accept
          |   |   +--rw prefix-list?  prefix-list-ref
        +--rw session-authentication-md5-password?  string
        +--rw graceful-restart
          |   +--rw enable?          boolean
          |   +--rw reconnect-time?  uint16
          |   +--rw recovery-time?   uint16
        +--rw session-ka-holdtime?          uint16
        +--rw session-ka-interval?          uint16
        +--rw address-family
          +--rw ipv4
            |   +--rw label-policy
            |   |   +--rw advertise
            |   |   |   +--rw prefix-list?  prefix-list-ref
            |   |   +--rw accept
            |   |   |   +--rw prefix-list?  prefix-list-ref
          +--rw ipv6
            +--rw label-policy
            |   +--rw advertise
            |   |   +--rw prefix-list?  prefix-list-ref
            +--rw accept
            |   +--rw prefix-list?  prefix-list-ref

```

Figure 3

### 3.2.1. Configuration Hierarchy

The LDP configuration container is logically divided into following high-level config areas:

```
Per-VRF parameters
  o Global parameters
  o Per-address-family parameters
  o LDP Capabilities parameters
  o Hello Discovery parameters
    - interfaces
      - Per-interface:
        Global
        Per-address-family
    - targeted
      - Per-target
  o Peer parameters
    - Global
    - Per-peer
      Per-address-family
      Capabilities parameters
  o Forwarding parameters
```

Figure 4

Following subsections briefly explain these configuration areas.

#### 3.2.1.1. Per-VRF parameters

LDP module resides under an 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.

##### 3.2.1.1.1. Per-VRF 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 as an LSR Id.

##### 3.2.1.1.2. Per-VRF Capabilities parameters

This container falls under 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.

#### 3.2.1.1.3. Per-VRF 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.

#### 3.2.1.1.4. Per-VRF 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 former is interface hello timers, and example of 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 mean to configure multiple target addresses to perform extended discovery to a specific destination target, as well as to fine-tune the per-target parameters.

#### 3.2.1.1.5. Per-VRF 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 using its LSR Id and hence LSR Id is the key for peer list

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 former is per-peer session password configuration, whereas the example of latter is prefix-list based label policies (inbound and outbound) that apply to a given peer.

### 3.2.1.1.6. Per-VRF 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.

### 3.2.2. All-VRFs Configuration

[Ed note: TODO]

### 3.3. 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/) as the configuration.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the protocol. [Ed note: This is where this model differs presently from [I-D.openconfig-netmod-opstate] and subject to alignment in later revisions]

Following is a simplified graphical representation of the data model for LDP operational state.

```

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
      +--ro state
        +--ro capability
          +--ro end-of-lib {capability-end-of-lib}?
            | +--ro enable?    boolean
          +--ro typed-wildcard-fec {capability-typed-wildcard-fec}?
            | +--ro enable?    boolean
          +--ro upstream-label-assignment {capability-upstream-label-assignment}?
            | +--ro enable?    boolean
          +--ro graceful-restart
            +--ro enable?      boolean
            +--ro helper-enable? boolean {graceful-restart-helper-mod
e}?
          +--ro reconnect-time?    uint16
          +--ro recovery-time?     uint16
          +--ro forwarding-holdtime? uint16

```

```

+--ro igmp-synchronization-delay?    uint16
+--ro lsr-id?                        yang:dotted-quad
+--rw address-family* [afi]
+--rw afi                            ldp-address-family
+--ro state
+--ro enable?                        boolean
+--ro label-policy
+--ro independent-mode
+--ro assign {policy-label-assignment-config}?
+--ro (prefix-option)?
+--:(prefix-list)
+--ro prefix-list?                    prefix-list-ref
+--:(host-routes-only)
+--ro host-routes-only?              boolean
+--ro advertise
+--ro explicit-null
+--ro enable?                        boolean
+--ro prefix-list?                    prefix-list-ref
+--ro prefix-list?                    prefix-list-ref
+--ro accept
+--ro prefix-list?                    prefix-list-ref
+--ro ordered-mode {policy-ordered-label-config}?
+--ro egress-lsr
+--ro prefix-list?                    prefix-list-ref
+--ro advertise
+--ro prefix-list?                    prefix-list-ref
+--ro accept
+--ro prefix-list?                    prefix-list-ref
+--ro ipv4
+--ro transport-address?              inet:ipv4-address
+--ro bindings
+--ro address* [address]
+--ro address                        inet:ipv4-address
+--ro advertisement-type?              advertised-received
+--ro peer?                          leafref
+--ro fec-label* [fec]
+--ro fec                            inet:ipv4-prefix
+--ro peer* [peer advertisement-type]
+--ro peer                          leafref
+--ro advertisement-type              advertised-received
+--ro label?                          mpls:mpls-label
+--ro used-in-forwarding?              boolean
+--ro ipv6
+--ro transport-address?              inet:ipv6-address
+--ro binding
+--ro address* [address]
+--ro address                        inet:ipv6-address
+--ro advertisement-type?              advertised-received

```



```

| | | | | +--ro peer? leafref
| | | | | +--ro fec-label* [fec]
| | | | | +--ro fec inet: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
+--rw discovery
+--rw interfaces
+--ro state
| +--ro hello-holdtime? uint16
| +--ro hello-interval? uint16
+--rw interface* [interface]
+--ro state
| +--ro hello-holdtime? uint16
| +--ro hello-interval? uint16
| +--ro igp-synchronization-delay? uint16 {per-interface-ti
mer-config}?
| +--ro next-hello? uint16
+--rw address-family* [afi]
+--rw afi ldp-address-family
+--ro state
+--ro enable? boolean
+--ro ipv4
| +--ro transport-address? union
| +--ro hello-adjacencies* [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? leafref
+--ro ipv6
+--ro transport-address? union
+--ro hello-adjacencies* [adjacent-address]
+--ro adjacent-address inet:ipv6-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?                    leafref
+--rw targeted
+--ro state
|   +--ro hello-holdtime?      uint16
|   +--ro hello-interval?     uint16
|   +--ro hello-accept {policy-extended-discovery-config}?
|       +--ro enable?          boolean
|       +--ro neighbor-list?   neighbor-list-ref
+--rw address-family* [afi]
+--rw afi                  ldp-address-family
+--ro state
|   +--ro ipv4
|       +--ro hello-adjacencies* [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?                  leafref
+--ro ipv6
|   +--ro hello-adjacencies* [local-address adjacent-address
]
|       +--ro local-address        inet:ipv6-address
|       +--ro adjacent-address      inet:ipv6-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?                  leafref
+--rw ipv4
|   +--rw target* [adjacent-address]
|       +--rw adjacent-address      inet:ipv4-address

```

```

|         |         |         +---ro state                boolean
|         |         |         +---ro enable?              boolean
|         |         |         +---ro local-address?       inet:ipv4-address
+---rw ipv6
|         |         |         +---rw target* [adjacent-address]
|         |         |         +---rw adjacent-address      inet:ipv6-address
|         |         |         +---ro state
|         |         |         +---ro enable?              boolean
|         |         |         +---ro local-address?       inet:ipv6-address
+---rw forwarding-nexthop {forwarding-nexthop-config}?
|         |         +---rw interfaces
|         |         |         +---rw interface* [interface]
|         |         |         +---rw interface            mpls-interface-ref
|         |         |         +---rw address-family* [afi]
|         |         |         +---rw afi                  ldp-address-family
|         |         |         +---ro state
|         |         |         +---ro ldp-disable?        boolean
+---rw peers
|         |         +---ro state
|         |         |         +---ro session-authentication-md5-password? string
|         |         |         +---ro session-ka-holdtime?   uint16
|         |         |         +---ro session-ka-interval?   uint16
|         |         |         +---ro session-downstream-on-demand {session-downstream-on-demand-con
fig)?
|         |         |         +---ro enable?              boolean
|         |         |         +---ro peer-list?           peer-list-ref
+---rw peer* [lsr-id]
|         |         |         +---rw lsr-id               yang:dotted-quad
|         |         |         +---ro state
|         |         |         +---ro admin-down?          boolean
|         |         |         +---ro capability
|         |         |         +---ro label-policy
|         |         |         |         +---ro advertise
|         |         |         |         |         +---ro prefix-list?    prefix-list-ref
|         |         |         |         +---ro accept
|         |         |         |         +---ro prefix-list?    prefix-list-ref
+---ro session-authentication-md5-password?    string
+---ro graceful-restart
|         |         |         +---ro enable?              boolean
|         |         |         +---ro reconnect-time?      uint16
|         |         |         +---ro recovery-time?       uint16
+---ro session-ka-holdtime?                    uint16
+---ro session-ka-interval?                    uint16
+---ro address-family
|         |         |         +---ro ipv4
|         |         |         |         +---ro label-policy
|         |         |         |         |         +---ro advertise
|         |         |         |         |         |         +---ro prefix-list?    prefix-list-ref
|         |         |         |         +---ro accept

```

```

|      +--ro prefix-list?    prefix-list-ref
+--ro hello-adjacencies* [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?          mpls-interface-ref
+--ro ipv6
|   +--ro label-policy
|   |   +--ro advertise
|   |   |   +--ro prefix-list?    prefix-list-ref
|   |   +--ro accept
|   |   |   +--ro prefix-list?    prefix-list-ref
|   +--ro hello-adjacencies* [local-address adjacent-address]
|   |   +--ro local-address      inet:ipv6-address
|   |   +--ro adjacent-address    inet:ipv6-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?          mpls-interface-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 peer-ldp-id?            yang:dotted-quad
+--ro received-peer-state
|   +--ro graceful-restart
|   |   +--ro enable?          boolean
|   |   +--ro reconnect-time?  uint16
|   |   +--ro recovery-time?   uint16
+--ro capability
+--ro end-of-lib

```

```

|         |  +--ro enable?    boolean
|         +--ro typed-wildcard-fec
|         |  +--ro enable?    boolean
|         +--ro upstream-label-assignment
|         +--ro enable?    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?            string
+--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

```

Figure 5

### 3.3.1. Derived States

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

- Neighbor Adjacencies

- Peer

- Bindings (FEC-label and address)

- Capabilities

#### 3.3.1.1. 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). This 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.

```

+--rw mpls-ldp!
+--rw discovery
+--rw interfaces
|   +--rw interface* [interface]
|       +--rw address-family* [af]
|           +--ro state
|               +--ro ipv4 (or ipv6)
|                   +--ro hello-adjacencies* [adjacent-address]
|                       +--ro adjacent-address
|                           . . . .
|                           . . . .
+--rw targeted
+--rw address-family* [afi]
+--rw afi            address-family
+--ro state
+--ro ipv4 (or ipv6)
+--ro hello-adjacencies* [local-address adjacent-addresses]
    +--ro local-address
    +--ro adjacent-address
        . . . .
        . . . .

```

Figure 6

### 3.3.1.2. Peer state

Peer related derived 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.

```

+--rw mpls-ldp!
  +--rw peers
    +--rw peer* [lsr-id]
      +--rw lsr-id
      +--ro state
        +--ro session-ka-holdtime?
        +-- . . . .
        +-- . . . .
        +--ro capability
        + +ro -- . . .
        +--ro address-family
          +--ro ipv4 (or ipv6)
            +--ro hello-adjacencies* [local-address adjacent-address]
            |
            | . . . .
            | . . . .
        +--ro received-peer-state
          +--ro . . . .
          +--ro capability
          |
          +--ro . . . .
        +--ro statistics
          +-- . . . .
          +-- . . . .

```

Figure 7

### 3.3.1.3. 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 200.1.1.1/32:
    advertised: local-label 16000
    peer 192.168.0.2:0
    peer 192.168.0.3:0
    peer 192.168.0.4:0
    received:
      peer 192.168.0.2:0, label 16002, used-in-forwarding=Yes
      peer 192.168.0.3:0, label 17002, used-in-forwarding=No
  FEC 200.1.1.2/32:
    . . . .
  FEC 201.1.0.0/16:
    . . . .

Address bindings:
  Addr 1.1.1.1:
    advertised
  Addr 1.1.1.2:
    advertised
  Addr 2.2.2.2:
    received, peer 192.168.0.2
  Addr 2.2.2.22:
    received, peer 192.168.0.2
  Addr 3.3.3.3:
    received, peer 192.168.0.3
  Addr 3.3.3.33:
    received, peer 192.168.0.3
```

Figure 8

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.

```

+--rw mpls-ldp!
+--rw global
+--rw address-family* [afi]
+--rw afi          address-family
+--ro state
+--ro ipv4 (or ipv6)
+--ro bindings
+--ro address* [address]
|   +--ro address
|   +--ro direction?   advertised-received
|   +--ro peer?        leafref
+--ro fec-label* [fec]
+--ro fec          inet:ipv4-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 9

#### 3.3.1.4. 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 global
|   +--ro state
|   |   +--ro capability
|   |   |   +--ro . . . .
|   |   |   +--ro . . . .
+--rw peers
+--rw peer* [lsr-id]
+--rw lsr-id   yang:dotted-quad
+--ro state
+--ro received-peer-state
+--ro capability
+--ro . . . .
+--ro . . . .

```

Figure 10

### 3.4. 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 with outgoing label.

Following is a simplified graphical representation of the data model for LDP notifications.

```

module: ietf-mpls-ldp
notifications:
  +---n mpls-ldp-peer-event
  |   +--ro event-type?   oper-status-event-type
  |   +--ro peer-ref?    leafref
  +---n mpls-ldp-hello-adjacency-event
  |   +--ro event-type?   oper-status-event-type
  |   +--ro (hello-adjacency-type)?
  |   |   +--:(targeted)
  |   |   |   +--ro targeted
  |   |   |   +--ro target-address?   inet:ip-address
  |   |   +--:(link)
  |   |   |   +--ro link
  |   |   |   +--ro next-hop-interface?   mpls-interface-ref
  |   |   |   +--ro next-hop-address?    inet:ip-address
  +---n mpls-ldp-fec-event
  |   +--ro event-type?   oper-status-event-type
  |   +--ro prefix?       inet:ip-prefix

```

Figure 11

### 3.5. Actions

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.

Following is a simplified graphical representation of the data model for LDP actions.

```

module: ietf-mpls-ldp
rpcs:
  +---x mpls-ldp-clear-peer
  |   +---w input
  |   +---w lsr-id?    union
  +---x mpls-ldp-clear-hello-adjacency
  |   +---w input
  |   +---w hello-adjacency
  |   +---w (hello-adjacency-type)?
  |   |   +---:(targeted)
  |   |   |   +---w targeted!
  |   |   |   +---w target-address?    inet:ip-address
  |   |   +---:(link)
  |   |   +---w link!
  |   |   +---w next-hop-interface?    mpls-interface-ref
  |   |   +---w next-hop-address?     inet:ip-address
  +---x mpls-ldp-clear-peer-statistics
  |   +---w input
  |   +---w lsr-id?    union

```

Figure 12

#### 4. mLDP YANG Model

##### 4.1. Overview

Due to tight dependency of mLDP on LDP, mLDP model builds on top of LDP model defined earlier in the document. Following are the main mLDP areas and documents that are within the scope of this model:

- o mLDP Base Specification [RFC6388]
- o mLDP Recursive FEC [RFC6512]
- o Targeted mLDP [RFC7060]
- o mLDP Fast-Reroute (FRR)
  - \* Node Protection [RFC7715]
  - \* Multicast-only
- o Hub-and-Spoke Multipoint LSPs [RFC7140]
- o mLDP In-band Signaling [RFC6826] (future revision)
- o mLDP In-band signaling in a VRF [RFC7246]

- o mLDP In-band Signaling with Wildcards [RFC7438] (future revision)
- o Configured Leaf LSPs (manually provisioned)

[Ed Note: Some of the topics in the above list are to be addressed/added in later revision of this document].

## 4.2. Configuration

### 4.2.1. Configuration Hierarchy

In terms of overall configuration layout, following figure highlights extensions to LDP configuration model to incorporate mLDP:

```

+-- mpls-ldp
+-- ...
+-- ...
+-- mldp
+-- ...
+-- ...
+-- address-family* [af]
+-- af
+-- ...
+-- ...
+-- global
+-- ...
+-- capability
+-- ...
+-- ...
+-- mldp
+-- ...
+-- ...
+-- discovery
+-- ...
+-- ...
+-- forwarding-nexthop
+-- interfaces
+-- interface* [interface]
+-- interface
+-- address-family* [af]
+-- af
+-- ...
+-- mldp-disable
+-- peers
+-- ...
+-- ...
+-- peer* [lsr-id]
+-- ...
+-- ...
+-- capability
+-- ...
+-- ...
+-- mldp
+-- ...
+-- ...

```

Figure 13

From above hierarchy, we can categorize mLDP configuration parameters into two types:

- o Parameters that leverage/extend LDP containers and parameters
- o Parameters that are mLDP specific

Following subsections first describe mLDP specific configuration parameters, followed by those leveraging LDP.

#### 4.2.2. mldp container

mldp container resides directly under "mpls-ldp" and holds the configuration related to items that are mLDP specific. The main items under this container are:

- o mLDP enabling: To enable mLDP under a (VRF) routing instance, mldp container is enabled under LDP. Given that mLDP requires LDP signalling, it is not sensible to allow disabling LDP control plane under a (VRF) network-instance while requiring mLDP to be enabled for the same. However, if a user wishes only to allow signalling for multipoint FECs on an LDP/mLDP enabled VRF instance, he/she can use LDP label-policies to disable unicast FECs under the VRF.
- o mLDP per-AF features: mLDP manages its own list of IP address-families and the features enabled underneath. The per-AF mLDP configuration items include:
  - \* Multicast-only FRR: This enables Multicast-only FRR functionality for a given AF under mLDP. The feature allows route-policy to be configured for finer control/applicability of the feature.
  - \* Recursive FEC: The recursive-fec feature [RFC6512] can be enabled per AF with a route-policy.
  - \* Configured Leaf LSPs: To provision multipoint leaf LSP manually, a container is provided per-AF under LDP. The configuration is flexible and allows a user to specify MP LSPs of type p2mp or mp2mp with IPv4 or IPv6 root address(es) by using either LSP-Id or (S,G).

Targeted mLDP feature specification [RFC7060] do not require any mLDP specific configuration. It, however, requires LDP upstream-label-assignment capability [RFC6389] to be enabled.

#### 4.2.3. Leveraging LDP containers

mLDP configuration model leverages following configuration areas and containers that are already defined for LDP:

- o Capabilities: A new container "mldp" is defined under Capabilities container. This new container specifies any mLDP specific capabilities and their parameters. Moreover, a new "mldp" container is also added under per-peer capability container to override/control mLDP specific capabilities on a peer level. In the scope of this document, the most important capabilities related to mLDP are p2mp, mp2mp, make-before-break, hub-and-spoke, and node-protection.
- o Discovery and Peer: mLDP requires LDP discovery and peer procedures to form mLDP peering. A peer is treated as mLDP peer only when either P2MP or MP2MP capabilities have been successfully exchanged with the peer. If a user wish to selectively enable or disable mLDP with a LDP-enabled peer, he/she may use per-peer mLDP capabilities configuration. [Ed Note: The option to control mLDP enabling/disabling on a peer-list is being explored for future ]. In most common deployments, it is desirable to disable mLDP (capabilities announcements) on a targeted-only LDP peering, where targeted-only peer is the one whose discovery sources are targeted only. In future revision, a configuration option for this support will also be provided.
- o Forwarding: By default, mLDP is allowed to select any of the LDP enabled interface as a downstream interface towards a nexthop (LDP/mLDP peer) for MP LSP programming. However, a configuration option is provided to allow mLDP to exclude a given interface from such a selection. Note that such a configuration option will be useful only when there are more than one interfaces available for the downstream selection.

This goes without saying that mLDP configuration tree follows the same approach as LDP, where the tree comprise leafs for intended configuration.

#### 4.2.4. YANG tree

The following figure captures the YANG tree for mLDP configuration. To keep the focus, the figure has been simplified to display only mLDP items without any LDP items.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
```



```

+--rw global
|
|   +--rw config
|   |
|   |   +--rw capability
|   |   |
|   |   |   +--rw mldp {mldp}?
|   |   |   |
|   |   |   |   +--rw p2mp
|   |   |   |   |
|   |   |   |   |   +--rw enable?    boolean
|   |   |   |   |
|   |   |   |   +--rw mp2mp
|   |   |   |   |
|   |   |   |   |   +--rw enable?    boolean
|   |   |   |   |
|   |   |   |   +--rw make-before-break
|   |   |   |   |
|   |   |   |   |   +--rw enable?    boolean
|   |   |   |   |   +--rw switchover-delay? uint16
|   |   |   |   |   +--rw timeout?    uint16
|   |   |   +--rw hub-and-spoke {capability-mldp-hsmp}?
|   |   |   |
|   |   |   |   +--rw enable?    boolean
|   |   |   +--rw node-protection {capability-mldp-node-protection}?
|   |   |   |
|   |   |   |   +--rw plr?        boolean
|   |   |   |   +--rw merge-point
|   |   |   |   |
|   |   |   |   |   +--rw enable?    boolean
|   |   |   |   |   +--rw targeted-session-teardown-delay? uint16
|   |   +--rw mldp {mldp}?
|   |   |
|   |   |   +--rw config
|   |   |   |
|   |   |   |   +--rw enable?    boolean
|   |   |   +--rw address-family* [afi]
|   |   |   |
|   |   |   |   +--rw afi
|   |   |   |   |
|   |   |   |   |   ldp-address-family
|   |   |   |   +--rw config
|   |   |   |   |
|   |   |   |   |   +--rw multicast-only-frr {mldp-mofrr}?
|   |   |   |   |   |
|   |   |   |   |   |   +--rw prefix-list?    prefix-list-ref
|   |   |   |   |   |
|   |   |   |   |   +--rw recursive-fec
|   |   |   |   |   |
|   |   |   |   |   |   +--rw prefix-list?    prefix-list-ref
|   |   |   +--rw configured-leaf-lsps
|   |   |   |
|   |   |   |   +--rw p2mp
|   |   |   |   |
|   |   |   |   |   +--rw roots-ipv4
|   |   |   |   |   |
|   |   |   |   |   |   +--rw root* [root-address]
|   |   |   |   |   |   |
|   |   |   |   |   |   |   +--rw root-address    inet:ipv4-address
|   |   |   |   |   |   |   +--rw lsp* [lsp-id source-address group-address]
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   +--rw lsp-id        uint16
|   |   |   |   |   |   |   |   +--rw source-address  inet:ipv4-address
|   |   |   |   |   |   |   |   +--rw group-address   inet:ipv4-address-no-zone
|   |   |   |   |   +--rw roots-ipv6
|   |   |   |   |   |
|   |   |   |   |   |   +--rw root* [root-address]
|   |   |   |   |   |   |
|   |   |   |   |   |   |   +--rw root-address    inet:ipv6-address
|   |   |   |   |   |   |   +--rw lsp* [lsp-id source-address group-address]
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   +--rw lsp-id        uint16
|   |   |   |   |   |   |   |   +--rw source-address  inet:ipv6-address
|   |   |   |   |   |   |   |   +--rw group-address   inet:ipv6-address-no-zone
|   |   |   +--rw mp2mp
|   |   |   |
|   |   |   |   +--rw roots-ipv4
|   |   |   |   |
|   |   |   |   |   +--rw root* [root-address]

```

```

|--rw root-address          inet:ipv4-address
+--rw lsp* [lsp-id source-address group-address]
|   |--rw lsp-id            uint16
|   |--rw source-address     inet:ipv4-address
|   |--rw group-address      inet:ipv4-address-no-zone
+--rw roots-ipv6
|   |--rw root* [root-address]
|   |--rw root-address       inet:ipv6-address
|   |--rw lsp* [lsp-id source-address group-address]
|   |   |--rw lsp-id        uint16
|   |   |--rw source-address inet:ipv6-address
|   |   |--rw group-address  inet:ipv6-address-no-zone
+--rw forwarding-nexthop {forwarding-nexthop-config}?
|   +--rw interfaces
|   |   +--rw interface* [interface]
|   |   |   |--rw interface      if:interface-ref
|   |   |   |--rw address-family* [afi]
|   |   |   |   |--rw afi        address-family
|   |   |   |   |--rw config
|   |   |   |   |   |--rw mldp-disable?    boolean {mldp}?
+--rw peers
|   +--rw peer* [lsr-id]
|   |   +--rw capability
|   |   |   +--rw mldp {mldp}?
|   |   |   |   +--rw p2mp
|   |   |   |   |   +--rw enable?    boolean
|   |   |   |   +--rw mp2mp
|   |   |   |   |   +--rw enable?    boolean
|   |   |   +--rw make-before-break
|   |   |   |   +--rw enable?          boolean
|   |   |   |   +--rw switchover-delay? uint16
|   |   |   |   +--rw timeout?         uint16
|   |   +--rw hub-and-spoke {capability-mldp-hsmp}?
|   |   |   +--rw enable?    boolean
|   |   +--rw node-protection {capability-mldp-node-protection}?
|   |   |   +--rw plr?       boolean
|   |   +--rw merge-point
|   |   |   +--rw enable?          boolean
|   |   +--rw targeted-session-teardown-delay?    uint16

```

Figure 14

### 4.3. Operational State

Operational state of mLDP can be queried and obtained from this read-only container "mldp" which resides under mpls-ldp container.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the mLDP protocol.

Following is a simplified graphical representation of the data model for mLDP operational state:

```

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
      +--ro state
        +--ro capability
          +--ro mldp {mldp}?
            +--ro p2mp
              | +--ro enable?    boolean
            +--ro mp2mp
              | +--ro enable?    boolean
            +--ro make-before-break
              | +--ro enable?          boolean
              | +--ro switchover-delay? uint16
              | +--ro timeout?         uint16
            +--ro hub-and-spoke {capability-mldp-hsmp}?
              | +--ro enable?    boolean
            +--ro node-protection {capability-mldp-node-protection}?
              +--ro plr?          boolean
              +--ro merge-point
                +--ro enable?          boolean
                +--ro targeted-session-teardown-delay? uint16
          +--rw mldp {mldp}?
            +--ro state
              +--ro enable?    boolean
            +--rw address-family* [afi]
              +--rw afi
                ldp-address-family
              +--ro state
                +--ro multicast-only-frr {mldp-mofrr}?
                  | +--ro prefix-list?    prefix-list-ref
                +--ro recursive-fec
                  | +--ro prefix-list?    prefix-list-ref
                +--ro ipv4
                  +--ro roots
                    +--ro root* [root-address]
                      +--ro root-address    inet:ipv4-address
                      +--ro is-self?        boolean
                      +--ro reachability* [address interface]
                        +--ro address        inet:ipv4-address
                        +--ro interface      mpls-interface-ref

```

```

| | | | | +--ro peer? leafref
|--ro bindings
+--ro opaque-type-lspid
|   +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
|       +--ro root-address inet:ipv4-address
|       +--ro lsp-id uint32
|       +--ro recur-root-address inet:ip-address
|       +--ro recur-rd route-distinguisher
|       +--ro multipoint-type? multipoint-type
|       +--ro peer* [direction peer advertisement-type]
|           +--ro direction downstream-upstream
|           +--ro peer leafref
|           +--ro advertisement-type advertised-received
|           +--ro label? mpls:mpls-label
|           +--ro mbb-role? enumeration
|           +--ro mofrr-role? enumeration
+--ro opaque-type-src
    +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
        +--ro root-address inet:ipv4-address
        +--ro source-address inet:ip-address
        +--ro group-address inet:ip-address-no-zon
e
| | | | | +--ro rd route-distinguisher
+--ro recur-root-address inet:ip-address
+--ro recur-rd route-distinguisher
+--ro multipoint-type? multipoint-type
+--ro peer* [direction peer advertisement-type]
    +--ro direction downstream-upstream
    +--ro peer leafref
    +--ro advertisement-type advertised-received
    +--ro label? mpls:mpls-label
    +--ro mbb-role? enumeration
    +--ro mofrr-role? enumeration
+--ro opaque-type-bidir
    +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
        +--ro root-address inet:ipv4-address
        +--ro rp inet:ip-address
        +--ro group-address inet:ip-address-no-zon
e
| | | | | +--ro rd route-distinguisher
+--ro recur-root-address inet:ip-address
+--ro recur-rd route-distinguisher
+--ro multipoint-type? multipoint-type
+--ro peer* [direction peer advertisement-type]
    +--ro direction downstream-upstream
    +--ro peer leafref
    +--ro advertisement-type advertised-received
    +--ro label? mpls:mpls-label
    +--ro mbb-role? enumeration
    +--ro mofrr-role? enumeration

```

```

+--ro ipv6
+--ro roots
|   +--ro root* [root-address]
|       +--ro root-address      inet:ipv6-address
|       +--ro is-self?          boolean
|       +--ro reachability* [address interface]
|           +--ro address      inet:ipv6-address
|           +--ro interface    mpls-interface-ref
|           +--ro peer?        leafref
+--ro bindings
|   +--ro opaque-type-lspid
|       +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
|           +--ro root-address      inet:ipv6-address
|           +--ro lsp-id            uint32
|           +--ro recur-root-address inet:ip-address
|           +--ro recur-rd          route-distinguisher
|           +--ro multipoint-type?  multipoint-type
|           +--ro peer* [direction peer advertisement-type]
|               +--ro direction      downstream-upstream
|               +--ro peer            leafref
|               +--ro advertisement-type advertised-received
|               +--ro label?          mpls:mpls-label
|               +--ro mbb-role?       enumeration
|               +--ro mofrr-role?     enumeration
|   +--ro opaque-type-src
|       +--ro fec-label* [root-address source-address group-a
address rd recur-root-address recur-rd]
|           +--ro root-address      inet:ipv6-address
|           +--ro source-address     inet:ip-address
|           +--ro group-address      inet:ip-address-no-zon
e
|           +--ro rd                route-distinguisher
|           +--ro recur-root-address inet:ip-address
|           +--ro recur-rd          route-distinguisher
|           +--ro multipoint-type?  multipoint-type
|           +--ro peer* [direction peer advertisement-type]
|               +--ro direction      downstream-upstream
|               +--ro peer            leafref
|               +--ro advertisement-type advertised-received
|               +--ro label?          mpls:mpls-label
|               +--ro mbb-role?       enumeration
|               +--ro mofrr-role?     enumeration
|   +--ro opaque-type-bidir
|       +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
|           +--ro root-address      inet:ipv6-address
|           +--ro rp                inet:ip-address
|           +--ro group-address      inet:ip-address-no-zon
e
|           +--ro rd                route-distinguisher
|           +--ro recur-root-address inet:ip-address
|           +--ro recur-rd          route-distinguisher

```

```

+---ro multipoint-type?      multipoint-type
+---ro peer* [direction peer advertisement-type]
    +---ro direction          downstream-upstream
    +---ro peer                leafref
    +---ro advertisement-type  advertised-received
    +---ro label?              mpls:mpls-label
    +---ro mbb-role?           enumeration
    +---ro mofrr-role?         enumeration
+---rw forwarding-nexthop {forwarding-nexthop-config}?
+---rw interfaces
    +---rw interface* [interface]
        +---rw address-family* [afi]
            +---ro state
                +---ro mldp-disable?    boolean {mldp}?
+---rw peers
+---rw peer* [lsr-id]
    +---ro state
        +---ro capability
            +---ro mldp {mldp}?
            +---ro p2mp
                | +---ro enable?    boolean
            +---ro mp2mp
                | +---ro enable?    boolean
            +---ro make-before-break
                | +---ro enable?      boolean
                | +---ro switchover-delay?  uint16
                | +---ro timeout?      uint16
            +---ro hub-and-spoke {capability-mldp-hsmp}?
                | +---ro enable?    boolean
            +---ro node-protection {capability-mldp-node-protection}?
                +---ro plr?          boolean
                +---ro merge-point
                    +---ro enable?      boolean
                +---ro targeted-session-teardown-delay?  uint16
+---ro received-peer-state
    +---ro capability
        +---ro mldp {mldp}?
        +---ro p2mp
            | +---ro enable?    boolean
        +---ro mp2mp
            | +---ro enable?    boolean
        +---ro make-before-break
            | +---ro enable?    boolean
        +---ro hub-and-spoke
            | +---ro enable?    boolean
        +---ro node-protection
            +---ro plr?          boolean
            +---ro merge-point?  boolean

```

Figure 15

#### 4.3.1. Derived states

Following are main areas for which mLDP operational derived state is defined:

- o Root
- o Bindings (FEC-label)
- o Capabilities

##### 4.3.1.1. Root state

Root address is a fundamental construct for MP FEC bindings and LSPs. The root state provides information on all the known roots in a given address-family, and their information on the root reachability (as learnt from RIB). In case of multi-path reachability to a root, the selection of upstream path is done on per-LSP basis at the time of LSP setup. Similarly, when protection mechanisms like MBB or MoFRR are in place, the path designation as active/standby or primary/backup is also done on per LSP basis. It is to be noted that a given root can be shared amongst multiple P2MP and/or MP2MP LSPs. Moreover, an LSP can be signaled to more than one root for RNR purposes.

The following diagram illustrates a root database on a branch/transit LSR:

```
root 1.1.1.1:
  path1:
    RIB: GigEthernet 1/0, 12.1.0.2;
    LDP: peer 192.168.0.1:0
  path2:
    RIB: GigEthernet 2/0, 12.2.0.2;
    LDP: peer 192.168.0.3:0

root 2.2.2.2:
  path1:
    RIB: 3.3.3.3;                (NOTE: This is a recursive path)
    LDP: peer 192.168.0.3:0      (NOTE: T-mLDP peer)

root 9.9.9.9:
  . . . .
```

Figure 16

A root entry on a root LSR itself will be presented as follows:

```
root 9.9.9.9:
  is-self
```

Figure 17

#### 4.3.1.2. Bindings state

Binding state provides information on mLDP FEC-label bindings for both P2MP and MP2MP FEC types. Like LDP, the FEC-label binding derived state is presented in a FEC-centric view per address-family, and provides information on both inbound (received) and outbound (advertised) bindings. The FEC is presented as (root-address, opaque-type-data) and the direction (upstream or downstream) is picked with respect to root reachability. In case of MBB or/and MoFRR, the role of a given peer binding is also provided with respect to MBB (active or standby) or/and MoFRR (primary or backup).

This document covers following type of opaque values with their keys in the operational model of mLDP bindings:



Opaque Type	Key	RFC
Generic LSP Identifier	LSP Id	[RFC6388]
Transit IPv4 Source	Source, Group	[RFC6826]
Transit IPv6 Source	Source, Group	[RFC6826]
Transit IPv4 Bidir	RP, Group	[RFC6826]
Transit IPv6 Bidir	RP, Group	[RFC6826]
Transit VPNv4 Source	Source, Group, RD	[RFC7246]
Transit VPNv6 Source	Source, Group, RD	[RFC7246]
Transit VPNv4 Bidir	RP, Group, RD	[RFC7246]
Transit VPNv6 Bidir	RP, Group, RD	[RFC7246]
Recursive Opaque	Root	[RFC6512]
VPN-Recursive Opaque	Root, RD	[RFC6512]

Table 1: MP Opaque Types and keys

It is to be noted that there are three basic types (LSP Id, Source, and Bidir) and then there are variants (VPN, recursive, VPN-recursive) on top of these basic types.

Following captures high level tree hierarchy for mLDP bindings state:

```

+--rw mpls-ldp!
  +--rw mldp
    +--rw address-family* [afi]
      +--rw afi          address-family
      +--ro state
        +--ro ipv4 (or ipv6)
          +--ro bindings
            +--ro opaque-type-xxx [root-address, type-specific-key]
              +--ro root-address
              +--ro ...
              +--ro recur-root-address      inet:ipv4-address
              +--ro recur-rd                route-distinguisher
              +--ro multipoint-type?        multipoint-type
              +--ro peer* [direction peer advertisement-type]
                +--ro direction            downstream-upstream
                +--ro peer                  leafref
                +--ro advertisement-type    advertised-received
                +--ro label?                mpls:mpls-label
                +--ro mbb-role?             enumeration
                +--ro mofrr-role?           enumeration

```

Figure 18

In the above tree, the type-specific-key varies with the base type as listed in earlier Table 1. For example, if the opaque type is Generic LSP Identifier, then the type-specific-key will be a uint32 value corresponding to the LSP. Please see the complete model for all other types.

Moreover, the binding tree defines only three types of sub-trees (i.e. lspid, src, and bidir) which is able to map the respective variants (vpn, recursive, and vpn-recursive) accordingly. For example, the key for opaque-type-src is [R, S, G, rd, recur-R, recur-RD], where basic type will specify (R, S,G,-, -, -), VPN type will specify (R, S,G, rd, -, -), recursive type will specify [R, S,G, -, recur-R, -] and VPN-recursive type will specify [R, S,G, -, recur-R, recur-rd].

It is important to take note of the following:

- o The address-family ipv4/ipv4 applies to "root" address in the mLDP binding tree. The other addresses (source, group, RP etc) do not have to be of the same address family type as the root.
- o The "recur-root-address" field applies to Recursive opaque type, and (recur-root-address, recur-rd) fields applies to VPN-Recursive opaque types as defined in [RFC6512]
- o In case of a recursive FEC, the address-family of the recur-root-address could be different than the address-family of the root address of original encapsulated MP FEC

The following diagram illustrates the FEC-label binding information structure for a P2MP (Transit IPv4 Source type) LSP on a branch/transit LSR:

```

FEC (root 2.2.2.2, S=192.168.1.1, G=224.1.1.1):
  type: p2mp
  upstream:
    advertised:
      peer 192.168.0.1:0, label 16000 (local)
  downstream:
    received:
      peer 192.168.0.2:0, label 17000 (remote)
      peer 192.168.0.3:0, label 18000 (remote)

```

Figure 19

The following diagram illustrates the FEC-label binding information structure for a similar MP2MP LSP on a branch/transit LSR:

```

FEC (root 2.2.2.2, RP=192.168.9.9, G=224.1.1.1):
  type: mp2mp
  upstream:
    advertised:
      peer 192.168.0.1:0, label 16000 (local)
    received:
      peer 192.168.0.1:0, label 17000 (remote)
  downstream:
    advertised:
      peer 192.168.0.2:0, label 16001 (local), MBB role=active
      peer 192.168.0.3:0, label 16002 (local), MBB role=standby
    received:
      peer 192.168.0.2:0, label 17001 (remote)
      peer 192.168.0.3:0, label 18001 (remote)

```

Figure 20

#### 4.3.1.3. Capabilities state

Like LDP, mLDP capabilities state comprise two types of information - global information and per-peer information.

#### 4.4. Notifications

mLDP notification module consists of notification related to changes in the operational state of an mLDP FEC. Following is a simplified graphical representation of the data model for mLDP notifications:

```

notifications:
  +---n mpls-ml dp-fec-event
    +---ro event-type?          oper-status-event-type
    +---ro tree-type?           multipoint-type
    +---ro root?                inet:ip-address
    +---ro (lsp-key-type)?
      +---:(lsp-id-based)
        | +---ro lsp-id?        uint16
      +---:(source-group-based)
        +---ro source-address?  inet:ip-address
        +---ro group-address?   inet:ip-address

```

Figure 21

#### 4.5. Actions

Currently, no RPCs/actions are defined for mLDP.

#### 5. Open Items

Following is a list of open items that are to be discussed and addressed in future revisions of this document:

- o Close on augmentation off "mpls" list in "ietf-mpls" defined in [I-D.saad-mpls-base-yang]
- o Align operational state modeling with other routing procols and [I-D.openconfig-netmod-opstate]
- o Complete the section on Protocol-centric implementations and all-vrfs
- o Specify default values for configuration parameters
- o Revisit and cut down on the scope of the document and number of features it is trying to cover
- o Split the model into a base and extended items
- o Add statistics for mLDP root LSPs and bindings
- o Extend the "Configured Leaf LSPs" for various type of opaque-types
- o Extend mLDP notifications for other types of opaque values as well
- o Close on single vs separate document for mLDP Yang

#### 6. YANG Specification

Following are actual YANG definition for LDP and mLDP constructs defined earlier in the document.

```
<CODE BEGINS> file "ietf-mpls-ldp@2016-07-08.yang" -->

module ietf-mpls-ldp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
  // replace with IANA namespace when assigned
  prefix ldp;

  import ietf-inet-types {
```

```
    prefix "inet";
}

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

import ietf-interfaces {
    prefix "if";
}

import ietf-ip {
    prefix "ip";
}

import ietf-routing {
    prefix "rt";
}

import ietf-mpls {
    prefix "mpls";
}

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

    WG Chair:   Loa Andersson
                <mailto:loa@pi.nu>

    WG Chair:   Ross Callon
                <mailto:rcallon@juniper.net>

    WG Chair:   George Swallow
                <mailto:swallow.ietf@gmail.com>

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

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

    Editor:     Xufeng Liu
                <mailto:xliu@kuatrotech.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) and Multipoint LDP (mLDP).";

revision 2016-07-08 {

description

"Initial revision.";

reference

"RFC XXXX: YANG Data Model for MPLS LDP and mLDP.";

}

/\*

\* Features

\*/

feature admin-down-config {

description

"This feature indicates that the system allows to configure administrative down on a VRF instance and a peer.";

}

feature all-af-policy-config {

description

"This feature indicates that the system allows to configure policies that are applied to all address families.";

}

feature capability-end-of-lib {

description

"This feature indicates that the system allows to configure LDP end-of-lib capability.";

}

feature capability-ml dp-hsmp {

description

"This feature indicates that the system allows to configure mLDP hub-and-spoke-multipoint capability.";

}

```
feature capability-mldp-node-protection {
  description
    "This feature indicates that the system allows to configure
    mLDP node-protection 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 global-session-authentication {
  description
    "This feature indicates that the system allows to configure
    authentication at global level.";
}

feature graceful-restart-helper-mode {
  description
    "This feature indicates that the system supports graceful
    restart helper mode.";
}

feature mldp {
  description
    "This feature indicates that the system supports Multicast
    LDP (mLDP).";
}

feature mldp-mofrr {
  description
    "This feature indicates that the system supports mLDP
    Multicast only FRR (MoFRR).";
}
```

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

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

/*
 * Typedefs
 */
typedef ldp-address-family {
  type identityref {
```



```
    base rt:address-family;
  }
  description
    "LDP address family type.";
}

typedef duration32-inf {
  type union {
    type uint32;
    type enumeration {
      enum "infinite" {
        description "The duration is infinite.";
      }
    }
  }
  units seconds;
  description
    "Duration represented as 32 bit seconds with infinite.";
}

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
    "Received or advertised.";
}

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 mpls-interface-ref {
  type leafref {
    path "/rt:routing/mpls:mpls/mpls:interface/mpls:name";
  }
  description
    "This type is used by data models that need to reference
    mpls interfaces.";
}

typedef multipoint-type {
  type enumeration {
    enum p2mp {
      description "Point to multipoint.";
    }
    enum mp2mp {
      description "Multipoint to multipoint.";
    }
  }
  description
    "p2mp or mp2mp.";
}

typedef neighbor-list-ref {
  type string;
  description
    "A type for a reference to a neighbor list.";
}

typedef peer-list-ref {
  type string;
  description
    "A type for a reference to a peer list.";
}

typedef prefix-list-ref {
  type string;
  description
```

```
    "A type for a reference to a prefix list.";
}

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.";
}

typedef route-distinguisher {
  type string {
  }
  description
    "Type definition for route distinguisher.";
  reference
    "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
}

/*
 * Identities
 */
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
    "Adjacency state attributes.";

  leaf-list flag {
    type identityref {
      base "adjacency-flag-base";
    }
    description "Adjacency flags.";
  }
  container hello-holdtime {
    description "Hello holdtime state.";
    leaf adjacent {
      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.";
    }
  }

  leaf next-hello {
    type uint16;
    units seconds;
    description "Time to send the next hello message.";
  }

  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 received.";
    }
} // statistics
} // adjacency-state-attributes

grouping basic-discovery-timers {
    description
        "Basic discovery timer attributes.";
    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";
    }
    leaf hello-interval {
        type uint16 {
            range 5..1200;
        }
        units seconds;
        description
            "The interval between consecutive LDP link Hello messages
            used in basic LDP discovery";
    }
} // basic-discovery-timers

grouping binding-address-state-attributes {
    description
        "Address binding attributes";
    leaf advertisement-type {
        type advertised-received;
```

```
        description
            "Received or advertised.";
    }
    leaf peer {
        type leafref {
            path "../.../.../.../.../peers/peer/lsr-id";
        }
        must "../advertisement-type = 'received'" {
            description
                "Applicable for received address.";
        }
        description
            "LDP peer from which this address is received.";
    } // peer
} // binding-address-state-attributes

grouping binding-label-state-attributes {
    description
        "Label binding attributes";
    list peer {
        key "peer advertisement-type";
        description
            "List of advertised and received peers.";
        leaf peer {
            type leafref {
                path "../.../.../.../.../.../peers/peer/lsr-id";
            }
            description
                "LDP peer from which this binding is received,
                or to which this binding is advertised.";
        }
        leaf advertisement-type {
            type advertised-received;
            description
                "Received or advertised.";
        }
        leaf label {
            type mpls:mpls-label;
            description
                "Advertised (outbound) or received (inbound)
                label.";
        }
        leaf used-in-forwarding {
            type boolean;
            description
                "'true' if the lable is used in forwarding.";
        }
    }
} // peer
```

```
} // binding-label-state-attributes

grouping extended-discovery-policy-attributes {
  description
    "LDP policy to control the acceptance of extended neighbor
    discovery hello messages.";
  container hello-accept {
    if-feature policy-extended-discovery-config;
    description
      "Extended discovery acceptance policies.";

    leaf enable {
      type boolean;
      description
        "'true' to accept; 'false' to deny.";
    }
    leaf neighbor-list {
      type neighbor-list-ref;
      description
        "The name of a peer ACL.";
    }
  } // hello-accept
} // extended-discovery-policy-attributes

grouping extended-discovery-timers {
  description
    "Extended discovery timer attributes.";
  leaf hello-holdtime {
    type uint16 {
      range 15..3600;
    }
    units seconds;
    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;
    description
      "The interval between consecutive LDP targeted Hello
      messages used in extended LDP discovery.";
  }
}
```

```
} // extended-discovery-timers

grouping global-attributes {
  description "Configuration attributes at global level.";

  uses instance-attributes;
} // global-attributes

grouping graceful-restart-attributes {
  description
    "Graceful restart configuration attributes.";
  container graceful-restart {
    description
      "Attributes for graceful restart.";
    leaf enable {
      type boolean;
      description
        "Enable or disable graceful restart.";
    }
    leaf helper-enable {
      if-feature graceful-restart-helper-mode;
      type boolean;
      description
        "Enable or disable graceful restart helper mode.";
    }
    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.";
    }
    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.";
    }
    leaf forwarding-holdtime {
      type uint16 {
```



```
        range 30..3600;
    }
    units seconds;
    description
        "Specifies the time interval, in seconds, before the
        termination of the recovery phase.";
    }
} // graceful-restart
} // graceful-restart-attributes

grouping graceful-restart-attributes-per-peer {
    description
        "Per peer graceful restart configuration attributes.";
    container graceful-restart {
        description
            "Attributes for graceful restart.";
        leaf enable {
            type boolean;
            description
                "Enable or disable graceful restart.";
        }
        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.";
        }
        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.";
        }
    }
} // graceful-restart
} // graceful-restart-attributes-per-peer

grouping instance-attributes {
    description "Configuration attributes at instance level.";
```

```
container capability {
  description "Configure capability.";
  container end-of-lib {
    if-feature capability-end-of-lib;
    description
      "Configure end-of-lib capability.";
    leaf enable {
      type boolean;
      description
        "Enable end-of-lib capability.";
    }
  }
  container typed-wildcard-fec {
    if-feature capability-typed-wildcard-fec;
    description
      "Configure typed-wildcard-fec capability.";
    leaf enable {
      type boolean;
      description
        "Enable typed-wildcard-fec capability.";
    }
  }
  container upstream-label-assignment {
    if-feature capability-upstream-label-assignment;
    description
      "Configure upstream label assignment capability.";
    leaf enable {
      type boolean;
      description
        "Enable upstream label assignment.";
    }
  }
  container mldp {
    if-feature mldp;

    description
      "Multipoint capabilities.";
    uses mldp-capabilities;
  }
} // capability

uses graceful-restart-attributes;

leaf igp-synchronization-delay {
  type uint16 {
    range 3..60;
  }
  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.";
  }
  leaf lsr-id {
    type yang:dotted-quad;
    description "Router ID.";
  }
} // instance-attributes

grouping ldp-adjacency-ref {
  description
    "An absolute reference to an LDP adjacency.";
  choice hello-adjacency-type {
    description
      "Interface or targeted adjacency.";
    case targeted {
      container targeted {
        description "Targeted adjacency.";
        leaf target-address {
          type inet:ip-address;
          description
            "The target address.";
        }
      } // targeted
    }
    case link {
      container link {
        description "Link adjacency.";
        leaf next-hop-interface {
          type mpls-interface-ref;
          description
            "Interface connecting to next-hop.";
        }
        leaf next-hop-address {
          type inet:ip-address;
          must "../next-hop-interface" {
            description
              "Applicable when interface is specified.";
          }
        }
        description
          "IP address of next-hop.";
      }
    } // link
  }
}
```

```
    }  
  } // ldp-adjacency-ref  
  
  grouping ldp-fec-event {  
    description  
      "A LDP FEC event.";   
    leaf prefix {  
      type inet:ip-prefix;  
      description  
        "FEC.";   
    }  
  } // ldp-fec-event  
  
  grouping ldp-peer-ref {  
    description  
      "An absolute reference to an LDP peer.";   
    leaf peer-ref {  
      type leafref {  
        path "/rt:routing/rt:control-plane-protocols/mpls-ldp/"  
          + "peers/peer/lsr-id";  
      }  
      description  
        "Reference to an LDP peer.";   
    }  
  } // ldp-peer-ref  
  
  grouping mldp-capabilities {  
    description  
      "mLDP capabilities.";   
    container p2mp {  
      description  
        "Configure point-to-multipoint capability.";   
      leaf enable {  
        type boolean;  
        description  
          "Enable point-to-multipoint.";   
      }  
    }  
    container mp2mp {  
      description  
        "Configure multipoint-to-multipoint capability.";   
      leaf enable {  
        type boolean;  
        description  
          "Enable multipoint-to-multipoint.";   
      }  
    }  
    container make-before-break {
```

```
description
  "Configure make-before-break capability.";
leaf enable {
  type boolean;
  description
    "Enable make-before-break.";
}
leaf switchover-delay {
  type uint16;
  units seconds;
  description
    "Switchover delay in seconds.";
}
leaf timeout {
  type uint16;
  units seconds;
  description
    "Timeout in seconds.";
}
}
container hub-and-spoke {
  if-feature capability-mldp-hsmp;
  description
    "Configure hub-and-spoke-multipoint capability.";
  reference
    "RFC7140: LDP Extensions for Hub and Spoke Multipoint
    Label Switched Path";
  leaf enable {
    type boolean;
    description
      "Enable hub-and-spoke-multipoint.";
  }
}
container node-protection {
  if-feature capability-mldp-node-protection;
  description
    "Configure node-protection capability.";
  reference
    "RFC7715: mLDP Node Protection.";
  leaf plr {
    type boolean;
    description
      "Point of Local Repair capable for MP LSP node
      protection.";
  }
}
container merge-point {
  description
    "Merge Point capable for MP LSP node protection.";
```

```
    leaf enable {
      type boolean;
      description
        "Enable merge point capability.";
    }
    leaf targeted-session-teardown-delay {
      type uint16;
      units seconds;
      description
        "Targeted session teardown delay.";
    }
  } // merge-point
} // mldp-capabilities

grouping mldp-configured-lsp-roots {
  description
    "mLDP roots containers.";

  container roots-ipv4 {

    when "../../af = 'ipv4'" {
      description
        "Only for IPv4.";
    }
    description
      "Configured IPv4 multicast LSPs.";
    list root {
      key "root-address";
      description
        "List of roots for configured multicast LSPs.";

      leaf root-address {
        type inet:ipv4-address;
        description
          "Root address.";
      }
    }

    list lsp {
      must "(lsp-id = 0 and source-address != '0.0.0.0' and "
        + "group-address != '0.0.0.0') or "
        + "(lsp-id != 0 and source-address = '0.0.0.0' and "
        + "group-address = '0.0.0.0')";
      description
        "A LSP can be identified by either <lsp-id> or
        <source-address, group-address>.";
    }
    key "lsp-id source-address group-address";
  }
}
```

```
    description
      "List of LSPs.";
    leaf lsp-id {
      type uint16;
      description "ID to identify the LSP.";
    }
    leaf source-address {
      type inet:ipv4-address;
      description
        "Source address.";
    }
    leaf group-address {
      type inet:ipv4-address-no-zone;
      description
        "Group address.";
    }
  } // list lsp
} // list root
} // roots-ipv4

container roots-ipv6 {

  when "../../../af = 'ipv6'" {
    description
      "Only for IPv6.";
  }
  description
    "Configured IPv6 multicast LSPs.";

  list root {
    key "root-address";
    description
      "List of roots for configured multicast LSPs.";

    leaf root-address {
      type inet:ipv6-address;
      description
        "Root address.";
    }
  }

  list lsp {
    must "(lsp-id = 0 and source-address != ':::' and "
      + "group-address != ':::') or "
      + "(lsp-id != 0 and source-address = ':::' and "
      + "group-address = ':::')" {
      description
        "A LSP can be identified by either <lsp-id> or
        <source-address, group-address>.";
    }
  }
}
```

```
    }
    key "lsp-id source-address group-address";
    description
      "List of LSPs.";
    leaf lsp-id {
      type uint16;
      description "ID to identify the LSP.";
    }
    leaf source-address {
      type inet:ipv6-address;
      description
        "Source address.";
    }
    leaf group-address {
      type inet:ipv6-address-no-zone;
      description
        "Group address.";
    }
  } // list lsp
} // list root
} // roots-ipv6
} // mldp-configured-lsp-roots

grouping mldp-fec-event {
  description
    "A mLDP FEC event.";
  leaf tree-type {
    type multipoint-type;
    description
      "p2mp or mp2mp.";
  }
  leaf root {
    type inet:ip-address;
    description
      "Root address.";
  }
  choice lsp-key-type {
    description
      "LSP ID based or source-group based .";
    case lsp-id-based {
      leaf lsp-id {
        type uint16;
        description
          "ID to identify the LSP.";
      }
    }
    case source-group-based {
      leaf source-address {

```



```
        type inet:ip-address;
        description

            "LSP source address.";
    }
    leaf group-address {
        type inet:ip-address;
        description
            "Multicast group address.";
    }
} // case source-group-based
}
} // mldp-fec-event

grouping mldp-binding-label-state-attributes {
    description
        "mLDP label binding attributes.";

    leaf multipoint-type {
        type multipoint-type;
        description
            "The type of mutipoint, p2mp or mp2mp.";
    }
    list peer {
        key "direction peer advertisement-type";
        description
            "List of advertised and received peers.";
        leaf direction {
            type downstream-upstream;
            description
                "Downstream or upstream.";
        }
        leaf peer {
            type leafref {
                path
                    ".../.../.../.../.../.../.../.../.../peers/peer/lsr-id";
            }
            description
                "LDP peer from which this binding is received,
                or to which this binding is advertised.";
        }
        leaf advertisement-type {
            type advertised-received;
            description
                "Advertised or received.";
        }
        leaf label {
            type mpls:mpls-label;
        }
    }
}
```

```
        description
            "Advertised (outbound) or received (inbound) label.";
    }
    leaf mbb-role {
        when "../direction = 'upstream'" {
            description
                "For upstream.";
        }
        type enumeration {
            enum none {
                description "MBB is not enabled.";
            }
            enum active {
                description "This LSP is active.";
            }
            enum inactive {
                description "This LSP is inactive.";
            }
        }
        description
            "The MBB status of this LSP.";
    }
    leaf mofrr-role {
        when "../direction = 'upstream'" {
            description
                "For upstream.";
        }
        type enumeration {
            enum none {
                description "MOFRR is not enabled.";
            }
            enum primary {
                description "This LSP is primary.";
            }
            enum backup {
                description "This LSP is backup.";
            }
        }
        description
            "The MOFRR status of this LSP.";
    }
} // peer
} // mldp-binding-label-state-attributes

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 outgoing label
      advertisements.";
  }
}
container accept {
  description
    "Label advertisement acceptance policies.";
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Applies the prefix list to incoming label
      advertisements.";
  }
} // accept
} // label-policy
} // peer-af-policy-container

grouping peer-attributes {
  description "Peer configuration attributes.";

  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.";
  }
  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.";
  }
}
```

```
    }
  } // peer-attributes

  grouping peer-authentication {
    description
      "Peer authentication attributes.";
    leaf session-authentication-md5-password {
      type string {
        length "1..80";
      }
      description
        "Assigns an encrypted MD5 password to an LDP
        peer";
    } // md5-password
  } // peer-authentication

  grouping peer-state-derived {
    description "Peer derived state attributes.";

    container label-advertisement-mode {
      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;
    description "Time to send the next KeepAlive message.";
  }

  leaf peer-ldp-id {
    type yang:dotted-quad;
    description "Peer LDP ID.";
  }
}
```

```
container received-peer-state {
  description "Peer features.";

  uses graceful-restart-attributes-per-peer;

  container capability {
    description "Configure capability.";
    container end-of-lib {
      description
        "Configure end-of-lib capability.";
      leaf enable {
        type boolean;
        description
          "Enable end-of-lib capability.";
      }
    }
  }
  container typed-wildcard-fec {
    description
      "Configure typed-wildcard-fec capability.";
    leaf enable {
      type boolean;
      description
        "Enable typed-wildcard-fec capability.";
    }
  }
  container upstream-label-assignment {
    description
      "Configure upstream label assignment capability.";
    leaf enable {
      type boolean;
      description
        "Enable upstream label assignment.";
    }
  }
  container mldp {
    if-feature mldp;
    description
      "Multipoint capabilities.";

    container p2mp {
      description
        "Configure point-to-multipoint capability.";
      leaf enable {
        type boolean;
        description
          "Enable point-to-multipoint.";
      }
    }
  }
}
```

```
container mp2mp {
  description
    "Configure multipoint-to-multipoint capability.";
  leaf enable {
    type boolean;
    description
      "Enable multipoint-to-multipoint.";
  }
}
container make-before-break {
  description
    "Configure make-before-break capability.";
  leaf enable {
    type boolean;
    description
      "Enable make-before-break.";
  }
}
container hub-and-spoke {
  description
    "Configure hub-and-spoke-multipoint capability.";
  reference
    "RFC7140: LDP Extensions for Hub and Spoke Multipoint
    Label Switched Path";
  leaf enable {
    type boolean;
    description
      "Enable hub-and-spoke-multipoint.";
  }
}
container node-protection {
  description
    "Configure node-protection capability.";
  reference
    "RFC7715: mLDP Node Protection.";
  leaf plr {
    type boolean;
    description
      "Point of Local Repair capable for MP LSP node
      protection.";
  }
  leaf merge-point {
    type boolean;
    description
      "Merge Point capable for MP LSP node protection.";
  } // merge-point
} // node-protection
} // mldp
```

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

  container session-holdtime {
    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.";
      }
    }
    description
      "Representing the operational status.";
  }

  container tcp-connection {
    description "TCP connection state.";
    leaf local-address {
      type inet:ip-address;
    }
  }
}
```

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

leaf up-time {
    type string;
    description "Up time. The interval format in ISO 8601.";
}

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.";
    }

    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 policy-container {
    description
        "LDP policy attributes.";
    container label-policy {
        description
            "Label policy attributes.";
        container independent-mode {
            description
                "Independent label policy attributes.";
            container assign {

                if-feature policy-label-assignment-config;
                description
                    "Label assignment policies";
                choice prefix-option {
                    description
                        "Use either prefix-list or host-routes-only.";
                    case prefix-list {
                        leaf prefix-list {
                            type prefix-list-ref;
                            description
                                "Assign labels according to certain prefixes.";
                        }
                    }
                    case host-routes-only {
                        leaf host-routes-only {
                            type boolean;
                            description
                                "'true' to apply host routes only.";
                        }
                    }
                }
            } // prefix-option
        }
    }
}
```

```
}
container advertise {
  description
    "Label advertising policies.";
  container 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 enable {
      type boolean;
      description
        "'true' to enable explicit null.";
    }
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Prefix list name. Applies the filters in the
         specified prefix list to label
         advertisements.
         If the prefix list is not specified, explicit
         null label advertisement is enabled for all
         directly connected prefixes.";
    }
  }
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Applies the prefix list to outgoing label
       advertisements.";
  }
}
container accept {
  description
    "Label advertisement acceptance policies.";
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Applies the prefix list to incoming label
       advertisements.";
  }
}
} // independent-mode
container ordered-mode {
  if-feature policy-ordered-label-config;
  description
```

```
    "Ordered label policy attributes.";
  container egress-lsr {
    description
      "Egress LSR label assignment policies";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Assign labels according to certain prefixes.";
    }
  }
  container advertise {
    description
      "Label advertising policies.";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Applies the prefix list to outgoing label
        advertisements.";
    }
  }
  container accept {
    description
      "Label advertisement acceptance policies.";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Applies the prefix list to incoming label
        advertisements.";
    }
  }
} // ordered-mode
} // label-policy
} // policy-container

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 messages sent or received.";
    }
} // statistics-peer-received-sent

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:control-plane-protocols" {
    description "LDP augmentation.";

    container mpls-ldp {
        presence "Container for LDP protocol.";
        description
            "Container for LDP protocol.";

        container global {
            description
                "Global attributes for LDP.";
            container config {
                description
                    "Configuration data.";
                uses global-attributes;
            }
            container state {
                config false;
                description
                    "Operational state data.";
                uses global-attributes;
            }
        }

        container mldp {
            if-feature mldp;
            description
                "mLDP attributes at per instance level. Defining
                attributes here does not enable any MP capabilities.
                MP capabilities need to be explicitly enabled under
                container capability.";

            container config {
                description
                    "Configuration data.";
                leaf enable {
                    type boolean;
                    description
                        "Enable mLDP.";
                }
            }
        }
    }
}
```

```
container state {
  config false;
  description

    "Operational state data.";
  leaf enable {
    type boolean;
    description
      "Enable mLDP.";
  }
}

list address-family {
  key "afi";
  description
    "Per-af params.";
  leaf afi {
    type ldp-address-family;
    description
      "Address family type value.";
  }
}

container config {
  description
    "Configuration data.";
  container multicast-only-frr {
    if-feature mldp-mofrr;
    description
      "Multicast only FRR (MoFRR) policy.";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Enables MoFRR for the specified access list.";
    }
  } // multicast-only-frr
  container recursive-fec {
    description
      "Recursive FEC policy.";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Enables recursive FEC for the specified access
        list.";
    }
  } // recursive-for
}
container state {
  config false;
```

```
description
  "Operational state data.";
container multicast-only-frr {
  if-feature mldp-mofrr;

  description
    "Multicast only FRR (MoFRR) policy.";
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Enables MoFRR for the specified access list.";
  }
} // multicast-only-frr
container recursive-fec {
  description
    "Recursive FEC policy.";
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Enables recursive FEC for the specified access
      list.";
  }
} // recursive-fec

container ipv4 {
  when "../..afi = 'ipv4'" {
    description
      "Only for IPv4.";
  }
  description
    "IPv4 state information.";
  container roots {
    description
      "IPv4 multicast LSP roots.";
    list root {
      key "root-address";
      description
        "List of roots for configured multicast LSPs.";

      leaf root-address {
        type inet:ipv4-address;
        description
          "Root address.";
      }

      leaf is-self {
        type boolean;
        description
```

```
        "This is the root.";
    }

    list reachability {
        key "address interface";
        description
            "A next hop for reachability to root,
            as a RIB view.";
        leaf address {
            type inet:ipv4-address;
            description
                "The next hop address to reach root.";
        }
        leaf interface {
            type mpls-interface-ref;
            description
                "Interface connecting to next-hop.";
        }
        leaf peer {
            type leafref {
                path
                    "../.../peers/peer/"
                    + "lsr-id";
            }
            description
                "LDP peer from which this next hop can be
                reached.";
        }
    }
} // list root
} // roots
container bindings {
    description
        "mLDP FEC to label bindings.";
    container opaque-type-lspid {
        description
            "The type of opaque value element is
            the generic LSP identifier";
        reference
            "RFC6388: Label Distribution Protocol
            Extensions for Point-to-Multipoint and
            Multipoint-to-Multipoint Label Switched
            Paths.";
        list fec-label {
            key
                "root-address lsp-id "
                + "recur-root-address recur-rd";
            description
```



```
    "List of FEC to label bindings.";
  leaf root-address {
    type inet:ipv4-address;
    description
      "Root address.";
  }
  leaf lsp-id {
    type uint32;
    description "ID to identify the LSP.";
  }
  leaf recur-root-address {
    type inet:ip-address;
    description
      "Recursive root address.";
    reference
      "RFC6512: Using Multipoint LDP When the
      Backbone Has No Route to the Root";
  }
  leaf recur-rd {
    type route-distinguisher;
    description
      "Route Distinguisher in the VPN-Recursive
      Opaque Value.";
    reference
      "RFC6512: Using Multipoint LDP When the
      Backbone Has No Route to the Root";
  }
  uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-lspid

container opaque-type-src {
  description
    "The type of opaque value element is
    the transit source TLV";
  reference
    "RFC6826: Multipoint LDP In-Band Signaling for
    Point-to-Multipoint and
    Multipoint-to-Multipoint Label Switched
    Paths.";
  list fec-label {
    key
      "root-address source-address group-address "
      + "rd recur-root-address recur-rd";
    description
      "List of FEC to label bindings.";
    leaf root-address {
      type inet:ipv4-address;
```

```
        description
            "Root address.";
    }
    leaf source-address {
        type inet:ip-address;
        description
            "Source address.";
    }
    leaf group-address {
        type inet:ip-address-no-zone;
        description
            "Group address.";
    }
    leaf rd {
        type route-distinguisher;
        description
            "Route Distinguisher.";
        reference
            "RFC7246: Multipoint Label Distribution
            Protocol In-Band Signaling in a Virtual
            Routing and Forwarding (VRF) Table
            Context.";
    }
    leaf recur-root-address {
        type inet:ip-address;
        description
            "Recursive root address.";
        reference
            "RFC6512: Using Multipoint LDP When the
            Backbone Has No Route to the Root";
    }
    leaf recur-rd {
        type route-distinguisher;
        description
            "Route Distinguisher in the VPN-Recursive
            Opaque Value.";
        reference
            "RFC6512: Using Multipoint LDP When the
            Backbone Has No Route to the Root";
    }
    uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-src

container opaque-type-bidir {
    description
        "The type of opaque value element is
        the generic LSP identifier";
```

```
reference
  "RFC6826: Multipoint LDP In-Band Signaling for
  Point-to-Multipoint and
  Multipoint-to-Multipoint Label Switched
  Paths.";
list fec-label {
  key
    "root-address rp group-address "
    + "rd recur-root-address recur-rd";
  description
    "List of FEC to label bindings.";
  leaf root-address {
    type inet:ipv4-address;
    description
      "Root address.";
  }
  leaf rp {
    type inet:ip-address;
    description
      "RP address.";
  }
  leaf group-address {
    type inet:ip-address-no-zone;
    description
      "Group address.";
  }
  leaf rd {
    type route-distinguisher;
    description
      "Route Distinguisher.";
    reference
      "RFC7246: Multipoint Label Distribution
      Protocol In-Band Signaling in a Virtual
      Routing and Forwarding (VRF) Table
      Context.";
  }
  leaf recur-root-address {
    type inet:ip-address;
    description
      "Recursive root address.";
    reference
      "RFC6512: Using Multipoint LDP When the
      Backbone Has No Route to the Root";
  }
  leaf recur-rd {
    type route-distinguisher;
    description
      "Route Distinguisher in the VPN-Recursive
```

```
        Opaque Value.";
        reference
        "RFC6512: Using Multipoint LDP When the
        Backbone Has No Route to the Root";
    }
    uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-bidir
} // bindings
} // ipv4

container ipv6 {
    when "../..afi = 'ipv6'" {
        description
        "Only for IPv6.";
    }
    description
    "IPv6 state information.";
    container roots {
        description
        "IPv6 multicast LSP roots.";
        list root {
            key "root-address";
            description
            "List of roots for configured multicast LSPs.";

            leaf root-address {
                type inet:ipv6-address;
                description
                "Root address.";
            }

            leaf is-self {
                type boolean;
                description
                "This is the root.";
            }
        }

        list reachability {
            key "address interface";
            description
            "A next hop for reachability to root,
            as a RIB view.";
            leaf address {
                type inet:ipv6-address;
                description
                "The next hop address to reach root.";
            }
        }
    }
}
```

```
    leaf interface {
      type mpls-interface-ref;
      description
        "Interface connecting to next-hop.";
    }
    leaf peer {
      type leafref {
        path
          "../.../.../.../.../.../.../.../peers/peer/"
          + "lsr-id";
      }
      description
        "LDP peer from which this next hop can be
        reached.";
    }
  }
} // list root
} // roots
container bindings {
  description
    "mLDP FEC to label bindings.";
  container opaque-type-lspid {
    description
      "The type of opaque value element is
      the generic LSP identifier";
    reference
      "RFC6388: Label Distribution Protocol
      Extensions for Point-to-Multipoint and
      Multipoint-to-Multipoint Label Switched
      Paths.";
    list fec-label {
      key
        "root-address lsp-id "
        + "recur-root-address recur-rd";
      description
        "List of FEC to label bindings.";
      leaf root-address {
        type inet:ipv6-address;
        description
          "Root address.";
      }
      leaf lsp-id {
        type uint32;
        description "ID to identify the LSP.";
      }
      leaf recur-root-address {
        type inet:ip-address;
        description

```

```
        "Recursive root address.";
        reference
        "RFC6512: Using Multipoint LDP When the
        Backbone Has No Route to the Root";
    }
    leaf recur-rd {
        type route-distinguisher;
        description
        "Route Distinguisher in the VPN-Recursive
        Opaque Value.";
        reference
        "RFC6512: Using Multipoint LDP When the
        Backbone Has No Route to the Root";
    }
    uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-lspid

container opaque-type-src {
    description
    "The type of opaque value element is
    the transit Source TLV";
    reference
    "RFC6826: Multipoint LDP In-Band Signaling for
    Point-to-Multipoint and
    Multipoint-to-Multipoint Label Switched
    Paths.";
    list fec-label {
        key
        "root-address source-address group-address "
        + "rd recur-root-address recur-rd";
        description
        "List of FEC to label bindings.";
        leaf root-address {
            type inet:ipv6-address;
            description
            "Root address.";
        }
        leaf source-address {
            type inet:ip-address;
            description
            "Source address.";
        }
        leaf group-address {
            type inet:ip-address-no-zone;
            description
            "Group address.";
        }
    }
}
```

```
leaf rd {
  type route-distinguisher;
  description
    "Route Distinguisher.";
  reference
    "RFC7246: Multipoint Label Distribution
    Protocol In-Band Signaling in a Virtual
    Routing and Forwarding (VRF) Table
    Context.";
}
leaf recur-root-address {
  type inet:ip-address;
  description
    "Recursive root address.";
  reference
    "RFC6512: Using Multipoint LDP When the
    Backbone Has No Route to the Root";
}
leaf recur-rd {
  type route-distinguisher;
  description
    "Route Distinguisher in the VPN-Recursive
    Opaque Value.";
  reference
    "RFC6512: Using Multipoint LDP When the
    Backbone Has No Route to the Root";
}
uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-src

container opaque-type-bidir {
  description
    "The type of opaque value element is
    the generic LSP identifier";
  reference
    "RFC6826: Multipoint LDP In-Band Signaling for
    Point-to-Multipoint and
    Multipoint-to-Multipoint Label Switched
    Paths.";
  list fec-label {
    key
      "root-address rp group-address "
      + "rd recur-root-address recur-rd";
    description
      "List of FEC to label bindings.";
    leaf root-address {
      type inet:ipv6-address;
```

```
        description
            "Root address.";
    }
    leaf rp {
        type inet:ip-address;
        description
            "RP address.";
    }
    leaf group-address {
        type inet:ip-address-no-zone;
        description
            "Group address.";
    }
    leaf rd {
        type route-distinguisher;
        description
            "Route Distinguisher.";
        reference
            "RFC7246: Multipoint Label Distribution
            Protocol In-Band Signaling in a Virtual
            Routing and Forwarding (VRF) Table
            Context.";
    }
    leaf recur-root-address {
        type inet:ip-address;
        description
            "Recursive root address.";
        reference
            "RFC6512: Using Multipoint LDP When the
            Backbone Has No Route to the Root";
    }
    leaf recur-rd {
        type route-distinguisher;
        description
            "Route Distinguisher in the VPN-Recursive
            Opaque Value.";
        reference
            "RFC6512: Using Multipoint LDP When the
            Backbone Has No Route to the Root";
    }
    uses mldp-binding-label-state-attributes;
} // fec-label
} // opaque-type-bidir
} // bindings
} // ipv6
} // state

container configured-leaf-lsps {
```



```
description
  "Configured multicast LSPs.";

container p2mp {
  description
    "Configured point-to-multipoint LSPs.";
  uses mldp-configured-lsp-roots;
}
container mp2mp {
  description
    "Configured multipoint-to-multipoint LSPs.";
  uses mldp-configured-lsp-roots;
}
} // configured-leaf-lsps
} // list address-family
} // mldp

list address-family {
  key "afi";
  description
    "Per-vrf per-af params.";
  leaf afi {
    type ldp-address-family;
    description
      "Address family type value.";
  }
}

container config {
  description
    "Configuration data.";
  leaf enable {
    type boolean;
    description
      "'true' to enable the address family.";
  }
  uses policy-container;

  container ipv4 {
    when "../..//afi = 'ipv4'" {
      description
        "Only for IPv4.";
    }
    description
      "IPv4 address family.";
    leaf transport-address {
      type inet:ipv4-address;
      description
        "The transport address advertised in LDP Hello
```

```
        messages.";
    }
} // ipv4
container ipv6 {
    when "../..afi = 'ipv6'" {
        description
            "Only for IPv6.";
    }
    description
        "IPv6 address family.";
    leaf transport-address {
        type inet:ipv6-address;
        description
            "The transport address advertised in LDP Hello
            messages.";
    }
} // ipv6
}
container state {
    config false;
    description
        "Operational state data.";
    leaf enable {
        type boolean;
        description
            "'true' to enable the address family.";
    }
}

uses policy-container;

container ipv4 {
    when "../..afi = 'ipv4'" {
        description
            "Only for IPv4.";
    }
    description
        "IPv4 address family.";
    leaf transport-address {
        type inet:ipv4-address;
        description
            "The transport address advertised in LDP Hello
            messages.";
    }
}

container bindings {
    description
        "LDP address and label binding information.";
    list address {
```

```
    key "address";
    description
        "List of address bindings.";
    leaf address {
        type inet:ipv4-address;
        description
            "Binding address.";
    }
    uses binding-address-state-attributes;
} // binding-address

list fec-label {
    key "fec";
    description
        "List of label bindings.";
    leaf fec {
        type inet:ipv4-prefix;
        description
            "Prefix FEC.";
    }
    uses binding-label-state-attributes;
} // fec-label
} // binding
} // ipv4
container ipv6 {
    when "../..afi = 'ipv6'" {
        description
            "Only for IPv6.";
    }
    description
        "IPv6 address family.";
    leaf transport-address {
        type inet:ipv6-address;
        description
            "The transport address advertised in LDP Hello
            messages.";
    }
}

container binding {
    description
        "LDP address and label binding information.";
    list address {
        key "address";
        description
            "List of address bindings.";
        leaf address {
            type inet:ipv6-address;
            description
```

```
        "Binding address.";
    }
    uses binding-address-state-attributes;
} // binding-address

list fec-label {
    key "fec";
    description
        "List of label bindings.";
    leaf fec {
        type inet:ipv6-prefix;
        description
            "Prefix FEC.";
    }
    uses binding-label-state-attributes;
} // fec-label
} // binding
} // ipv6
} // state
} // address-family

container discovery {
    description
        "Neighbor discovery configuration.";

    container interfaces {
        description
            "A list of interfaces for basic discovery.";
        container config {
            description
                "Configuration data.";
            uses basic-discovery-timers;
        }
        container state {
            config false;
            description

                "Operational state data.";
            uses basic-discovery-timers;
        }
    }

    list interface {
        key "interface";
        description
            "List of LDP interfaces.";
        leaf interface {
            type mpls-interface-ref;
            description
```

```
        "Interface.";
    }
    container config {
        description
            "Configuration data.";
        uses basic-discovery-timers {
            if-feature per-interface-timer-config;
        }
        leaf igp-synchronization-delay {
            if-feature per-interface-timer-config;
            type uint16 {
                range 3..60;
            }
            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.";
        }
    }
    container state {
        config false;
        description
            "Operational state data.";
        uses basic-discovery-timers {
            if-feature per-interface-timer-config;
        }
        leaf igp-synchronization-delay {
            if-feature per-interface-timer-config;
            type uint16 {
                range 3..60;
            }
            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.";
        }
        leaf next-hello {
            type uint16;
            units seconds;
            description "Time to send the next hello message.";
        }
    }
} // state
```

```
list address-family {
  key "afi";
  description
    "Per-vrf per-af params.";
  leaf afi {
    type ldp-address-family;
    description
      "Address family type value.";
  }
  container config {
    description
      "Configuration data.";
    leaf enable {
      type boolean;
      description
        "Enable the address family on the interface.";
    }
  }

  container ipv4 {
    must "/if:interfaces/if:interface"
      + "[name = current()/../../../../interface]/"
      + "ip:ipv4" {
      description
        "Only if IPv4 is enabled on the interface.";
    }
    description
      "IPv4 address family.";
    leaf transport-address {
      type union {
        type enumeration {
          enum "use-interface-address" {
            description
              "Use interface address as the transport
              address.";
          }
        }
        type inet:ipv4-address;
      }
    }
    description
      "IP address to be advertised as the LDP
      transport address.";
  }
}

container ipv6 {
  must "/if:interfaces/if:interface"
    + "[name = current()/../../../../interface]/"
    + "ip:ipv6" {
  }
```

```
        description
            "Only if IPv6 is enabled on the interface.";
    }
    description
        "IPv6 address family.";
    leaf transport-address {
        type union {
            type enumeration {
                enum "use-interface-address" {
                    description
                        "Use interface address as the transport
                        address.";
                }
            }
            type inet:ipv4-address;
        }
    }
    description
        "IP address to be advertised as the LDP
        transport address.";
    }
} // ipv6
}
container state {
    config false;
    description
        "Operational state data.";
    leaf enable {
        type boolean;
        description
            "Enable the address family on the interface.";
    }
}

container ipv4 {
    must "/if:interfaces/if:interface"
        + "[name = current()/../../../interface]/"
        + "ip:ipv4" {
        description
            "Only if IPv4 is enabled on the interface.";
    }
    description
        "IPv4 address family.";
    leaf transport-address {
        type union {
            type enumeration {

                enum "use-interface-address" {
                    description
                        "Use interface address as the transport
```

```
        address.";
    }
}
type inet:ipv4-address;
}
description
    "IP address to be advertised as the LDP
    transport address.";
}

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

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

    uses adjacency-state-attributes;

    leaf peer {
        type leafref {
            path "../.../.../.../.../.../peers/peer/"
                + "lsr-id";
        }
        description
            "LDP peer from this adjacency.";
    }
} // hello-adjacencies
}

container ipv6 {
    must "/if:interfaces/if:interface"
        + "[name = current()/../.../.../interface]/"
        + "ip:ipv6" {
        description
            "Only if IPv6 is enabled on the interface.";
    }
}
description
    "IPv6 address family.";
leaf transport-address {
    type union {
        type enumeration {
            enum "use-interface-address" {
                description
                    "Use interface address as the transport
                    address.";
            }
        }
    }
}
```



```

        }
    }
    type inet:ipv4-address;
}
description
    "IP address to be advertised as the LDP
    transport address.";
}

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

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

    uses adjacency-state-attributes;

    leaf peer {
        type leafref {
            path "../../../../../peers/peer/"
                + "lsr-id";
        }
        description
            "LDP peer from this adjacency.";
    }
} // hello-adjacencies
} // ipv6
}
} // address-family
} // list interface
} // interfaces

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

        description
            "Configuration data.";
        uses extended-discovery-timers;
        uses extended-discovery-policy-attributes;
    }
    container state {

```

```
    config false;
    description
      "Operational state data.";
    uses extended-discovery-timers;
    uses extended-discovery-policy-attributes;
  }

  list address-family {
    key "afi";
    description
      "Per-af params.";
    leaf afi {
      type ldp-address-family;
      description
        "Address family type value.";
    }
  }

  container state {
    config false;
    description
      "Operational state data.";

    container ipv4 {
      when "../..//afi = 'ipv4'" {
        description
          "For IPv4.";
      }
      description
        "IPv4 address family.";
      list hello-adjacencies {
        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 peer {
        type leafref {
```

```

        path "../../../../../../../peers/peer/"
          + "lsr-id";
      }
      description
        "LDP peer from this adjacency.";
    }
  } // hello-adjacencies
} // ipv4

container ipv6 {
  when "../../../afi = 'ipv6'" {
    description
      "For IPv6.";
  }
  description
    "IPv6 address family.";
  list hello-adjacencies {
    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 adjacency-state-attributes;

  leaf peer {
    type leafref {
      path "../../../../../../../peers/peer/"
        + "lsr-id";
    }
    description
      "LDP peer from this adjacency.";
  }
} // hello-adjacencies
} // ipv6
} // state

container ipv4 {
  when "../../../afi = 'ipv4'" {
    description

```

```
        "For IPv4.";
    }
    description
        "IPv4 address family.";
    list target {
        key "adjacent-address";
        description
            "Targeted discovery params.";

        leaf adjacent-address {
            type inet:ipv4-address;
            description
                "Configures a remote LDP neighbor and enables
                 extended LDP discovery of the specified
                 neighbor.";
        }
    }
    container config {
        description
            "Configuration data.";
        leaf enable {
            type boolean;
            description
                "Enable the target.";
        }
        leaf local-address {
            type inet:ipv4-address;
            description
                "The local address.";
        }
    }
    container state {
        config false;
        description
            "Operational state data.";
        leaf enable {
            type boolean;
            description
                "Enable the target.";
        }
        leaf local-address {
            type inet:ipv4-address;
            description
                "The local address.";
        }
    }
    } // state
} // ipv4
container ipv6 {
```

```
when "../afi = 'ipv6'" {
  description
    "For IPv6.";
}
description
  "IPv6 address family.";
list target {
  key "adjacent-address";
  description
    "Targeted discovery params.";

  leaf adjacent-address {
    type inet:ipv6-address;
    description
      "Configures a remote LDP neighbor and enables
       extended LDP discovery of the specified
       neighbor.";
  }
  container config {
    description
      "Configuration data.";
    leaf enable {
      type boolean;
      description
        "Enable the target.";
    }
    leaf local-address {
      type inet:ipv6-address;
      description
        "The local address.";
    }
  }
}
container state {
  config false;
  description
    "Operational state data.";
  leaf enable {
    type boolean;
    description
      "Enable the target.";
  }
  leaf local-address {
    type inet:ipv6-address;
    description
      "The local address.";
  }
} // state
}
```

```
        } // ipv6
      } // address-family
    } // targeted
  } // discovery

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

    container interfaces {
      description
        "A list of interfaces on which forwarding is
        disabled.";

      list interface {
        key "interface";
        description
          "List of LDP interfaces.";
        leaf interface {
          type mpls-interface-ref;
          description
            "Interface.";
        }
      }
      list address-family {
        key "afi";
        description
          "Per-vrf per-af params.";
        leaf afi {
          type ldp-address-family;
          description
            "Address family type value.";
        }
      }
      container config {
        description
          "Configuration data.";
        leaf ldp-disable {
          type boolean;
          description
            "Disable LDP forwarding on the interface.";
        }
        leaf mldp-disable {
          if-feature mldp;
          type boolean;
          description
            "Disable mLDP forwarding on the interface.";
        }
      }
    }
  }
```

```
        container state {
            config false;
            description
                "Operational state data.";
            leaf ldp-disable {
                type boolean;
                description
                    "Disable LDP forwarding on the interface.";
            }
            leaf mldp-disable {
                if-feature mldp;

                type boolean;
                description
                    "Disable mLDP forwarding on the interface.";
            }
        } // address-family
    } // list interface
} // interfaces
} // forwarding-nexthop
uses policy-container {
    if-feature all-af-policy-config;
}
} // global

container peers {
    description
        "Peers configuration attributes.";

    container config {
        description
            "Configuration data.";
        uses peer-authentication {
            if-feature global-session-authentication;
        }
        uses peer-attributes;

        container session-downstream-on-demand {
            if-feature session-downstream-on-demand-config;
            description
                "Session downstream-on-demand attributes.";
            leaf enable {
                type boolean;
                description
                    "'true' if session downstream-on-demand is enabled.";
            }
            leaf peer-list {
```

```
        type peer-list-ref;
        description
            "The name of a peer ACL.";
    }
}
}
container state {
    config false;
    description
        "Operational state data.";
    uses peer-authentication {
        if-feature global-session-authentication;
    }
    uses peer-attributes;

    container session-downstream-on-demand {
        if-feature session-downstream-on-demand-config;
        description
            "Session downstream-on-demand attributes.";
        leaf enable {
            type boolean;
            description
                "'true' if session downstream-on-demand is enabled.";
        }
        leaf peer-list {
            type peer-list-ref;
            description
                "The name of a peer ACL.";
        }
    }
}

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

    leaf lsr-id {
        type yang:dotted-quad;
        description "LSR ID.";
    }

    container config {
        description
            "Configuration data.";
        leaf admin-down {
            type boolean;
            default false;
        }
    }
}
```



```
        description
            "'true' to disable the peer.";
    }

    container capability {
        description
            "Per peer capability";
        container mldp {
            if-feature mldp;
            description
                "mLDP capabilities.";
            uses mldp-capabilities;
        }
    }

    uses peer-af-policy-container {
        if-feature all-af-policy-config;
    }

    uses peer-authentication;

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

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

    container address-family {
        description
            "Per-vrf per-af params.";
        container ipv4 {
            description
                "IPv4 address family.";
            uses peer-af-policy-container;
        }
        container ipv6 {
            description
                "IPv6 address family.";
            uses peer-af-policy-container;
        } // ipv6
    } // address-family
}

container state {
    config false;
    description
        "Operational state data.";
```

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

container capability {
    description
        "Per peer capability";
    container mldp {
        if-feature mldp;
        description
            "mLDP capabilities.";
        uses mldp-capabilities;
    }
}

uses peer-af-policy-container {
    if-feature all-af-policy-config;
}

uses peer-authentication;

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

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

container address-family {
    description
        "Per-vrf per-af params.";
    container ipv4 {
        description
            "IPv4 address family.";
        uses peer-af-policy-container;

        list hello-adjacencies {
            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 mpls-interface-ref;
      description "Interface for this adjacency.";
    }
  } // hello-adjacencies
} // ipv4
container ipv6 {
  description
    "IPv6 address family.";
  uses peer-af-policy-container;

  list hello-adjacencies {
    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 adjacency-state-attributes;

  leaf interface {
    type mpls-interface-ref;
    description "Interface for this adjacency.";
  }
} // hello-adjacencies
} // ipv6
} // address-family

uses peer-state-derived;
} // state
} // list peer
```

```
    } // peers
  } // container mpls-ldp
}

/*
 * RPCs
 */
rpc mpls-ldp-clear-peer {
  description
    "Clears the session to the peer.";
  input {
    leaf lsr-id {
      type union {
        type yang:dotted-quad;
        type uint32;
      }
      description
        "LSR ID of peer to be cleared. If this is not provided
        then all peers are cleared";
    }
  }
}

rpc mpls-ldp-clear-hello-adjacency {
  description
    "Clears the hello adjacency";
  input {
    container hello-adjacency {
      description
        "Link adjacency or targettted adjacency. If this is not
        provided then all hello adjacencies are cleared";
      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";
            }
          } // targeted
        }
        case link {
          container link {

```

```

        presence "Present to clear link adjacencies.";
        description
            "Clear link adjacencies.";
        leaf next-hop-interface {
            type mpls-interface-ref;
            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.";
        } // next-hop-address
    } // link
}
}
}
}
}

rpc mpls-ldp-clear-peer-statistics {
    description
        "Clears protocol statistics (e.g. sent and received
        counters).";
    input {
        leaf lsr-id {
            type union {
                type yang:dotted-quad;
                type uint32;
            }
            description
                "LSR ID of peer whose statistic are to be cleared.
                If this is not provided then all peers statistics are
                cleared";
        }
    }
}

/*
 * 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.";
    }
    uses ldp-peer-ref;
}

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.";
    }
    uses ldp-adjacency-ref;
}

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.";
    }
    uses ldp-fec-event;
}

notification mpls-mldp-fec-event {
    description
        "Notification event for a change of FEC status.";
    leaf event-type {
        type oper-status-event-type;
        description "Event type.";
    }
    uses mldp-fec-event;
}
}

<CODE ENDS>
```

Figure 22

## 7. Security Considerations

The configuration, state, action and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC6536].

## 8. IANA Considerations

This document does not extend LDP or mLDP base protocol specification and hence there are no IANA considerations.

Note to the RFC Editor: Please remove IANA section before the publication.

## 9. Acknowledgments

The authors would like to acknowledge Eddie Chami, Nagendra Kumar, Mannan Venkatesan, Pavan Beeram for their contribution to this document. We also acknowledge Ladislav Lhotka for his useful comments as the YANG Doctor.

## 10. References

### 10.1. Normative References

- [I-D.ietf-netmod-routing-cfg]  
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.
- [I-D.rtgyangdt-rtgwg-ni-model]  
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Network Instance Model", draft-rtgyangdt-rtgwg-ni-model-00 (work in progress), May 2016.
- [I-D.saad-mpls-base-yang]  
Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base", draft-saad-mpls-base-yang-00 (work in progress), May 2016.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <<http://www.rfc-editor.org/info/rfc3478>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<http://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, <<http://www.rfc-editor.org/info/rfc5331>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc5919>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://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, <<http://www.rfc-editor.org/info/rfc6241>>.



- [RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <<http://www.rfc-editor.org/info/rfc6388>>.
- [RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <<http://www.rfc-editor.org/info/rfc6389>>.
- [RFC6512] Wijnands, IJ., Rosen, E., Napierala, M., and N. Leymann, "Using Multipoint LDP When the Backbone Has No Route to the Root", RFC 6512, DOI 10.17487/RFC6512, February 2012, <<http://www.rfc-editor.org/info/rfc6512>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <<http://www.rfc-editor.org/info/rfc6826>>.
- [RFC7060] Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP Multipoint Extensions on Targeted LDP Sessions", RFC 7060, DOI 10.17487/RFC7060, November 2013, <<http://www.rfc-editor.org/info/rfc7060>>.
- [RFC7140] Jin, L., Jounay, F., Wijnands, IJ., and N. Leymann, "LDP Extensions for Hub and Spoke Multipoint Label Switched Path", RFC 7140, DOI 10.17487/RFC7140, March 2014, <<http://www.rfc-editor.org/info/rfc7140>>.
- [RFC7246] Wijnands, IJ., Ed., Hitchen, P., Leymann, N., Henderickx, W., Gulko, A., and J. Tantsura, "Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context", RFC 7246, DOI 10.17487/RFC7246, June 2014, <<http://www.rfc-editor.org/info/rfc7246>>.
- [RFC7438] Wijnands, IJ., Ed., Rosen, E., Gulko, A., Joerde, U., and J. Tantsura, "Multipoint LDP (mLDP) In-Band Signaling with Wildcards", RFC 7438, DOI 10.17487/RFC7438, January 2015, <<http://www.rfc-editor.org/info/rfc7438>>.

- [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, <<http://www.rfc-editor.org/info/rfc7552>>.
- [RFC7715] Wijnands, IJ., Ed., Raza, K., Atlas, A., Tantsura, J., and Q. Zhao, "Multipoint LDP (mLDP) Node Protection", RFC 7715, DOI 10.17487/RFC7715, January 2016, <<http://www.rfc-editor.org/info/rfc7715>>.

## 10.2. Informative References

- [I-D.ietf-rtgwg-policy-model]  
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase,  
"Routing Policy Configuration Model for Service Provider  
Networks", draft-ietf-rtgwg-policy-model-01 (work in  
progress), April 2016.
- [I-D.iwijnand-mpls-mlbp-multi-topology]  
Wijnands, I. and K. Raza, "mLDP Extensions for Multi  
Topology Routing", draft-iwijnand-mpls-mlbp-multi-  
topology-03 (work in progress), June 2013.
- [I-D.openconfig-netmod-opstate]  
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling  
of Operational State Data in YANG", draft-openconfig-  
netmod-opstate-01 (work in progress), July 2015.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private  
Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February  
2006, <<http://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,  
<<http://www.rfc-editor.org/info/rfc7307>>.

## Appendix A. Additional Contributors

Stephane Litkowski  
Orange.  
Email: [stephane.litkowski@orange.com](mailto:stephane.litkowski@orange.com)

Reshad Rahman  
Cisco Systems Inc.  
Email: [rrahman@cisco.com](mailto:rrahman@cisco.com)

Danial Johari  
Cisco Systems Inc.  
Email: [dajohari@cisco.com](mailto:dajohari@cisco.com)

## Authors' Addresses

Kamran Raza  
Cisco Systems, Inc.  
Email: [skraza@cisco.com](mailto:skraza@cisco.com)

Rajiv Asati  
Cisco Systems, Inc.  
Email: [rajiva@cisco.com](mailto:rajiva@cisco.com)

Sowmya Krishnaswamy  
Cisco Systems, Inc.  
Email: sowkrish@cisco.com

Xufeng Liu  
Ericsson  
Email: xliu@kuatrotech.com

Jeff Tantsura  
Ericsson  
Email: jeff.tantsura@ericsson.com

Santosh Esale  
Juniper Networks  
Email: sesale@juniper.net

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

Loa Andersson  
Huawei Technologies  
Email: loa@pi.nu

Himanshu Shah  
Ciena Corporation  
Email: hshah@ciena.com

Matthew Bocci  
Alcatel-Lucent  
Email: matthew.bocci@alcatel-lucent.com

Internet Engineering Task Force  
Internet-Draft  
Obsoletes: 3107 (if approved)  
Intended status: Standards Track  
Expires: December 2, 2016

E. Rosen, Ed.  
Juniper Networks, Inc.  
May 31, 2016

Using BGP to Bind MPLS Labels to Address Prefixes  
draft-rosen-mpls-rfc3107bis-01

Abstract

This document specifies a set of procedures for using BGP to advertise that a specified router has bound a specified MPLS label (or a specified sequence of MPLS labels, organized as a contiguous part of a label stack) to a specified address prefix. This can be done by sending a BGP UPDATE message whose Network Layer Reachability Information field contains both the prefix and the MPLS label(s), and whose Next Hop field identifies the node at which said prefix is bound to said label(s). This document obsoletes RFC 3107.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 2, 2016.

Copyright Notice

Copyright (c) 2016 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 (<http://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. Using BGP to Bind an Address Prefix to One or More MPLS Labels . . . . .	4
2.1. Multiple Labels Capability . . . . .	5
2.2. NLRI Encoding when the Multiple Labels Capability is Not Used . . . . .	8
2.3. NLRI Encoding when the Multiple Labels Capability is Used . . . . .	9
2.4. How to Explicitly Withdraw the Binding of a Label to a Prefix . . . . .	11
2.5. Changing the Label that is Bound to a Prefix . . . . .	12
3. Installing and/or Propagating SAFI-4 or SAFI-128 Routes . . . . .	13
3.1. Comparability of Routes . . . . .	13
3.2. Modification of Label(s) Field When Propagating . . . . .	14
3.2.1. When the Next Hop Field is Unchanged . . . . .	14
3.2.2. When the Next Hop Field is Changed . . . . .	14
4. Data Plane . . . . .	15
5. Relationship Between SAFI-4 and SAFI-1 Routes . . . . .	17
6. IANA Considerations . . . . .	18
7. Security Considerations . . . . .	18
8. Acknowledgements . . . . .	19
9. References . . . . .	19
9.1. Normative References . . . . .	19
9.2. Informative References . . . . .	21
Author's Address . . . . .	21

## 1. Introduction

[RFC3107] specifies encodings and procedures for using BGP to indicate that a particular router has bound either a single MPLS label or a sequence of MPLS labels (organized as a contiguous part of an MPLS label stack) ([RFC3031], [RFC3032]) to a particular address prefix. This is done by sending a BGP UPDATE message whose Network Layer Reachability Information field contains both the prefix and the MPLS label(s), and whose Next Hop field identifies the node at which said prefix is bound to said label(s). Each such UPDATE also advertises a path to the specified prefix, via the specified next hop.

Although there are many implementations and deployments of [RFC3107], there are a number of issues with [RFC3107] that have impeded

interoperability in the past, and may potentially impede interoperability in the future:

- o Although [RFC3107] specifies an encoding that allows a sequence of MPLS labels (rather than just a single label) to be bound to a prefix, it does not specify the semantics of binding a sequence of labels to a prefix.
- o Many implementations of [RFC3107] assume that only one label will be bound to a prefix, and cannot properly process a BGP UPDATE message that binds a sequence of labels to a prefix. Thus an implementation attempting to provide this feature is likely to experience problems interoperating with other implementations.
- o [RFC3107]'s procedures for withdrawing the binding of a label or sequence of labels to a prefix are not specified clearly and correctly.
- o [RFC3107] specifies an optional feature, known as "Advertising Multiple Routes to a Destination", that, to the best of the author's knowledge, has never been implemented as specified. The functionality that this feature was intended to provide can and has been implemented in a different way using the procedures of [ADD-PATHS], which were not available at the time that [RFC3107] was written. In [RFC3107], this feature was controlled by a BGP Capability Code that has never been implemented, and is now essentially obsolete.
- o It is possible for a BGP speaker to receive two BGP UPDATES that advertise paths to the same address prefix, where one UPDATE binds a label (or sequence of labels) to the prefix and the other does not. [RFC3107] is silent on the issue of how the presence of two such UPDATES impacts the BGP decision process, and does not say explicitly whether one or the other or both of these UPDATES should be propagated. This has led different implementations to handle this situation in different ways.
- o Much of [RFC3107] applies to the VPN-IPv4 ([RFC4364]) and VPN-IPv6 ([RFC4659]) address families, but those address families are not mentioned in [RFC3107].

This document replaces and obsoletes [RFC3107]. It defines a new BGP Capability to be used when binding a sequence of labels to a prefix; by using this Capability, the interoperability problems alluded to above can be avoided. This document also removes the unimplemented "Advertising Multiple Routes to a Destination" feature, while specifying how to use [ADD-PATHS] to provide the same functionality. This document also addresses the issue of the how UPDATES that bind

labels to a given prefix interact with UPDATES that advertise paths to that prefix but do not bind labels to it. However, for backwards compatibility, it declares most of these interactions to be matters of local policy.

The places where this specification differs from [RFC3107] are indicated in the text. It is believed that implementations that conform to the current document will interoperate correctly with existing deployed implementations of [RFC3107].

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

## 2. Using BGP to Bind an Address Prefix to One or More MPLS Labels

BGP may be used to advertise that a particular node, call it N, has bound a particular MPLS label, or a particular sequence of MPLS labels (organized as a contiguous part of an MPLS label stack), to a particular address prefix. This is done by sending a Multiprotocol BGP UPDATE message, i.e., an UPDATE message with an MP\_REACH\_NLRI attribute ([RFC4760]). The "Network Address of Next Hop" field of that attribute contains an IP address of node N. The label(s) and the prefix are encoded in the NLRI field of the MP\_REACH\_NLRI. The encoding of the NLRI field is specified in Sections 2.2 and 2.3.

If the prefix is an IPv4 address prefix or a VPN-IPv4 ([RFC4364]) address prefix, the Address Family Identifier (AFI) of the MP\_REACH\_NLRI attribute is set to 1. If the prefix is an IPv6 address prefix or a VPN-IPv6 prefix ([RFC4659]), the AFI is set to 2.

If the prefix is an IPv4 address prefix or an IPv6 address prefix, the Subsequent Address Family Identifier (SAFI) field is set to 4. If the prefix is a VPN-IPv4 address prefix or a VPN-IPv6 address prefix, the SAFI is set to 128.

The use of SAFI 4 or SAFI 128 when the AFI is other than 1 or 2 is outside the scope of this document.

This document does not specify the format of the "Network Address of Next Hop" field of the MP\_REACH\_NLRI attribute. The format of the next hop field depends upon a number of factors, and is discussed in a number of other RFCs: see [RFC4364], [RFC4659], [RFC4798], and [RFC5549].

There are a variety of applications that make use of alternative methods of using BGP to advertise MPLS label bindings. See, e.g.,



[RFC7432], [RFC6514], or [TUNNEL-ENCAPS]. The method described in the current document is not claimed to be the only way of using BGP to advertise MPLS label bindings.. Discussion of which method to use for which application is outside the scope of the current document.

In the remainder of this document, we will use the term "SAFI-x UPDATE" to refer to a BGP UPDATE message containing an MP\_REACH\_NLRI attribute or an MP\_UNREACH\_NLRI attribute ([RFC4760] whose SAFI field contains the value x.

This document defines a BGP Optional Capabilities parameter ([RFC5492]) known as the "Multiple Labels Capability".

- o Unless this Capability is sent on a given BGP session by both of that session's BGP speakers, a SAFI-4 or SAFI-128 UPDATE message sent on that session from either speaker MUST bind a prefix to only a single label, and MUST use the encoding of Section 2.2.
- o If this Capability is sent by both BGP speakers on a given session, an UPDATE message on that session, from either speaker, MUST use the encoding of Section 2.3, and MAY bind a prefix to a sequence of more than one label.

The encoding of the Multiple Labels Capability is specified in Section 2.1.

Procedures for explicitly withdrawing a label binding are given in Section 2.4. Procedures for changing the label(s) bound to a given prefix by a given node are given in Section 2.5.

Procedures for propagating SAFI-4 and SAFI-128 UPDATES are discussed in Section 3.

When a BGP speaker installs and propagates a SAFI-4 or SAFI-128 update, and if it changes the value of the Network Address of Next Hop field, it must program its data plane appropriately. This is discussed in Section 4.

## 2.1. Multiple Labels Capability

[RFC5492] defines the "Capabilities Optional Parameter". A BGP speaker can include a Capabilities Optional Parameter in a BGP OPEN message. The Capabilities Optional Parameter is a triple including a one-octet Capability Code, a one-octet Capability length, and a variable-length Capability Value.

This document defines a Capability Code, known as the "Multiple Labels Capability" code. IANA will assign a codepoint for this

Capability Code. (This Capability Code is new to this document and does not appear in [RFC3107].)

If a BGP speaker has not sent the Multiple Labels Capability in its BGP Open message on a particular BGP session, or if it has not received the Multiple Labels Capability in the BGP Open message from its peer on that BGP session, that BGP speaker MUST NOT send on that session any UPDATE message that binds more than one MPLS label to any given prefix. Further, when advertising the binding of a single label to a prefix, the BGP speaker MUST use the encoding specified in Section 2.2.

The value field of the Multiple Labels Capability (shown in Figure 1) consists of one or more triples, where each triple consists of four octets. The first two octets of a triple specify an AFI value, the third octet specifies a SAFI value, and the fourth specifies a Count. If one of the triples is <AFI,SAFI,Count>, the Count is the maximum number of labels that the BGP speaker can process in a received UPDATE of the specified AFI/SAFI.

If the Capability contains more than one triple with a given AFI/SAFI, all but the first MUST be ignored.

Any triple of the form <AFI=x,SAFI=y,Count=0> MUST be ignored.

If the Capability contains the triple <AFI=x,SAFI=y,Count=255>, then no limit has been placed on the number of labels that can be advertised in UPDATES of the corresponding AFI/SAFI.

A Multiple Labels Capability whose length is not a multiple of four MUST be considered to be malformed.

[RFC4724] ("Graceful Restart Mechanism for BGP") describes a procedure that allows routes learned over a given BGP session to be maintained when the session fails and then restarts. This procedure requires the entire RIB to be transmitted when the session restarts. If the Multiple Labels Capability for a given AFI/SAFI had been exchanged on the failed session, but is not exchanged on the restarted session, then any prefixes advertised in that AFI/SAFI with multiple labels MUST be explicitly withdrawn. Similarly, if the maximum label count, specified in the Capability for a given AFI/SAFI is reduced, any prefixes advertised with more labels than are valid for the current session MUST be explicitly withdrawn.

[Enhanced-GR] ("Accelerated Routing Convergence for BGP Graceful Restart") describes another procedure that allows the routes learned over a given BGP session to be maintained when the session fails and

then restarts. These procedures MUST NOT be applied if either of the following conditions hold:

- o The Multiple Labels Capability for a given AFI/SAFI had been exchanged prior to the restart, but has not been exchanged on the restarted session.
- o The Multiple Labels Capability for a given AFI/SAFI had been exchanged with a given Count prior to the restart, but has been exchanged with a smaller count on the restarted session.

If either of these conditions holds, the complete set of routes for of the given AFI/SAFI MUST be exchanged.

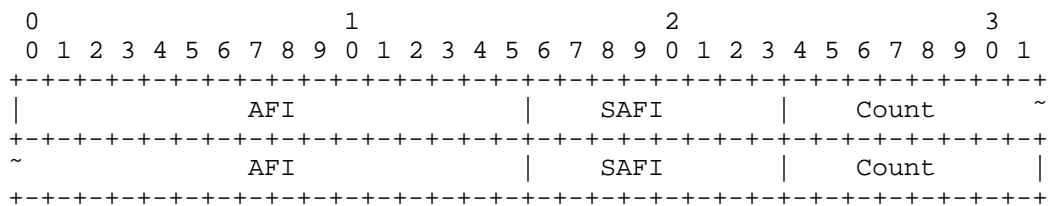


Figure 1: Value Field of Multiple Labels Capability

If a BGP OPEN message contains multiple copies of the Multiple Labels Capability, only the first copy is significant; subsequent copies MUST BE ignored.

If a BGP speaker has sent the Multiple Labels Capability in its BGP OPEN message for a particular BGP session, and has also received the Multiple Labels Capability in its peer's BGP OPEN message for that session, and if both Capabilities specify AFI/SAFI x/y, then:

when using an UPDATE of AFI x and SAFI y to advertise the binding of a label or sequence of labels to a given prefix, the BGP speaker MUST use the encoding of Section 2.3. This encoding MUST be used even if only one label is being bound to a given prefix.

If both BGP speakers of a given BGP session have sent the Multiple Labels Capability, but AFI/SAFI x/y has not been specified in both Capabilities, then UPDATES of AFI/SAFI x/y on that session MUST use the encoding of Section 2.2, and such UPDATES can only bind one label to a prefix.

A BGP speaker SHOULD NOT send an UPDATE that binds more labels to a given prefix than its peer is capable of receiving, as specified in the Multiple Labels Capability sent by that peer. If a BGP speaker receives an UPDATE that binds more labels to a given prefix than the

number of prefixes the BGP speaker is prepared to receive (as announced in its Multiple Labels Capability), the BGP speaker MUST apply the "treat-as-withdraw" strategy of [RFC7606] to that UPDATE.

Notwithstanding the number of labels that a BGP speaker has claimed to be able to receive, its peer MUST NOT attempt to send more labels than can be properly encoded in the NLRI field of the MP\_REACH\_NLRI attribute. Please note that there is only a limited amount of space in the NLRI field for labels:

- o per [RFC4760] the size of this field is limited to 255 bits (not 255 octets), including the number of bits in the prefix;
- o in a SAFI-128 update, the prefix is at least 64 bits long, and may be as long as 192 bits (e.g., in a VPN-IPv6 host route).

## 2.2. NLRI Encoding when the Multiple Labels Capability is Not Used

If the Multiple Labels Capability has not been both sent and received on a given BGP session, then in a BGP UPDATE on that session whose MP\_REACH\_NLRI attribute contains one of the AFI/SAFI combinations specified in Section 2, the NLRI field is encoded as shown in Figure 2:

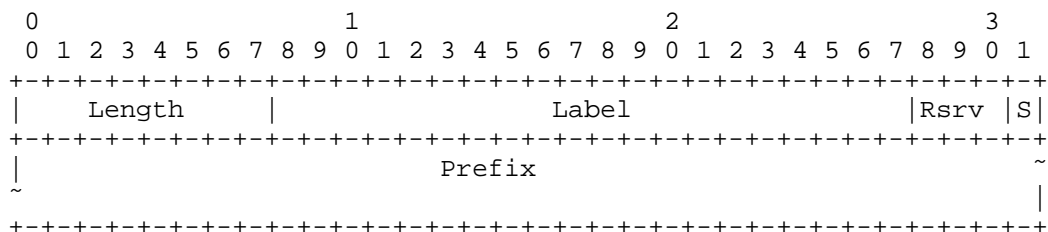


Figure 2: NLRI With One Label

### - Length:

The Length field consists of a single octet. It specifies the length in bits of the remainder of the NLRI field.

Note that the length will always be the sum of 20 (number of bits in label field) plus 3 (number of bits in Rsrv field) plus 1 (number of bits in S field) plus the length in bits of the prefix.

In an MP\_REACH\_NLRI attribute whose AFI/SAFI is 1/4, the prefix length will be 32 bits or less. In an MP\_REACH\_NLRI attribute whose AFI/SAFI is 2/4, the prefix length will be 128 bits or less. In an MP\_REACH\_NLRI attribute whose SAFI is 128, the prefix will

be 96 bits or less if the AFI is 1, and will be 192 bits or less if the AFI is 2.

As specified in [RFC4760], actual length of the NLRI field will be the number of bits specified in the length field, rounded up to the nearest integral number of octets.

- Label:

The Label field is a 20-bit field, containing an MPLS label value (see [RFC3032]).

- Rsrv:

This 3-bit field SHOULD be set to zero on transmission, and MUST be ignored on reception.

- S:

This 1-bit field MUST be set to one on transmission, and MUST be ignored on reception.

Note that the UPDATE message not only advertises the binding between the prefix and the label, it also advertises a path to the prefix via the node identified in the "Network Address of Next Hop" field of the MP\_REACH\_NLRI attribute.

[RFC3107] requires that if only a single label is bound to a prefix, the S bit must be set. If the S bit is not set, [RFC3107] specifies that additional labels will appear in the NLRI. However, some implementations assume that the NLRI will contain only a single label, and do not check the setting of the S bit. The procedures specified in the current document will interwork with such implementations. As long as the Multiple Labels Capability is not sent and received by both BGP speakers on a given BGP session, this document REQUIRES that only one label be specified in the NLRI, that the S bit be set on transmission, and that it be ignored on reception.

If the procedures of [ADD-PATHS] are being used, a four-octet "path identifier" (as defined in Section 3 of [ADD-PATHS]) is part of the NLRI, and precedes the Length field.

### 2.3. NLRI Encoding when the Multiple Labels Capability is Used

If the Multiple Labels Capability has been both sent and received on a given BGP session, then in a BGP UPDATE on that session whose MP\_REACH\_NLRI attribute contains one of the AFI/SAFI combinations

specified in Section 2, the NLRI field is encoded as shown in Figure 3:

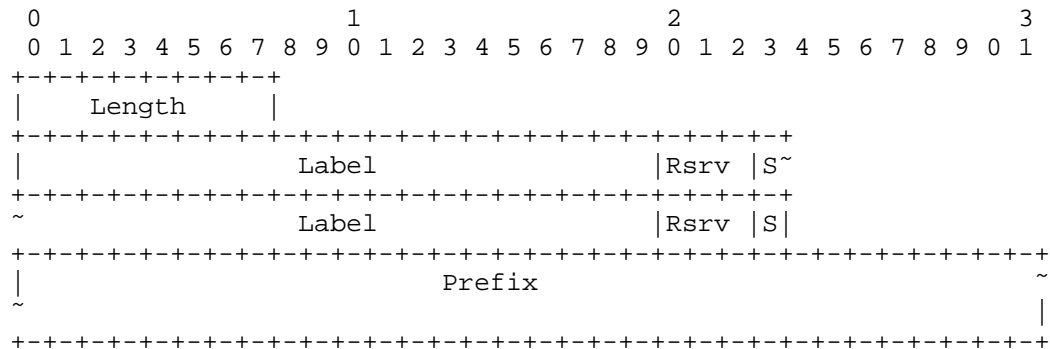


Figure 3: NLRI With Multiple Labels

- Length:

The Length field consists of a single octet. It specifies the length in bits of the remainder of the NLRI field.

Note that for each label, the length is increased by 24 bits (20 bits in label field plus 3 bits in Rsrv field plus 1 S bit).

In an MP\_REACH\_NLRI attribute whose AFI/SAFI is 1/4, the prefix length will be 32 bits or less. In an MP\_REACH\_NLRI attribute whose AFI/SAFI is 2/4, the prefix length will be 128 bits or less. In an MP\_REACH\_NLRI attribute whose SAFI is 128, the prefix will be 96 bits or less if the AFI is 1, and will be 192 bits or less if the AFI is 2.

As specified in [RFC4760], actual length of the NLRI field will be the number of bits specified in the length field, rounded up to the nearest integral number of octets.

- Label:

The Label field is a 20-bit field, containing an MPLS label value ([RFC3032]).

- Rsrv:

This 3-bit field SHOULD be set to zero on transmission, and MUST be ignored on reception.

- S:

In all labels except the last (i.e., in all labels except the one immediately preceding the prefix), the S bit MUST be 0. In the last label, the S bit MUST be 1.

Note that failure to set the S bit in the last label will make it impossible to parse the NLRI correctly. See Section 3 paragraph j of [RFC7606] for a discussion of error handling when the NLRI cannot be parsed.

Note that the UPDATE message not only advertises the binding between the prefix and the labels, it also advertises a path to the prefix via the node identified in the "next hop" field of the MP\_REACH\_NLRI attribute.

If the procedures of [ADD-PATHS] are being used, a four-octet "path identifier" (as defined in Section 3 of [ADD-PATHS]) is part of the NLRI, and precedes the Length field.

#### 2.4. How to Explicitly Withdraw the Binding of a Label to a Prefix

Suppose a BGP speaker has announced, on a given BGP session, the binding of a given label or sequence of labels to a given prefix. Suppose it now wishes to withdraw that binding. To do so, it may send a BGP UPDATE message with an MP\_UNREACH\_NLRI attribute. The NLRI field of this attribute is encoded as follows:

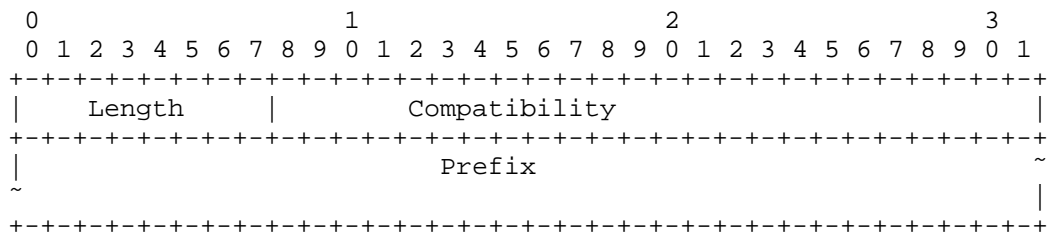


Figure 4: NLRI For Withdrawal

Upon transmission, the Compatibility field SHOULD be set to 0x800000. Upon reception, the value of the Compatibility field MUST be ignored.

This encoding is used for explicitly withdrawing the binding, on a given BGP session, between the specified prefix and whatever label or sequence of labels had previously been bound by the procedures of this document to that prefix on the given session. This encoding is used whether or not the Multiple Labels Capability has been sent or received on the session. Note that label/prefix bindings that were not advertised on the given session cannot be withdrawn by this method.

When using an MP\_UNREACH\_NLRI attribute to withdraw a route whose NLRI was previously specified in an MP\_REACH\_NLRI attribute, the lengths and values of the respective prefixes must match, and the respective AFI/SAFIs must match. If the procedures of [ADD-PATHS] are being used, the respective values of the "path identifier" fields must match as well. Note that the prefix length is not the same as the NLRI length; to determine the prefix length of a prefix in an MP\_UNREACH\_NLRI, the length of the Compatibility field must be subtracted from the length of the NLRI.

An explicit withdrawal in a SAFI-x UPDATE on a given BGP session not only withdraws the binding between the prefix and the label(s), it also withdraws the path to that prefix that was previously advertised in a SAFI-x UPDATE on that session.

[RFC3107] allowed one to specify a particular label value in the Compatibility field. However, the functionality that required the presence of a particular label value (or sequence of label values) was never implemented, and that functionality is not present in the current document. Hence the value of this field is of no significance; there is never any reason for this field to contain a label value or a sequence of label values.

[RFC3107] also allowed one to withdraw a binding without specifying the label explicitly, by setting the Compatibility field to 0x800000. However, some implementations set it to 0x000000. In order to ensure backwards compatibility, this document RECOMMENDS that the Compatibility field be set to 0x800000, but REQUIRES that it be ignored upon reception.

## 2.5. Changing the Label that is Bound to a Prefix

Suppose a BGP speaker, S1, has received on a given BGP session, a SAFI-4 or SAFI-128 UPDATE, U1, that specifies label (or sequence of labels) L1, prefix P, and next hop N1. As specified above, this indicates that label (or sequence of labels) L1 is bound to prefix P at node N1. Suppose that S1 now receives, on the same session, an UPDATE, U2, of the same AFI/SAFI, that specifies label (or sequence of labels) L2, prefix P, and the same next hop, N1.

- o If [ADD-PATHS] is not being used, UPDATE U2 MUST be interpreted as meaning that L2 is now bound to P at N1, and that L1 is no longer bound to P at N1. That is, the UPDATE U1 is implicitly withdrawn, and is replaced by UPDATE U2.
- o Suppose that [ADD-PATHS] is being used, that UPDATE U1 has Path Identifier I1, and that UPDATE U2 has Path Identifier I2.



- \* If I1 is the same as I2, UPDATE U2 MUST be interpreted as meaning that L2 is now bound to P at N1, and that L1 is no longer bound to P at N1. UPDATE U1 is implicitly withdrawn.
- \* If I1 is not the same as I2, U2 MUST be interpreted as meaning that L2 is now bound to P at N1, but MUST NOT be interpreted as meaning that L1 is no longer bound to P at N1. Under certain conditions (specification of which is outside the scope of this document), S1 may choose to load balance traffic between the path represented by U1 and the path represented by U2. To send traffic on the path represented by U1, S1 uses the label(s) advertised in U1; to send traffic on the path represented by U2, S1 uses the label(s) advertised in U2. (Although these two paths have the same next hop, one must suppose that they diverge further downstream.)

Suppose a BGP speaker, S1, has received, on a given BGP session, a SAFI-4 or SAFI-128 UPDATE that specifies label L1, prefix P, and next hop N1. Suppose that S1 now receives, on a different BGP session, an UPDATE, of the same AFI/SAFI, that specifies label L2, prefix P, and the same next hop, N1. BGP speaker S1 SHOULD treat this as indicating that N1 has at least two paths to P, and S MAY use this fact to do load-balancing of any traffic that it has to send to P.

Note that this section discusses only the case where two UPDATES have the same next hop. Procedures for the case where two UPDATES have different next hops are adequately described in [RFC4271].

### 3. Installing and/or Propagating SAFI-4 or SAFI-128 Routes

#### 3.1. Comparability of Routes

Suppose a BGP speaker has received two SAFI-4 UPDATES specifying the same Prefix, and that either:

- o the two UPDATES are received on different BGP sessions, or
- o the two UPDATES are received on the same session, add-paths is used on that session, and the NLRIs of the two updates have different path identifiers.

These two routes MUST be considered to be comparable, even if they specify different labels. Thus the BGP bestpath selection procedures (Section 9.1 of [RFC4271]) are applied to select one of them as the better path. If the procedures of [ADD-PATHS] are not being used on a particular BGP session, only the best path is propagated on that session. If the procedures of [ADD-PATHS] are being used on a

particular BGP session, then both paths may be propagated on that session, though with different path identifiers..

The same applies to SAFI-128 routes.

### 3.2. Modification of Label(s) Field When Propagating

#### 3.2.1. When the Next Hop Field is Unchanged

When a SAFI-4 or SAFI-128 route is propagated, if the "Network Address of Next Hop" field is left unchanged, the label field(s) MUST also be left unchanged.

Note that a given route MUST NOT be propagated to a given peer if the route's NLRI has multiple labels, but the peer can only handle a single label in the NLRI. Similarly, a given route SHOULD NOT be propagated to a given peer if the route's NLRI has more labels than the peer has announced (through its "Multiple Labels" Capability) that it can handle. In either case, if a previous route with the same AFI, SAFI, and prefix (but with fewer labels) has already been propagated to the peer, that route MUST be withdrawn from that peer, using the procedure of Figure 4.

#### 3.2.2. When the Next Hop Field is Changed

If the "Network Address of Next Hop" field is changed before a SAFI-4 or SAFI-128 route is propagated, the label field(s) of the propagated route MUST contain the label(s) that that is (are) bound to the prefix at the new next hop.

Suppose BGP speaker S1 has received an UPDATE that binds a particular sequence of one or more labels to a particular prefix. If S1 chooses to propagate this route after changing its next hop, S1 may change the label in any of the following ways, depending upon local policy:

- o A single label may be replaced by a single label, of the same or different value.
- o A sequence of multiple labels may be replaced by a single label.
- o A single label may be replaced by a sequence of multiple labels.
- o A sequence of multiple labels may be replaced by a sequence of multiple labels; the number of labels may be left the same, or may be changed.

Of course, when deciding whether to propagate, to a given BGP peer, an UPDATE binding a sequence of more than one label, a BGP speaker

must attend to the information provided by the Multiple Labels Capability (see Section 2.1). A BGP speaker MUST NOT send multiple labels to a peer with which it has not exchanged the "Multiple Labels" Capability, and SHOULD NOT send more labels to a given peer than the peer has announced (via the "Multiple Labels" Capability) than it can handle.

It is possible that a BGP speaker's local policy will tell it to encode N labels in a given route's NLRI before propagating the route, but that one of the BGP speaker's peers cannot handle N labels in the NLRI. In this case, the BGP speaker has two choices:

- o It can propagate the route to the given peer with fewer than N labels. However, whether this makes sense, and if so, how to choose the labels, is also a matter of local policy
- o It can decide not to propagate the route to the given peer. In that case, if a previous route with the same AFI, SAFI, and prefix (but with fewer labels) has already been propagated to that peer, that route MUST be withdrawn from that peer, using the procedure of Figure 4.

#### 4. Data Plane

In the following, we will use the phrase "node S tunnels packet P to node N", where packet P is an MPLS packet. By this phrase, we mean that node S encapsulates packet P and causes packet P to be delivered to node N, in such a way that P's label stack before encapsulation will be seen unchanged by N, but will not be seen by the nodes (if any) between S and N.

If the tunnel is a Label Switched Path (LSP), encapsulating the packet may be as simple as pushing on another MPLS label. If node N is a layer 2 adjacency of node S, S a layer 2 encapsulation may be all that is needed. Other sorts of tunnels (e.g., IP tunnels, GRE tunnels, UDP tunnels) may also be used, depending upon the particular deployment scenario.

Suppose BGP speaker S1 receives a SAFI-4 or SAFI-128 BGP UPDATE with an MP\_REACH\_NLRI specifying label L1, prefix P, and next hop N1, and suppose S1 installs this route as its (or one of its) bestpath(s) towards P. And suppose S1 propagates this route after changing the next hop to itself and changing the label to L2. Suppose further that S1 receives an MPLS data packet, and in the process of forwarding that MPLS data packet, S1 sees label L2 rise to the top of the packet's label stack. Then to forward the packet further, S1 must replace L2 with L1 as the top entry in the packet's label stack, and S1 must then tunnel the packet to N1.

Suppose that the route received by S1 specified not a single label, but a sequence of k labels <L11, L12, ..., L1k>, where L11 is the first label appearing in the NLRI, and L1k is the last. And suppose again that S1 propagates this route after changing the next hop to itself and changing the label field to the single label L2. Suppose further that S1 receives an MPLS data packet, and in the process of forwarding that MPLS data packet, S1 sees label L2 rise to the top of the packet's label stack. In this case, instead of simply replacing L2 with L1, S1 removes L2 from the top of the label stack, and then pushes labels L1k through L11 onto the label stack, such that L11 is now at the top of the label stack. Then S1 must tunnel the packet to N1. (Note that L1k will not be at the bottom of the packet's label stack, and hence will not have the "bottom of stack" bit set, unless L2 had previously been at the bottom of the packet's label stack.)

The above paragraph assumes that when S1 propagates a SAFI-4 or SAFI-128 route after setting the next hop to itself, it replaces the label or labels specified in the NLRI of that route with a single label. However, it is also possible, as determined by local policy, for a BGP speaker to specify multiple labels when it propagates a SAFI-4 or SAFI-128 route after setting the next hop to itself.

Suppose, for example, that S1 supports context labels ([RFC5331]). Let L21 be a context label supported by S1, and let L22 be a label that is in the label space identified (at S1) by L21. Suppose S1 receives a SAFI-4 or SAFI-128 UPDATE whose prefix is P, whose label field is <L11,L12,...L1k>, and whose next hop is N1. Before propagating the UPDATE, S1 may set the next hop to itself (by replacing N1 with S1), and may replace the label stack <L11,L12,...L1k> with the pair of labels <L21,L22>.

In this case, if S1 receives an MPLS data packet whose top label is L21 and whose second label is L22, S1 will remove both L21 and L22 from the label stack, and replace them with <L11,L12,...L1k>. Note that the fact that L21 is a context label is known only to S1; other BGP speakers do not know how S1 will interpret L21 (or L22).

The ability to replace one or more labels by one or more labels can provide great flexibility, but must be done carefully. Let's suppose again that S1 receives an UPDATE that specifies prefix P, label stack <L11,L12,...,L1k>, and next hop N1. And suppose that S1 propagates this UPDATE to BGP speaker S2 after setting next hop self and after replacing the label field with <L21,L22,...L2k>. Finally, suppose that S1 programs its data plane so that when it processes a received MPLS packet whose top label is L21, it replaces L21 with <L11,L12,...,L1k>, and then tunnels the packet to N1.

In this case, BGP speaker S2 will have received a route with prefix P, label field <L21,L22,...L2k>, and next hop S1. If S2 decides to forward an IP packet according to this route, it will push <L21,L22,...L2k> onto the packet's label stack, and tunnel the packet to S1. S1 will replace L21 with <L11,L12,...,L1k>, and will tunnel the packet to N1. N1 will receive the packet with the following label stack: <L11,L12,...L1k,L22,...L2k>. While this may be useful in certain scenarios, it may provide unintended results in other scenarios.

Procedures for choosing, setting up, maintaining, or determining the liveness of a particular tunnel or type of tunnel, are outside the scope of this document.

It is a matter of local policy whether SAFI-4 routes can be used as the basis for forwarding IP packets, or whether SAFI-4 routes can only be used for forwarding MPLS packets. If BGP speaker S1 is forwarding IP packets according to SAFI-4 routes, then consider an IP packet with destination address D, such that P is the "longest prefix match" for D from among the routes that are being used to forward IP packets. And suppose the packet is being forwarded according to a SAFI-4 route whose prefix is P, whose next hop is N1, and whose sequence of labels is L1. To forward the packet according to this route, S1 must create a label stack for the packet, push on the sequence of labels L1, and then tunnel the packet to N1.

#### 5. Relationship Between SAFI-4 and SAFI-1 Routes

It is possible that a BGP speaker will receive both a SAFI-1 route for prefix P and a SAFI-4 route for prefix P. The significance of this is a matter of local policy.

For example, some implementations may regard SAFI-1 routes and SAFI-4 routes as completely independent, and may treat them in a "ships in the night" fashion. In this case, bestpath selection for the two SAFIs is independent, and there will be a best SAFI-1 route to P as well as a best SAFI-4 route to P. Which packets get forwarded according to the routes of which SAFI is then a matter of local policy.

Other implementations may treat the SAFI-1 and SAFI-4 routes for a given prefix as comparable, such that the best route to prefix P is either a SAFI-1 route or a SAFI-4 route, but not both. In implementations of this sort, if load balancing is done among a set of equal cost routes, some of the equal cost routes may be SAFI-1 routes and some may be SAFI-4 routes. Whether this is allowed is again a matter of local policy.

Some implementations may allow a single BGP session to carry UPDATES of both SAFI-1 and SAFI-4; other implementations may disallow this.

A BGP speaker may receive a SAFI-4 route over a given BGP session, but may have other BGP sessions for which SAFI-4 is not enabled. In this case, the BGP speaker MAY convert the SAFI-4 route to a SAFI-1 route and then propagate the result over the session on which SAFI-4 is not enabled. Whether this is done is a matter of local policy.

## 6. IANA Considerations

IANA is requested to assign a codepoint for "Multiple Labels Capability" in the "BGP Capability Codes" registry, with this document as the reference.

IANA is requested to modify the "BGP Capability Codes" registry to mark Capability Code 4 ("Multiple routes to a destination") as deprecated, with this document as the reference.

IANA is requested to change the reference for SAFI 4 in the "Subsequent Address Family Identifiers (SAFI) Parameters" to this document. IANA is also requested to add this document as a reference for SAFI 128 in that same registry.

## 7. Security Considerations

The security considerations of the BGP specification ([RFC4271]) apply.

If a BGP implementation, not conformant with the current document, encodes multiple labels in the NLRI but has not sent and received the "Multiple Labels" Capability, a BGP implementation that does conform with the current document will likely reset the BGP session.

This document specifies that certain data packets be "tunneled" from one BGP speaker to another. This requires that the packets be encapsulated while in flight. This document does not specify the encapsulation to be used. However, if a particular encapsulation is used, the security considerations of that encapsulation are applicable.

If a particular tunnel encapsulation does not provide integrity and authentication, it is possible that a data packet's label stack can be modified, through error or malfeasance, while the packet is in flight. This can result in misdelivery of the packet.

There are various techniques one can use to constrain the distribution of BGP UPDATE messages. If a BGP UPDATE advertises the

binding of a particular (set of) label(s) to a particular address prefix, such techniques can be used to control the set of BGP speakers that are intended to learn of that binding. However, if BGP sessions do not provide privacy, other routers may learn of that binding.

When a BGP speaker processes a received MPLS data packet whose top label it advertised, there is no guarantee that the label in question was put on the packet by a router that was intended to know about the label binding. If a BGP speaker is using the procedures of this document, it may be useful for that speaker to distinguish its "internal" interfaces from its "external" interfaces, and to avoid advertising the same labels to BGP speakers reached on internal interfaces as to BGP speakers reached on external interfaces. Then a data packet can be discarded if its top label was not advertised over the type of interface from which the packet was received. This reduces the likelihood of forwarding packets whose labels have been "spoofed" by untrusted sources.

## 8. Acknowledgements

This draft obsoletes RFC 3107. We wish to thank Yakov Rekhter, co-author of RFC 3107, for his work on that document. We also wish to thank Ravi Chandra, Enke Chen, Srihari R. Sangli, Eric Gray, and Liam Casey for their review of and comments on that document.

We thank Alexander Okonnikov and David Lamparter for pointing out a number of the errors in RFC 3107.

We wish to thank Lili Wang and Kaliraj Vairavakkalai for their help and advice during the preparation of this document.

We also thank Bruno Decraene, Jie Dong, Jeff Haas, Jakob Heitz, Keyur Patel, and Kevin Wang for their review of and comments on this document.

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc3032>>.
- [RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <<http://www.rfc-editor.org/info/rfc3107>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<http://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<http://www.rfc-editor.org/info/rfc4364>>.
- [RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", RFC 4659, DOI 10.17487/RFC4659, September 2006, <<http://www.rfc-editor.org/info/rfc4659>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<http://www.rfc-editor.org/info/rfc4760>>.
- [RFC4798] De Clercq, J., Ooms, D., Prevost, S., and F. Le Faucheur, "Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)", RFC 4798, DOI 10.17487/RFC4798, February 2007, <<http://www.rfc-editor.org/info/rfc4798>>.
- [RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <<http://www.rfc-editor.org/info/rfc5492>>.
- [RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<http://www.rfc-editor.org/info/rfc5549>>.



- [RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <<http://www.rfc-editor.org/info/rfc7606>>.

## 9.2. Informative References

- [ADD-PATHS] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", internet-draft draft-ietf-idr-add-paths-15, May 2016.
- [Enhanced-GR] Patel, K., Chen, E., Fernando, R., and J. Scudder, "Accelerated Routing Convergence for BGP Graceful Restart", internet-draft draft-ietf-idr-enhanced-gr-05, December 2014.
- [RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, DOI 10.17487/RFC4724, January 2007, <<http://www.rfc-editor.org/info/rfc4724>>.
- [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, <<http://www.rfc-editor.org/info/rfc5331>>.
- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <<http://www.rfc-editor.org/info/rfc6514>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<http://www.rfc-editor.org/info/rfc7432>>.
- [TUNNEL-ENCAPS] Rosen, E., Patel, K., and G. Vandevelde, "The BGP Tunnel Encapsulation Attribute", internet-draft draft-ietf-idr-tunnel-encaps-01, December 2015.

Author's Address

Eric C. Rosen (editor)  
Juniper Networks, Inc.  
10 Technology Park Drive  
Westford, Massachusetts 01886  
United States

Email: [erosen@juniper.net](mailto:erosen@juniper.net)

MPLS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: November 12, 2016

T. Saad  
K. Raza  
R. Gandhi  
Cisco Systems Inc  
X. Liu  
Ericsson  
V. Beeram  
Juniper Networks  
H. Shah  
Ciena  
I. Bryskin  
X. Chen  
Huawei Technologies  
R. Jones  
Brocade  
B. Wen  
Comcast  
May 11, 2016

A YANG Data Model for MPLS Static LSPs  
draft-saad-mpls-static-yang-03

Abstract

This document contains the specification for the MPLS Static Label Switched Paths (LSPs) YANG model. The model allows for the provisioning of static LSP(s) on LER(s) and LSR(s) devices along a LSP path without the dependency on any signaling protocol. The MPLS Static LSP model augments the MPLS base YANG model with specific data to configure and manage MPLS Static LSP(s).

Status of This Memo

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

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

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

This Internet-Draft will expire on November 12, 2016.

## Copyright Notice

Copyright (c) 2016 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 (<http://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. Terminology . . . . .	3
1.2. MPLS Static LSPs Model Tree Diagram . . . . .	4
1.3. MPLS Static LSP YANG Module . . . . .	5
2. IANA Considerations . . . . .	11
3. Security Considerations . . . . .	11
4. References . . . . .	11
4.1. Normative References . . . . .	11
4.2. Informative References . . . . .	12
Authors' Addresses . . . . .	12

## 1. Introduction

This document describes a YANG data model for configuring and managing the Static LSPs feature. The model allows the configuration of LER and LSR devices with the necessary MPLS cross-connects or bindings to realize an end-to-end LSP service.

A static LSP is established by manually specifying incoming and outgoing MPLS label(s) and necessary forwarding information on each of the traversed Label Edge Router (LER) and Label Switched Router (LSR) devices (ingress, transit, or egress nodes) of the forwarding path.

For example, on an ingress LER device, the model is used to associate a specific Forwarding Equivalence Class (FEC) of packets- e.g. matching a specific IP prefix in a Virtual Routing or Forwarding (VRF) instance- to an MPLS outgoing label imposition, next-hop(s) and respective outgoing interface(s) to forward the packet. On an LSR device, the model is used to create a binding that swaps the incoming label with an outgoing label and forwards the packet on one or

multiple egress path(s). On an egress LER, it is used to create a binding that decapsulates the incoming MPLS label and performs forwarding based on the inner MPLS label (if present) or IP forwarding in the packet.

The MPLS Static LSP YANG model is defined in module "ietf-mpls-static" and augments the MPLS Base YANG model defined in module "ietf-mpls" in [I-D.saad-mpls-base-yang]. The approach described in [I-D.openconfig-netmod-opstate] is adopted to represent data pertaining to configuration intended, applied state and derived state data elements. Each container in the model holds a "config" and "state" sub-container. The "config" sub-container is used to represent the intended configurable parameters, and the state sub-container is used to represent both the applied configurable parameters and any derived state, such as counters or statistical information.

### 1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

The following terms are defined in [RFC6020]:

- o augment,
- o configuration data,
- o data model,
- o data node,
- o feature,
- o mandatory node,
- o module,
- o schema tree,
- o state data,
- o RPC operation.

## 1.2. MPLS Static LSPs Model Tree Diagram

The MPLS Static LSP tree diagram is shown in Figure 1.

```

module: ietf-mpls-static
augment /rt:routing/mpls:mpls:
  +--rw static-lsps
    +--rw static-lsp* [name]
      +--rw name          string
      +--rw config
        +--rw in-segment
          +--rw (type)?
            +--:(ip-prefix)
              | +--rw ip-prefix?          inet:ip-prefix
            +--:(mpls-label)
              | +--rw incoming-label?    mpls:mpls-label
          +--rw operation?                enumeration
          +--rw (out-segment)?
            +--:(simple-path)
              | +--rw next-hop?           inet:ip-address
              | +--rw outgoing-label?     mpls:mpls-label
              | +--rw outgoing-interface? if:interface-ref
            +--:(path-list)
              +--rw paths* [path-index]
                +--rw path-index          uint16
                +--rw backup-path-index?   uint16
                +--rw next-hop?           inet:ip-address
                +--rw outgoing-labels*     mpls:mpls-label
                +--rw outgoing-interface?  if:interface-ref
                +--rw loadshare?           uint16
                +--rw role?                enumeration
        +--ro state
          +--ro in-segment
            +--ro (type)?
              +--:(ip-prefix)
                | +--ro ip-prefix?        inet:ip-prefix
              +--:(mpls-label)
                | +--ro incoming-label?    mpls:mpls-label
            +--ro operation?                enumeration
          +--ro (out-segment)?
            +--:(simple-path)
              | +--ro next-hop?           inet:ip-address
              | +--ro outgoing-label?     mpls:mpls-label
              | +--ro outgoing-interface?  if:interface-ref
            +--:(path-list)
              +--ro paths* [path-index]
                +--ro path-index          uint16
                +--ro backup-path-index?   uint16

```

+-ro next-hop?	inet:ip-address
+-ro outgoing-labels*	mpls:mpls-label
+-ro outgoing-interface?	if:interface-ref
+-ro loadshare?	uint16
+-ro role?	enumeration

Figure 1: MPLS Static LSP tree diagram

### 1.3. MPLS Static LSP YANG Module

<CODE BEGINS>file "ietf-mpls-static@2016-05-11.yang"

```
module ietf-mpls-static {  
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-static";  
    prefix "mpls-static";  
  
    import ietf-mpls {  
        prefix mpls;  
    }  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-inet-types {  
        prefix inet;  
    }  
  
    import ietf-interfaces {  
        prefix "if";  
    }  
  
    organization "IETF MPLS Working Group";  
  
    contact  
        "WG Web:    <http://tools.ietf.org/wg/mpls/>  
        WG List:    <mailto:mpls@ietf.org>  
        WG Chair:   Loa Andersson  
                   <mailto:loa@pi.nu>  
        WG Chair:   Ross Callon  
                   <mailto:rcallon@juniper.net>  
        WG Chair:   George Swallow
```

```
<mailto:swallow.ietf@gmail.com>

Editor:   Tarek Saad
          <mailto:tsaad@cisco.com>

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

Editor:   Rakesh Gandhi
          <mailto:rgandhi@cisco.com>

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

Editor:   Vishnu Pavan Beeram
          <mailto:vbeeram@juniper.net>

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

Editor:   Igor Bryskin
          <mailto:Igor.Bryskin@huawei.com>

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

Editor:   Raqib Jones
          <mailto:raqib@Brocade.com>

Editor:   Bin Wen
          <mailto:Bin_Wen@cable.comcast.com>;
```

description

```
"This YANG module augments the 'ietf-routing' module with basic
configuration and operational state data for MPLS static";
```

```
revision "2016-05-11" {
  description
    "Latest revision:
     - Addressed MPLS-RT review comments";
  reference "RFC 3031: A YANG Data Model for Static MPLS LSPs";
}
```

```
grouping path-basic_config {
  description "common definitions for statics";

  leaf next-hop {
    type inet:ip-address;
```



```
    description "next hop IP address for the LSP";
  }

  leaf outgoing-label {
    type mpls:mpls-label;
    description
      "label value to push at the current hop for the
      LSP";
  }

  leaf outgoing-interface {
    type if:interface-ref;
    description
      "The outgoing interface";
  }
}

grouping path-properties_config {
  description
    "MPLS path properties";
  leaf path-index {
    type uint16;
    description
      "Path identifier";
  }

  leaf backup-path-index {
    type uint32;
    description
      "Backup path identifier";
  }

  leaf next-hop {
    type inet:ip-address;
    description
      "The address of the next-hop";
  }

  leaf-list outgoing-labels {
    type mpls:mpls-label;
    ordered-by user;
    description
      "The outgoing MPLS labels to impose";
  }

  leaf outgoing-interface {
    type if:interface-ref;
  }
}
```

```
    description
      "The outgoing interface";
  }

  leaf loadshare {
    type uint16;
    description
      "This value is used to compute a loadshare to perform un-equal
      load balancing when multiple outgoing path(s) are specified. A
      share is computed as a ratio of this number to the total under
      all configured path(s).";
  }

  leaf role {
    type enumeration {
      enum PRIMARY {
        description
          "Path as primary traffic carrying";
      }
      enum BACKUP {
        description
          "Path acts as backup";
      }
      enum PRIMARY_AND_BACKUP {
        description
          "Path acts as primary and backup simultaneously";
      }
    }
    description
      "The MPLS path role";
  }
}

grouping static-lsp_config {
  description "common definitions for static LSPs";

  container in-segment {
    description
      "MPLS incoming segment";
    choice type {
      description
        "Basic FEC choice";
      case ip-prefix {
        leaf ip-prefix {
          type inet:ip-prefix;
          description "An IP prefix";
        }
      }
    }
  }
}
```

```
    case mpls-label {
      leaf incoming-label {
        type mpls:mpls-label;
        description "label value on the incoming packet";
      }
    }
  }
}

leaf operation {
  type enumeration {
    enum impose-and-forward {
      description
        "Operation impose outgoing label(s) and forward to
        next-hop";
    }
    enum pop-and-forward {
      description
        "Operation pop incoming label and forward to next-hop";
    }
    enum pop-impose-and-forward {
      description
        "Operation pop incoming label, impose one or more
        outgoing label(s) and forward to next-hop";
    }
    enum swap-and-forward {
      description
        "Operation swap incoming label, with outgoing label and
        forward to next-hop";
    }
    enum pop-and-lookup {
      description
        "Operation pop incoming label and perform a lookup";
    }
  }
  description
    "The MPLS operation to be executed on the incoming packet";
}

choice out-segment {
  description "The MPLS out-segment type choice";
  case simple-path {
    uses path-basic_config;
  }
  case path-list {
    list paths {
      key path-index;
      description

```

```

        "The list of MPLS paths associated with the FEC";
        uses path-properties_config;
    }
}
}

grouping static-lsp {
    description "grouping for top level list of static LSPs";
    container config {
        description
            "Holds the intended configuration";
        uses static-lsp_config;
    }
    container state {
        config false;
        description
            "Holds the state and inuse configuration";
        uses static-lsp_config;
    }
}

augment "/rt:routing/mpls:mpls" {
    description "Augmentations for MPLS Static LSPs";
    container static-lsps {
        description
            "Statically configured LSPs, without dynamic signaling";
        list static-lsp {
            key name;
            description "list of defined static LSPs";

            leaf name {
                type string;
                description "name to identify the LSP";
            }
            uses static-lsp;
        }
    }
}
}

<CODE ENDS>

```

Figure 2: MPLS Static LSP YANG module

## 2. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-mpls-static XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-mpls-static namespace: urn:ietf:params:xml:ns:yang:ietf-mpls-static prefix: ietf-mpls-static reference: RFC3031

## 3. Security Considerations

The YANG module defined in this memo 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 NETCONF access control model [RFC6536] provides means 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 data nodes defined in the YANG module which 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.

## 4. References

### 4.1. Normative References

- [I-D.saad-mpls-base-yang]  
Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base and Static LSPs", draft-saad-mpls-base-yang-00 (work in progress), November 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6536>>.

#### 4.2. Informative References

- [I-D.openconfig-netmod-opstate]  
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

#### Authors' Addresses

Tarek Saad  
Cisco Systems Inc

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

Kamran Raza  
Cisco Systems Inc

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

Rakesh Gandhi  
Cisco Systems Inc

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

Xufeng Liu  
Ericsson

Email: xufeng.liu.ietf@gmail.com

Vishnu Pavan Beeram  
Juniper Networks

Email: vbeeram@juniper.net

Himanshu Shah  
Ciena

Email: hshah@ciena.com

Igor Bryskin  
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Xia Chen  
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones  
Brocade

Email: raqib@Brocade.com

Bin Wen  
Comcast

Email: Bin\_Wen@cable.comcast.com

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: May 18, 2018

Yimin Shen  
Minto Jeyananth  
Juniper Networks  
Bruno Decraene  
Orange  
Hannes Gredler  
RtBrick Inc  
Carsten Michel  
Deutsche Telekom  
Huaimo Chen  
Yuanlong Jiang  
Huawei Technologies Co., Ltd.  
November 14, 2017

MPLS Egress Protection Framework  
draft-shen-mpls-egress-protection-framework-07

Abstract

This document specifies a fast reroute framework for protecting IP/MPLS services and MPLS transport tunnels against egress node and egress link failures. In this framework, the penultimate-hop router of an MPLS tunnel acts as the point of local repair (PLR) for egress node failure, and the egress router of the MPLS tunnel acts as the PLR for egress link failure. Each of them pre-establishes a bypass tunnel to a protector. Upon an egress node or link failure, the corresponding PLR performs local failure detection and local repair, by rerouting packets over the corresponding bypass tunnel. The protector in turn performs context label switching or context IP forwarding to send the packets to the ultimate service destination(s). This mechanism can be used to reduce traffic loss before global repair reacts to the failure and control plane protocols converge on the topology changes due to the failure. The framework is applicable to all types of IP/MPLS services and MPLS tunnels. Under the framework, service protocol extensions may be further specified to support service label distribution to the protector.

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 18, 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 . . . . .	3
2. Specification of Requirements . . . . .	5
3. Terminology . . . . .	5
4. Requirements . . . . .	7
5. Egress node protection . . . . .	8
5.1. Reference topology . . . . .	8
5.2. Egress node failure and detection . . . . .	8
5.3. Protector and PLR . . . . .	9
5.4. Protected egress . . . . .	10
5.5. Egress-protected tunnel and service . . . . .	11
5.6. Egress-protection bypass tunnel . . . . .	11
5.7. Context ID, context label, and context based forwarding . . . . .	12
5.8. Advertisement and path resolution for context ID . . . . .	14
5.9. Egress-protection bypass tunnel establishment . . . . .	15
5.10. Local repair on PLR . . . . .	15
5.11. Service label distribution from egress router to protector . . . . .	16
5.12. Centralized protector mode . . . . .	16
6. Egress link protection . . . . .	18
7. Global repair . . . . .	21
8. Example: Layer-3 VPN egress protection . . . . .	21
8.1. Egress node protection . . . . .	23
8.2. Egress link protection . . . . .	24
8.3. Global repair . . . . .	24

9. IANA Considerations . . . . .	24
10. Security Considerations . . . . .	24
11. Acknowledgements . . . . .	25
12. References . . . . .	25
12.1. Normative References . . . . .	25
12.2. Informative References . . . . .	25
Authors' Addresses . . . . .	26

## 1. Introduction

In MPLS networks, label switched paths (LSPs) are widely used as transport tunnels to carry IP and MPLS services across MPLS domains. Examples of MPLS services are layer-2 VPNs, layer-3 VPNs, hierarchical LSPs, and others. In general, a tunnel may carry multiple services of one or multiple types, if the tunnel can satisfy both individual and aggregate requirements (e.g. CoS, QoS) of these services. The egress router of the tunnel should host the corresponding service instances of the services. An MPLS service instance is responsible for forwarding service packets via an egress link to the service destination, based on a service label. An IP service instance is responsible for doing the same based on a service IP address. The egress link is often called a PE-CE (provider edge - customer edge) link or attachment circuit (AC).

Today, local repair based fast reroute mechanisms [RFC4090], [RFC5286], [RFC7490], [RFC7812] have been widely deployed to protect MPLS tunnels against transit link/node failures. They can achieve fast restoration of traffic in the order of tens of milliseconds. Local repair refers to the scenario where the router upstream to an anticipated failure (aka. PLR, i.e. point of local repair) pre-establishes a bypass tunnel to the router downstream of the failure (aka. MP, i.e. merge point), and pre-installs the forwarding state of the bypass tunnel in the data plane. The PLR also uses a rapid mechanism (e.g. link layer OAM, BFD, and others) to locally detect the failure in the data plane. When the failure occurs, the PLR reroutes traffic through the bypass tunnel to the MP, allowing the traffic to continue to flow to the tunnel's egress router.

This document describes a fast reroute framework for egress node and egress link protection. Similar to transit link/node protection, this framework relies on a PLR to perform local failure detection and local repair. In egress node protection, the PLR is the penultimate-hop router of a tunnel. In egress link protection, the PLR is the egress router of the tunnel. The framework relies on a so-called "protector" to serve as the tailend of a bypass tunnel. The protector is a router that hosts "protection service instances" and has its own connectivity or paths to service destinations. When a PLR is doing local repair, the protector is responsible for

performing "context label switching" for rerouted MPLS service packets and "context IP forwarding" for rerouted IP service packets. Thus, the service packets can continue to reach service destinations with minimum disruption.

This framework considers an egress node failure as a failure of a tunnel, as well as a failure of all the services carried by the tunnel, because service packets can no longer reach the service instances on the egress router. Therefore, the framework addresses egress node protection at both tunnel level and service level simultaneously. Likewise, the framework considers an egress link failure as a failure of all the services traversing the link, and addresses egress link protection at the service level.

This framework requires that the destination (a CE or site) of a service MUST be dual-homed or have dual paths to an MPLS network, normally via two MPLS edge routers. One of them is the egress router of the service's transport tunnel, and the other is a backup egress router which hosts "backup service instances". In the "co-located" protector mode in this document, the backup egress router serves as a protector, and hence each backup service instance acts as a protection instance. In the "centralized" protector mode (Section 5.12), a protector and a backup egress router are decoupled, and each protection service instance and its corresponding backup service instance are hosted on separate routers.

The framework is described by mainly referring to P2P (point-to-point) tunnels. However, it is equally applicable to P2MP (point-to-multipoint), MP2P (multipoint-to-point) and MP2MP (multipoint-to-multipoint) tunnels, when a sub-LSP can be viewed as a P2P tunnel.

The framework is a multi-service and multi-transport framework. It assumes a generic model where each service is comprised of a common set of components, including a service instance, a service label, and a service label distribution protocol, and the service is transported over an MPLS tunnel of any type. The framework also assumes service labels to be downstream assigned, i.e. assigned by egress routers. Therefore, the framework is generally applicable to most existing and future services. Services which use upstream-assigned service labels are out of scope of this document and left for further study.

The framework does not require extensions for the existing signaling and label distribution protocols (e.g. RSVP, LDP, BGP, etc.) of MPLS tunnels. It expects transport tunnels and bypass tunnels to be established by using the generic mechanisms provided by the protocols. On the other hand, it does not preclude future extensions to the protocols which may facilitate the procedures. One example of such extension is [RSVP-EP]. The framework may need extensions for

IGPs and service label distribution protocols, to support protection establishment and context label switching. This document provides guidelines for these extensions, but the specific details SHOULD be addressed in separate documents.

The framework is intended to complement control-plane convergence and global repair, which are traditionally used to recover networks from egress node and egress link failures. Control-plane convergence relies on control protocols to react on the topology changes due to a failure. Global repair relies on an ingress router to remotely detect a failure and switch traffic to an alternative path. An example of global repair is the BGP Prefix Independent Convergence mechanism [BGP-PIC] for BGP established services. Compared with these mechanisms, this framework is considered as faster in traffic restoration, due to the nature of local failure detection and local repair. However, it is RECOMMENDED that the framework SHOULD be used in conjunction with control-plane convergence or global repair, in order to take the advantages of both approaches to achieve more effective protection. That is, the framework provides fast and temporary repair, and control-plane convergence or global repair provides ultimate and permanent repair.

## 2. Specification of Requirements

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

## 3. Terminology

Egress router - A router at the egress endpoint of a tunnel. It hosts service instances for all the services carried by the tunnel, and has connectivity with the destinations of the services.

Egress node failure - A failure of an egress router.

Egress link failure - A failure of the egress link (e.g. PE-CE link, attachment circuit) of a service.

Egress failure - An egress node failure or an egress link failure.

Egress-protected tunnel - A tunnel whose egress router is protected by a mechanism according to this framework. The egress router is hence called a protected egress router.

Egress-protected service - An IP or MPLS service which is carried by an egress-protected tunnel, and hence protected by a mechanism according to this framework.

Backup egress router - Given an egress-protected tunnel and its egress router, this is another router which has connectivity with all or a subset of the destinations of the egress-protected services carried by the egress-protected tunnel.

Backup service instance - A service instance which is hosted by a backup egress router, and corresponding to an egress-protected service on a protected egress router.

Protector - A role acted by a router as an alternate of a protected egress router, to handle service packets in the event of an egress failure. A protector may be physically co-located with or decoupled from a backup egress router, depending on the co-located or centralized protector mode.

Protection service instance - A service instance hosted by a protector, corresponding to the service instance of an egress-protected service on a protected egress router. A protection service instance is a backup service instance, if the protector is co-located with a backup egress router.

PLR - A router at the point of local repair. In egress node protection, it is the penultimate-hop router on an egress-protected tunnel. In egress link protection, it is the egress router of the egress-protected tunnel.

Protected egress {E, P} - A virtual node consisting of an ordered pair of egress router E and protector P. It serves as the virtual destination of an egress-protected tunnel, and as the virtual location of the egress-protected services carried by the tunnel.

Context identifier (ID) - A globally unique IP address assigned to a protected egress {E, P}.

Context label - A non-reserved label assigned to a context ID by a protector.

Egress-protection bypass tunnel - A tunnel used to reroute service packets around an egress failure.

Co-located protector mode - The scenario where a protector and a backup egress router are co-located as one router, and hence each backup service instance serves as a protection service instance.

Centralized protector mode - The scenario where a protector is a dedicated router, and is decoupled from backup egress routers.

Context label switching - Label switching performed by a protector, in the label space of an egress router indicated by a context label.

Context IP forwarding - IP forwarding performed by a protector, in the IP address space of an egress router indicated by a context label.

#### 4. Requirements

This document considers the followings as the design requirements of this egress protection framework.

- o The framework must support P2P tunnels. It should equally support P2MP, MP2P and MP2MP tunnels, by treating each sub-LSP as a P2P tunnel.
- o The framework must support multi-service and multi-transport networks. It must accommodate existing and future signaling and label-distribution protocols of tunnels and bypass tunnels, including RSVP, LDP, BGP, IGP, segment routing, and others. It must also accommodate existing and future IP/MPLS services, including layer-2 VPNs, layer-3 VPNs, hierarchical LSP, and others. It must provide a generic solution for environments where different types of services and tunnels may co-exist.
- o The framework must consider minimizing disruption during deployment. It should only involve routers close to egress, and be transparent to ingress routers and other transit routers.
- o In egress node protection, for scalability and performance reasons, a PLR must be agnostic to services and service labels, like PLRs in transit link/node protection. It must maintain bypass tunnels and bypass forwarding state on a per-transport-tunnel basis, rather than per-service-destination or per-service-label basis. It should also support bypass tunnel sharing between transport tunnels.
- o A PLR must be able to use its local visibility or information of routing and/or TE topology to compute or resolve a path for a bypass tunnel to a protector.
- o A protector must be able to perform context label switching for rerouted MPLS service packets, based on service label(s) assigned by an egress router. It must be able to perform context IP forwarding for rerouted IP service packets, in the public or private IP address space used by an egress router.

- o The framework must be able to work seamlessly with transit link/node protection mechanisms to achieve end-to-end coverage.
- o The framework must be able to work in conjunction with global repair and control plane convergence.

## 5. Egress node protection

### 5.1. Reference topology

This document refers to the following topology when describing the procedures of egress node protection.

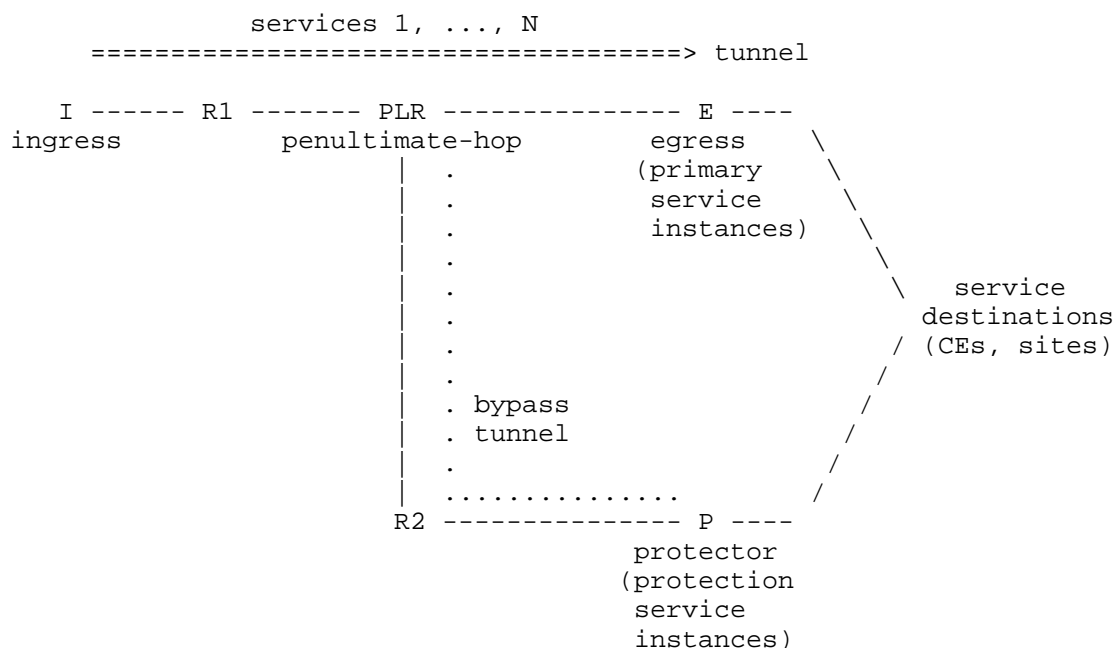


Figure 1

### 5.2. Egress node failure and detection

An egress node failure refers to the failure of an MPLS tunnel's egress router. At the service level, it also means a service instance failure for each IP/MPLS service carried by the tunnel.

Ideally, an egress node failure can be detected by an adjacent router (i.e. PLR in this framework) using a node liveness detection

mechanism, or based on a collective failure of all the links to that node. However, the assumption is that the mechanisms SHOULD be reasonably fast, i.e. faster than control plane failure detection and remote failure detection. Otherwise, local repair will not be able to provide much benefit compared to control plane convergence or global repair. In general, the speed, accuracy, and reliability of a mechanism are the key factors to decide its applicability in egress node protection. This document provides the following guidelines in this regard.

- o If the PLR has a reasonably fast mechanism to detect and differentiate a link failure (of the link between the PLR and the egress node) and an egress node failure, it SHOULD set up both link protection and egress node protection, and trigger one and only one protection upon a corresponding failure.
- o If the PLR has a fast mechanism to detect a link failure and an egress node failure, but cannot distinguish them; Or, if the PLR has a fast mechanism to detect a link failure only, but not an egress node failure, the PLR has two options:
  1. It MAY set up link protection only, and leave the egress node failure to global repair and control plane convergence to handle.
  2. It MAY set up egress node protection only, and treat a link failure as a trigger for the egress node protection. However, the assumption is that treating a link failure as an egress node failure MUST NOT have a negative impact on services. Otherwise, it SHOULD adopt the previous option.

### 5.3. Protector and PLR

A router is assigned to the "protector" role to protect a tunnel and the services carried by the tunnel against an egress node failure. The protector is responsible for hosting a protection service instance for each protected service, serving as the tailend of a bypass tunnel, and performing context label switching and/or context IP forwarding for rerouted service packets.

A tunnel can be protected by only one protector at a given time. Multiple tunnels to a given egress router may be protected by a common protector or different protectors. A protector may protect multiple tunnels with a common egress router or different egress routers.

For each tunnel, its penultimate-hop router acts as a PLR. The PLR pre-establishes a bypass tunnel to the protector, and pre-installs



bypass forwarding state in the data plane. Upon detection of an egress node failure, the PLR reroutes all the service packets received on the tunnel through the bypass tunnel to the protector. For MPLS service packets, the PLR keeps service labels intact in the packets. The protector in turn forwards the rerouted service packets towards the ultimate service destinations. Specifically, it performs context label switching for MPLS service packets, based on service labels assigned by the protected egress router; It performs context IP forwarding for IP service packets, based on their destination addresses.

The protector MUST have its own connectivity with each service destination, via a direct link or a multi-hop path, which MUST NOT traverse the protected egress router or be affected by the egress node failure. This also requires that each service destination MUST be dual-homed or have dual paths to the egress router and a backup egress router which serves as the protector. Each protection service instance on the protector relies on such connectivity to set up forwarding state for context label switching and/or context IP forwarding.

#### 5.4. Protected egress

This document introduces the notion of "protected egress" as a virtual node consisting of the egress router E of a tunnel and a protector P. It is denoted by an ordered pair of {E, P}, indicating the primary-and-protector relationship between the two routers. It serves as the virtual destination of the tunnel, and the virtual location of service instances for the services carried by the tunnel. The tunnel and services are considered as being "associated" with the protected egress {E, P}.

A given egress router E may be the tailend of multiple tunnels. In general, the tunnels may be protected by multiple protectors, e.g. P1, P2, and so on, with each Pi protecting a subset of the tunnels. Thus, these routers form multiple protected egresses, i.e. {E, P1}, {E, P2}, and so on. Each tunnel is associated with one and only one protected egress {E, Pi}. All the services carried by the tunnel are then automatically associated with the same protected egress {E, Pi}. Conversely, a service associated with a protected egress {E, Pi} MUST be carried by a tunnel associated with the protected egress {E, Pi}. This mapping MUST be ensured by the ingress router of the tunnel and the service (Section 5.5).

Two routers X and Y may be protectors for each other. In this case, they form two distinct protected egresses {X, Y} and {Y, X}.

### 5.5. Egress-protected tunnel and service

A tunnel, which is associated with a protected egress {E, P}, is called an egress-protected tunnel. It is associated with one and only one protected egress {E, P}. Multiple egress-protected tunnels may be associated with a given protected egress {E, P}. In this case, they share the common egress router and protector, but may or may not share a common ingress router, or a common PLR (i.e. penultimate-hop router).

An egress-protected tunnel is considered as logically "destined" for its protected egress {E, P}. However, its path MUST be resolved and established with E as the physical tailend.

A service, which is associated with a protected egress {E, P}, is called an egress-protected service. The egress router E hosts the primary instance of the service, and the protector P hosts the protection instance of the service.

An egress-protected service is associated with one and only one protected egress {E, P}. Multiple egress-protected services may be associated with a given protected egress {E, P}. In this case, these services share the common egress router and protector, but may or may not share a common egress-protected tunnel or a common ingress router.

An egress-protected service MUST be mapped to an egress-protected tunnel by its ingress router, based on the common protected egress {E, P} of the service and the tunnel. This is achieved by introducing the notion of "context ID" for protected egress {E, P}, as described in (Section 5.7).

### 5.6. Egress-protection bypass tunnel

An egress-protected tunnel destined for a protected egress {E, P} MUST have a bypass tunnel from its PLR to the protector P. This bypass tunnel is called an egress-protection bypass tunnel. The bypass tunnel is considered as logically "destined" for the protected egress {E, P}. However, due to its bypass nature, it MUST be resolved and established with P as the physical tailend and E as the node to avoid. The bypass tunnel MUST have the property that it MUST NOT be affected by any topology change caused by an egress node failure.

An egress-protection bypass tunnel is associated with one and only one protected egress {E, P}. A PLR may share an egress-protection bypass tunnel for multiple egress-protected tunnels associated with a common protected egress {E, P}. For multiple egress-protected tunnels associated with a common protected egress {E, P}, there may be one or

multiple egress-protection bypass tunnels from one or multiple PLRs to the protector P, depending on the paths of the egress-protected tunnels.

#### 5.7. Context ID, context label, and context based forwarding

In this framework, a globally unique IPv4/v6 address is assigned to a protected egress {E, P} to serve as the identifier of the protected egress {E, P}. It is called a "context ID" due to its specific usage in context label switching and context IP forwarding on the protector. It is an IP address that is logically owned by both the egress router and the protector. For the egress node, it indicates the protector. For the protector, it indicates the egress router, particularly the egress router's forwarding context. For other routers in the network, it is an address reachable via both the egress router and the protector in the routing domain and the TE domain (Section 5.8), similar to an anycast address.

The main purpose of a context ID is to coordinate ingress router, egress router, PLR and protector in setting up egress protection. Given an egress-protected service associated with a protected egress {E, P}, its context ID is used as below:

- o If the service is an MPLS service, when E distributes a service label binding message to the ingress router, E attaches the context ID to the message. If the service is an IP service, when E advertises the service destination address to the ingress router, E also attaches the context ID to the advertisement message. How the context ID is encoded in the messages is a choice of the service protocol, and may need protocol extensions to define a "context ID" object.
- o The ingress router uses the context ID as destination to establish or resolve an egress-protected tunnel. The ingress router then maps the service to the tunnel for transportation. In this process, the special semantics of the context ID is transparent to the ingress router. The ingress router only treats the context ID as an IP address of E, and behaves in the same manner as in establishing or resolving a regular transport tunnel, although the end result is an egress-protected tunnel.
- o The context ID is conveyed to the PLR by the signaling protocol of the egress-protected tunnel, or learned by the PLR via an IGP (i.e. OSPF or ISIS) or a topology-driven label distribution protocol (e.g. LDP). The PLR uses the context ID as destination to establish or resolve an egress-protection bypass tunnel to P while avoiding E.

- o P maintains a dedicated label space or a dedicated IP address space for E, depending on whether the service is MPLS or IP. This is referred to as "E's label space" or "E's IP address space", respectively. P uses the context ID to identify the space.
- o If the service is an MPLS service, E also distributes the service label binding message to P. This is the same label binding message that E advertises to the ingress router, attached with the context ID. Based on the context ID, P installs the service label in an MPLS forwarding table corresponding to E's label space. If the service is an IP service, P installs an IP route in an IP forwarding table corresponding to E's IP address space. In either case, the protection service instance on P interprets the service and constructs forwarding state for the route based on P's own connectivity to the service's destination.
- o P assigns a non-reserved label to the context ID. In the data plane, this label represents the context ID and indicates E's label space and IP address space. Therefore, it is called a "context label".
- o The PLR may establish the egress-protection bypass tunnel to P in several manners. If the bypass tunnel is established by RSVP, the PLR signals the bypass tunnel with the context ID as destination, and P binds the context label to the bypass tunnel. If the bypass tunnel is established by LDP, P advertises the context label for the context ID as an IP prefix FEC. If the bypass tunnel is established by the PLR in a hierarchical manner, the PLR treats the context label as a one-hop LSP over a regular bypass tunnel to P (e.g. a bypass tunnel to P's loopback IP address). If the bypass tunnel is constructed by using segment routing, the bypass tunnel is represented by a stack of SID labels with the context label as the inner-most SID label (Section 5.9). In any case, the bypass tunnel is a UHP tunnel whose incoming label at P is the context label.
- o During local repair, all the service packets received by P on the bypass tunnel have the context label as top label. P first pops the context label. For an MPLS service packet, P further looks up the service label in E's label space indicated by the context label, which is called context label switching. For an IP service packet, P looks up the IP destination address in E's IP address space indicated by the context label, which is called context IP forwarding.

## 5.8. Advertisement and path resolution for context ID

Path resolution and computation for a context ID are done on ingress routers for egress-protected tunnels, and on PLRs for egress-protection bypass tunnels. Therefore, given a protected egress {E, P} and its context ID, E and P MUST coordinate the context ID in the routing domain and the TE domain via IGP advertisement. The context ID MUST be advertised in such a manner that all egress-protected tunnels MUST have E as tailend, and all egress-protection bypass tunnels MUST have P as tailend while avoiding E.

This document suggests two approaches:

1. The first approach is called "proxy mode". It requires E and P, but not the PLR, to have the knowledge of the egress protection schema. E and P advertise the context ID as a virtual proxy node (i.e. a logical node) connected to the two routers, with the link between the proxy node and E having more preferable IGP and TE metrics than the link between the proxy node and P. Therefore, all egress-protected tunnels destined for the context ID should automatically follow the shortest IGP or TE paths to E. Each PLR will no longer view itself as a penultimate-hop, but rather two hops away from the proxy node, via E. The PLR will be able to find a bypass path via P to the proxy node, while the bypass tunnel should actually be terminated by P.
2. The second approach is called "alias mode". It requires P and the PLR, but not E, to have the knowledge of the egress protection schema. E simply advertises the context ID as a regular IP address. P advertises the context ID and the context label by using a "context ID label binding" advertisement. The advertisement MUST be understood by the PLR. In both routing domain and TE domain, the context ID is only reachable via E. This ensures that all egress-protected tunnels destined for the context ID should have E as tailend. Based on the "context ID label binding" advertisement, the PLR can establish an egress-protection bypass tunnel in several manners (Section 5.9). The "context ID label binding" advertisement is defined as IGP mirroring context segment in [SR-ARCH], [SR-OSPF] and [SR-ISIS]. These IGP extensions are generic in nature, and hence can be used for egress protection purposes.

In a scenario where an egress-protected tunnel is an inter-area or inter-AS tunnel, its associated context ID MUST be propagated from the residing area/AS to the other areas/AS' via IGP or BGP, so that the ingress router of the tunnel can have the reachability to the context ID. The propagation process of the context ID SHOULD be the same as that of a regular IP address in an inter-area/AS environment.

### 5.9. Egress-protection bypass tunnel establishment

A PLR MUST know the context ID of a protected egress {E, P} in order to establish an egress-protection bypass tunnel. The information is obtained from the signaling or label distribution protocol of the egress-protected tunnel. The PLR may or may not need to have the knowledge of the egress protection schema. All it does is to set up a bypass tunnel to a context ID while avoiding the next-hop router (i.e. egress router). This is achievable by using a constraint-based computation algorithm similar to those which are commonly used in the computation of traffic engineering paths and loop-free alternate (LFA) paths. Since the context ID is advertised in the routing domain and the TE domain by IGP according to Section 5.8, the PLR should be able to resolve or establish such a bypass path with the protector as tailend. In some cases like the proxy mode, the PLR may do so in the same manner as transit node protection.

An egress-protection bypass tunnel may be established via several methods:

- (1) It may be established by a signaling protocol (e.g. RSVP), with the context ID as destination. The protector binds the context label to the bypass tunnel.
- (2) It may be formed by a topology driven protocol (e.g. LDP with various LFA mechanisms). The protector advertises the context ID as an IP prefix FEC, with the context label bound to it.
- (3) It may be constructed as a hierarchical tunnel. When the protector uses the alias mode (Section 5.8), the PLR will have the knowledge of the context ID, context label, and protector (i.e. the advertiser). The PLR can then establish the bypass tunnel in a hierarchical manner, with the context label as a one-hop LSP over a regular bypass tunnel to the protector's IP address (e.g. loopback address). This regular bypass tunnel may be established by RSVP, LDP, segment routing, and others.

### 5.10. Local repair on PLR

In this framework, a PLR is agnostic to services and service labels. This obviates the need to maintain bypass forwarding state on a per-service basis, and allows bypass tunnel sharing between egress-protected tunnels. The PLR may share an egress-protection bypass tunnel for multiple egress-protected tunnels associated with a common protected egress {E, P}. During local repair, the PLR reroutes all service packets received on the egress-protected tunnels via the egress-protection bypass tunnel. Service labels remain intact in MPLS service packets.

Label operation during the rerouting depends on the bypass tunnel's characteristics. If the bypass tunnel is a single level tunnel, the rerouting will involve swapping the incoming label of an egress-protected tunnel to the outgoing label of the bypass tunnel. If the bypass tunnel is a hierarchical tunnel, the rerouting will involve swapping the incoming label of an egress-protected tunnel to a context label, and pushing the outgoing label of a regular bypass tunnel. If the bypass tunnel is constructed by segment routing, the rerouting will involve swapping the incoming label of an egress-protected tunnel to a context label, and pushing a stack of SID labels of the bypass tunnel.

#### 5.11. Service label distribution from egress router to protector

As mentioned in previous sections, when a protector receives a rerouted MPLS service packet, it performs context label switching based on the packet's service label which is assigned by the corresponding egress router. In order to achieve this, the protector MUST maintain such kind of service labels in dedicated label spaces on a per protected egress {E, P} basis, i.e. one label space for each egress router that it protects.

Also, there MUST be a service label distribution protocol session between each egress router and the protector. Through this protocol, the protector learns the label binding of each egress-protected service. This is the same label binding that the egress router advertises to the corresponding ingress router, attached with a context ID. The corresponding protection service instance on the protector recognizes the service, and resolves forwarding state based on its own connectivity with the service's destination. It then installs the service label with the forwarding state in the label space of the egress router, which is indicated by the context ID (i.e. context label).

Different service protocols may use different mechanisms for such kind of label distribution. Specific protocol extensions may be needed on a per-protocol basis or per-service-type basis. The details of the extensions SHOULD be specified in separate documents. As an example, RFC 8104 specifies the LDP extensions for pseudowire services.

#### 5.12. Centralized protector mode

In this framework, it is assumed that the service destination of an egress-protected service MUST be dual-homed to two edge routers of an MPLS network. One of them is the protected egress router, and the other is a backup egress router. So far in this document, the discussion has been focusing on the scenario where a protector and a

Topologically, a centralized protector may be decoupled from all backup egress routers, or it may be co-located with one backup egress router while decoupled from the other backup egress routers. The procedures in this section assume the scenario where a protector and a backup egress router are decoupled.



them to the service destination. Specifically, in the case of an MPLS service, the protector MUST swap the service label in each received service packet to the label of the backup service advertised by the backup egress router, and then push the label (or label stack) of the transport tunnel.

In order for a centralized protector to map an egress-protected MPLS service to a service hosted on a backup egress router, there MUST be a service label distribution protocol session between the backup egress router and the protector. Through this session, the backup egress router advertises the service label of the backup service, attached with the FEC of the egress-protected service and the context ID of the protected egress {E, P}. Based on this information, the protector associates the egress-protected service with the backup service, resolves or establishes a transport tunnel to the backup egress router, and accordingly sets up forwarding state for the label of the egress-protected service in the label space of the egress router.

The service label which the backup egress router advertises to the protector can be the same as the label which the backup egress router advertises to the ingress router(s), if and only if the forwarding state of the label does not direct service packets towards the protected egress router. Otherwise, the label is not usable for egress protection, because it will create a loop, which MUST be avoided. In this case, the backup egress router MUST advertise a unique service label for egress protection, and set its forwarding state to use the backup egress router's connectivity with the service destination.

## 6. Egress link protection

Egress link protection is achievable through procedures similar to that of egress node protection. In normal situations, an egress router forwards service packets to a service destination based on a service label, whose forwarding state points to an egress link. In egress link protection, the egress router acts as PLR, by performing local failure detection and local repair. Specifically, the egress router pre-establishes an egress-protection bypass tunnel to a protector, and installs bypass forwarding state for the service label, pointing to the bypass tunnel. During local repair, the egress router reroutes service packets via the bypass tunnel to the protector. The protector in turn forwards the packets to the service destination (in the co-located protector mode, as shown in Figure-3), or forwards the packets to a backup egress router (in the centralized protector mode, as shown in Figure-4).

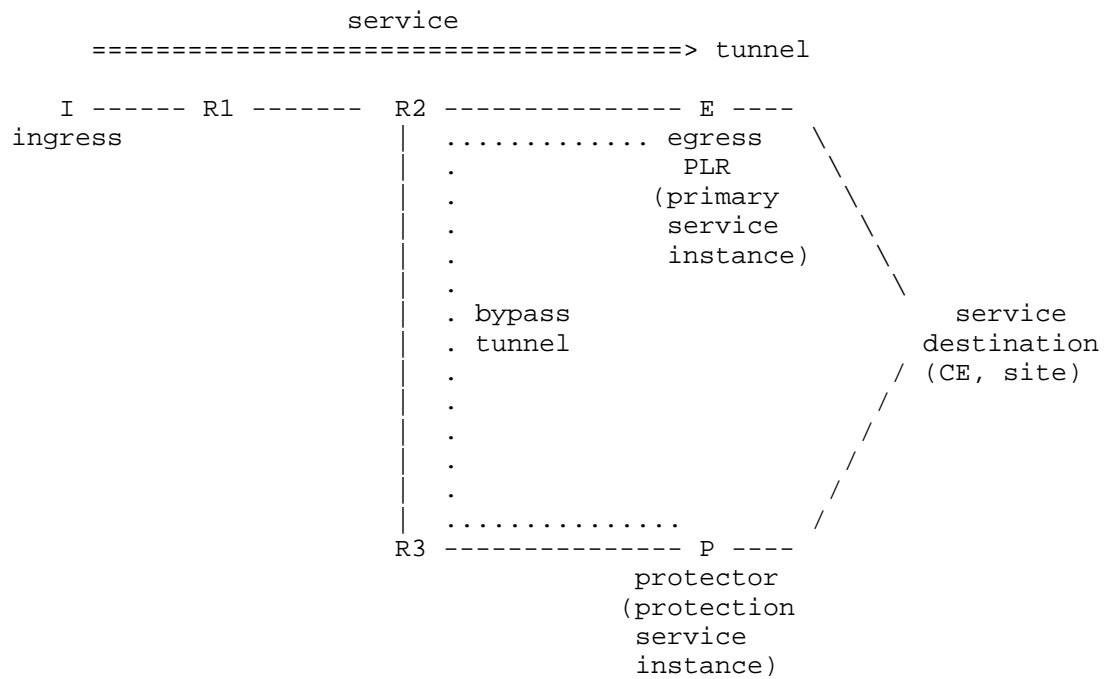


Figure 3

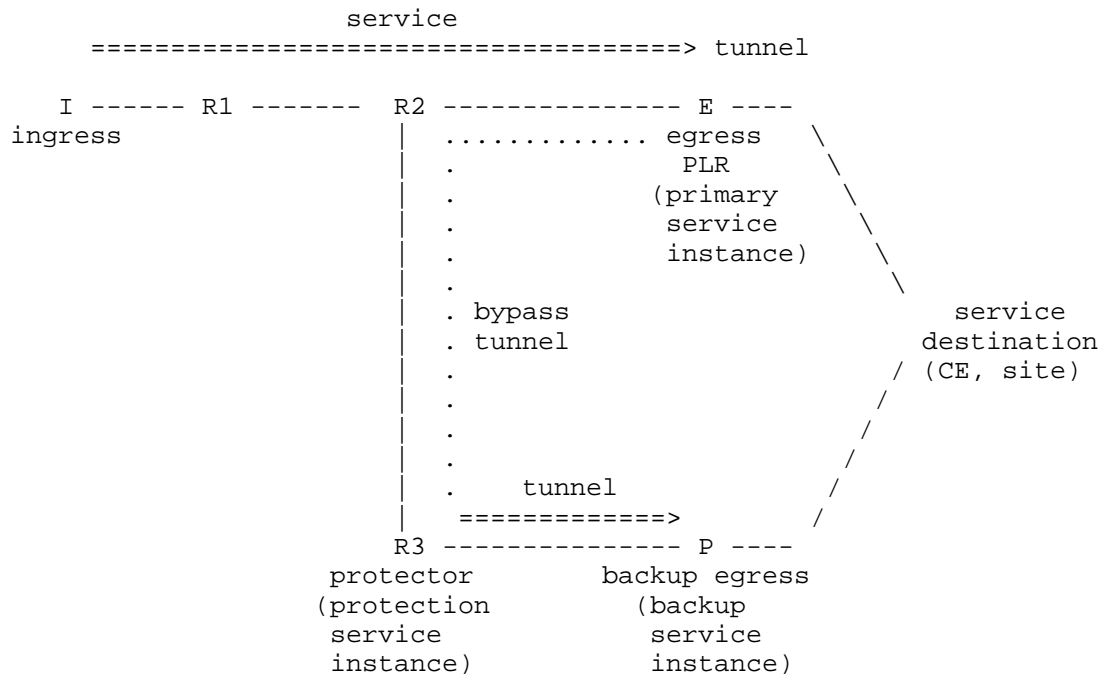


Figure 4

There are two approaches to set up the bypass forwarding state on the egress router, depending on whether the egress router knows the service label advertised by the backup egress router. The difference is that one approach requires the protector to perform context label switching, and the other one does not. Both approaches are equally supported by this framework, and may be used in parallel.

(1) The first approach applies when the egress router does not know the service label advertised by the backup egress router. In this case, the egress router sets up the bypass forwarding state as a label push with the outgoing label of the egress-protection bypass tunnel. Rerouted packets will have the egress router's service label intact. Therefore, the protector MUST perform context label switching, and the bypass tunnel MUST be destined for the context ID of the {E, P} and established as described in Section 5.9. This approach is consistent with egress node protection. Hence, a protector can serve in egress node and egress link protection in a consistent manner, and both the co-located protector mode and the centralized protector mode may be used (Figure-3 and Figure-4).

(2) The second approach applies when the egress router knows the service label advertised by the backup egress route, via a label distribution protocol session. In this case, the backup egress router serves as the protector for egress link protection, regardless of the protector of egress node protection, which should be the same router in the co-located protector mode but may be a different router in the centralized protector mode. The egress router sets up the bypass forwarding state as a label swap from the incoming service label to the service label of the protector, followed by a push with the outgoing label (or label stack) of the egress link protection bypass tunnel. The bypass tunnel is a regular tunnel destined for an IP address of the protector, instead of the context ID of the {E, P}. The protector simply forwards rerouted service packets based on its own service label, rather than performing context label switching. With this approach, only the co-located protector mode is applicable.

Note that for a bidirectional service, the physical link of an egress link may carry service traffic bi-directionally. Therefore, an egress link failure may simultaneously be an ingress link failure for the traffic in the opposite direction. However, protection for ingress link failure SHOULD be provided by a separate mechanism, and hence is out of the scope of this document.

#### 7. Global repair

This framework provides a fast but temporary repair for egress node and egress link failures. For permanent repair, it is RECOMMENDED that the traffic SHOULD be moved to an alternative tunnel or alternative services which are fully functional. This is referred to as global repair. Possible triggers of global repair include control plane notifications of tunnel and service status, end-to-end OAM and fault detection at tunnel or service levels, and others. The alternative tunnel and services may be pre-established as standby, or dynamically established as a result of the triggers or network protocol convergence.

#### 8. Example: Layer-3 VPN egress protection

This section shows an example of egress protection for a layer-3 VPN.

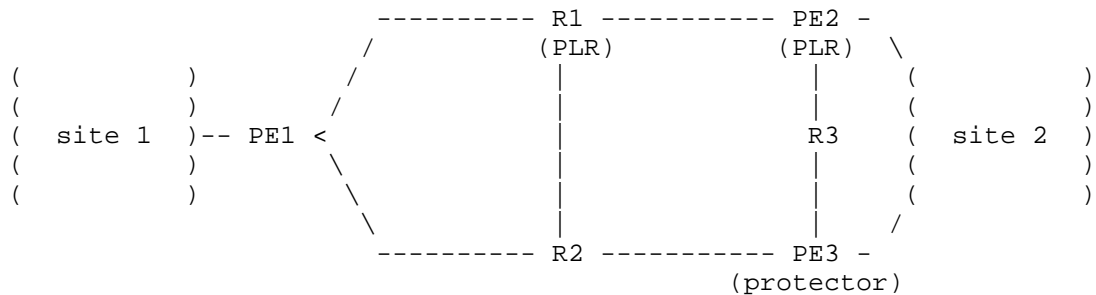


Figure 5

In this example, the site 1 (subnet 203.0.113.192/26) of a given VPN is attached to PE1, and site 2 (subnet 203.0.113.128/26) is dual-homed to PE2 and PE3. PE2 is the primary PE for site 2, and PE3 is the backup PE. Each PE hosts a VPN instance. R1 and R2 are transit routers in the MPLS network. The network uses OSPF as routing protocol, and RSVP-TE as tunnel signaling protocol. The PEs use BGP to exchange VPN prefixes and VPN labels between each other.

Using the framework in this document, the network assigns PE3 to be a protector for PE2 to protect the VPN traffic in the direction from site 1 to site 2. This is the co-located protector mode. Hence, PE2 and PE3 form a protected egress {PE2, PE3}. A context ID 198.51.100.1 is assigned to the protected egress {PE2, PE3}. The VPN instance on PE3 serves as a protection instance for the VPN instance on PE2. On PE3, a context label 100 is assigned to the context ID, and a label table pe2.mpls is created to represent PE2's label space. PE3 installs the label 100 in its default MPLS forwarding table, with nexthop pointing to the label table pe2.mpls. PE2 and PE3 are coordinated to use the proxy mode to advertise the context ID in the routing domain and the TE domain.

PE2 uses per-VRF VPN label allocation mode. It assigns a single label 9000 to the VRF of the VPN. For a given VPN prefix 203.0.113.128/26 in site 2, PE2 advertises it along with the label 9000 and other attributes to PE1 and PE3 via BGP. In particular, the NEXT\_HOP attribute is set to the context ID 198.51.100.1.

Similarly, PE3 also uses per-VRF VPN label allocation mode. It assigns a single label 10000 to the VRF of the VPN. For the VPN prefix 203.0.113.128/26 in site 2, PE3 advertises it along with the label 10000 and other attributes to PE1 and PE2 via BGP. In particular, the NEXT\_HOP attribute is set to an IP address of PE3.

Upon receipt and acceptance of the BGP advertisement, PE1 uses the context ID 198.51.100.1 as destination to compute a TE path for an egress-protected tunnel. The resulted path is PE1->R1->PE2. PE1 then uses RSVP to signal the tunnel, with the context ID 198.51.100.1 as destination, and with the "node protection desired" flag set in the SESSION\_ATTRIBUTE of RSVP Path message. Once the tunnel comes up, PE1 maps the VPN prefix 203.0.113.128/26 to the tunnel and installs a route for the prefix in the corresponding VRF. The route's nexthop is a push with the VPN label 9000, followed by a push with the outgoing label of the egress-protected tunnel.

Upon receipt of the above BGP advertisement from PE2, PE3 (i.e. the protector) recognizes the context ID 198.51.100.1 in the NEXT\_HOP attribute, and installs a route for label 9000 in the label table pe2.mpls. PE3 sets the route's nexthop to a "protection VRF". This protection VRF contains IP routes corresponding to the IP prefixes in the dual-homed site 2, including 203.0.113.128/26. The nexthops of these routes MUST be based on PE3's connectivity with site 2, even if this connectivity is not the best path in PE3's VRF due to metrics (e.g. MED, local preference, etc.), and MUST NOT use any path traversing PE2. Note that the protection VRF is a logical concept, and it may simply be PE3's own VRF if the VRF satisfies the requirement.

#### 8.1. Egress node protection

R1, i.e. the penultimate-hop router of the egress-protected tunnel, serves as the PLR for egress node protection. Based on the "node protection desired" flag and the destination address (i.e. context ID 198.51.100.1) of the tunnel, R1 computes a bypass path to 198.51.100.1 while avoiding PE2. The resulted bypass path is R1->R2->PE3. R1 then signals the path (i.e. egress-protection bypass tunnel), with 198.51.100.1 as destination.

Upon receipt of an RSVP Path message of the egress-protection bypass tunnel, PE3 recognizes the context ID 198.51.100.1 as the destination, and hence responds with the context label 100 in an RSVP Resv message.

After the egress-protection bypass tunnel comes up, R1 installs a bypass nexthop for the egress-protected tunnel. The bypass nexthop is a swap from the incoming label of the egress-protected tunnel to the outgoing label of the egress-protection bypass tunnel.

When R1 detects a failure of PE2, it will invoke the above bypass nexthop to reroute VPN service packets. The packets will have the label of the bypass tunnel as outer label, and the VPN label 9000 as inner label. When the packets arrive at PE3, they will have the

context label 100 as outer label, and the VPN label 9000 as inner label. The context label will first be popped, and then the VPN label will be looked up in the label table pe2.mpls. The lookup will cause the VPN label to be popped, and the IP packets will finally be forwarded to site 2 based on the protection VRF.

## 8.2. Egress link protection

PE2 serves as the PLR for egress link protection. It has already learned the VPN label 10000 from PE3, and hence it uses the approach (2) described in Section 6 to set up bypass forwarding state. It signals an egress-protection bypass tunnel to PE3, by using the path PE2->R3->PE3, and PE3's IP address as destination. After the bypass tunnel comes up, PE2 installs a bypass nexthop for the VPN label 9000. The bypass nexthop is a label swap from the incoming label 9000 to the VPN label 10000 of PE3, followed by a label push with the outgoing label of the bypass tunnel.

When PE3 detects a failure of the egress link, it will invoke the above bypass nexthop to reroute VPN service packets. The packets will have the label of the bypass tunnel as outer label, and the VPN label 10000 as inner label. When the packets arrive at PE3, the VPN label 10000 will be popped, and the IP packets will be forwarded based on the VRF indicated by on the VPN label 10000.

## 8.3. Global repair

Eventually, global repair will take effect, as control plane protocols converge on the new topology. PE1 will choose PE3 as new entrance to site 2. Before that happens, the VPN traffic has been protected by the above local repair.

## 9. IANA Considerations

This document has no request for new IANA allocation.

## 10. Security Considerations

The framework in this document relies on fast reroute around a network failure. Specifically, service traffic is temporarily rerouted from a PLR to a protector. In the centralized protector mode, the traffic is further rerouted from the protector to a backup egress router. Such kind of fast reroute is planned and anticipated, and hence it should not be viewed as a new security threat.

The framework requires a service label distribution protocol to run between an egress router and a protector. The available security

measures of the protocol MAY be used to achieve a secured session between the two routers.

## 11. Acknowledgements

This document leverages work done by Yakov Rekhter, Kevin Wang and Zhaohui Zhang on MPLS egress protection. Thanks to Alexander Vainshtein, Rolf Winter, and Lizhong Jin for their valuable comments that helped shape this document and improve its clarity.

## 12. References

### 12.1. Normative References

- [SR-ARCH] Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing (work in progress), 2017.
- [SR-OSPF] 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 (work in progress), 2017.
- [SR-ISIS] Previdi, S., 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 (work in progress), 2017.

### 12.2. Informative References

- [RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, DOI 10.17487/RFC4090, May 2005, <<https://www.rfc-editor.org/info/rfc4090>>.
- [RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <<https://www.rfc-editor.org/info/rfc5286>>.
- [RFC7490] Bryant, S., Filsfils, C., Previdi, S., Shand, M., and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)", RFC 7490, DOI 10.17487/RFC7490, April 2015, <<https://www.rfc-editor.org/info/rfc7490>>.



- [RFC7812] Atlas, A., Bowers, C., and G. Enyedi, "An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)", RFC 7812, DOI 10.17487/RFC7812, June 2016, <<https://www.rfc-editor.org/info/rfc7812>>.
- [RFC8104] Shen, Y., Aggarwal, R., Henderickx, W., and Y. Jiang, "Pseudowire (PW) Endpoint Fast Failure Protection", RFC 8104, DOI 10.17487/RFC8104, March 2017, <<https://www.rfc-editor.org/info/rfc8104>>.
- [BGP-PIC] Bashandy, P., Filsfils, C., and P. Mohapatra, "BGP Prefix Independent Convergence", draft-ietf-rtgwg-bgp-pic-05.txt (work in progress), 2017.
- [RSVP-EP] Chen, H., Liu, A., Saad, T., Xu, F., Huang, L., and N. So, "Extensions to RSVP-TE for LSP Egress Local Protection", draft-ietf-teas-rsvp-egress-protection (work in progress), 2017.

## Authors' Addresses

Yimin Shen  
Juniper Networks  
10 Technology Park Drive  
Westford, MA 01886  
USA

Phone: +1 9785890722  
Email: [yshen@juniper.net](mailto:yshen@juniper.net)

Minto Jeyananth  
Juniper Networks  
1133 Innovation Way  
Sunnyvale, CA 94089  
USA

Phone: +1 4089367563  
Email: [minto@juniper.net](mailto:minto@juniper.net)

Bruno Decraene  
Orange

Email: [bruno.decraene@orange.com](mailto:bruno.decraene@orange.com)

Hannes Gredler  
RtBrick Inc

Email: hannes@rtbrick.com

Carsten Michel  
Deutsche Telekom

Email: c.michel@telekom.de

Huaimo Chen  
Huawei Technologies Co., Ltd.

Email: huaimo.chen@huawei.com

Yuanlong Jiang  
Huawei Technologies Co., Ltd.  
Bantian, Longgang district  
Shenzhen 518129  
China

Email: jiangyuanlong@huawei.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 12, 2017

P. Turaga  
R. Raszuk  
Bloomberg LP  
September 8, 2016

MPLS Test Labels  
draft-turaga-mpls-test-labels-01

Abstract

This document describes an underlying mechanism for automatic diagnostics of link quality between any two devices connected together by standard point to point link.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 12, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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. Terminology . . . . .	2
2. Introduction . . . . .	2
3. Requirements . . . . .	3
4. MPLS Special Purpose Loop Label . . . . .	4
4.1. Operation of MPLS Special Purpose Loop Label . . . . .	6
4.2. Comparison with stated test requirements . . . . .	8
4.3. Probe size and rate calculation . . . . .	8
5. MPLS Special Purpose Swap-to-Drop and Drop Labels . . . . .	9
6. Probe's QOS marking . . . . .	9
7. Bandwidth Considerations for link under test . . . . .	10
8. I2RS and YANG modelling . . . . .	10
9. IANA Considerations . . . . .	10
10. Security Considerations . . . . .	10
11. Contributors . . . . .	10
12. Acknowledgments . . . . .	10
13. References . . . . .	11
13.1. Normative References . . . . .	11
13.2. Informative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Terminology

- o RTT - Round Trip Time
- o TTL - Time to Live
- o BFD - BiDirectional Failure Detection
- o LFM - Link Fault Management
- o ICMP - Internet Control Message Protocol

## 2. Introduction

Real time monitoring of WAN or MAN link quality presents a real operational challenge. The common use of circuit emulation techniques by carriers makes detection of the circuits degradation difficult. Very often such reduced link quality results in increased queuing times or packet drops beyond SLA guarantees. Furthermore, the characteristics of link degradation is different from link to link.

The problem space described above is further complicated due to the following reasons:

- o Link anomalies may not occur at the same uniform rate or be of the same constant and continuous pattern. This transient characteristic maybe a function of load or other temporary problems for example transport network over-subscription.
- o Encountered degraded service behavior may not translate to link errors or packet discards on either end of the suspected link because the emulated link consisting of multiple independent L2 segments in the carrier's network.

Currently available tools on the circuit endpoints (usually routers) do not allow easy way to diagnose circuit health. Tools used today to detect link issues include:

- o Creating hardware or software loops manually - this results in the actual link under test to be taken out of service. Test traffic is then sent through the link and based on the results of the test, link quality issues are detected.
- o Regular pings/probes on directly connected links between routers/network devices - Depending on the size of the probe packets and the rate at which they are sent between the network devices and the loss, the link issues are detected. The issue with this approach is that network processor on the router has to process all these packets. This causes an additional processing load on the routers.
- o BFD, IP protocol hellos etc are based on detecting neighbor state based on tiny and lightweight hellos. Such probes were designed for fast detection of end-to-end link state events .. not to evaluate link quality. If say N hellos send in T interval are lost it is an indication about link or peer down event.
- o The layer 2 OAM tools are not capable of addressing the requirements since by definition an emulated link consists of number of different L2 links hidden by the emulation layer and its encapsulation. L2 OAM could only indicate potential problems within single layer 2 link. They are light weight and some of these issues can only be detected at various levels of data rates (within agreed SLAs) transiting via such links.

### 3. Requirements

The following are some of the key considerations required to be addressed in an alternative diagnostics solutions:

- o The testing should be atomic in nature - the UUT in this document is a single p2p link.
- o The test should not be subject to any alterations by externally injected packets
- o The probe packets should never be able to transit L3 node to any other L3 node

- o The level of diagnostics data should be configurable such that operator is able to inject anywhere from 0.1% to 100% of test load of a given max link capacity with build in automatic consideration of existing average of production traffic load (unless link is considered as taken out-of-service).
- o The duration of the test traffic should be either configurable by the operator or controlled by built-in detection heuristics.
- o The frequency of the test traffic should be either configurable by the operator or controlled by built-in detection heuristics.
- o Probes should not be subject to process switching by the route processors on either end of the link during the burst.
- o The solution should strive to minimize amount of required protocol extensions for as easy as possible inter-operability characteristics.
- o In the topologies where Link Aggregation is used, the aggregated bandwidth of the link should be considered instead of the individual links. The probe accounting should be recorded as total of all link members. Probe's hashing should follow normal data plane load balancing rules as configured on the directly connected peering routers.

#### 4. MPLS Special Purpose Loop Label

The mechanism for the set of proposed requirements can be constructed by combining two standards based protocol elements: TTL field processing and mpls label lookup.

Special purpose mpls label will allow to setup a scoped link based loop and TTL field can be used to limit the loop duration.

The MPLS local loop label can be either special purpose MPLS label (value 4-6 or 8-12) or Extended special purpose MPLS label (values 16-239). The use of the extended special purpose label would inherently increase the label stack to two for the probe packets with top most label to be of value 15 (extension label per [RFC7274]).

By injecting N number of such probe packets (with max payload up to given MTU value) it is possible to control test load. The observation of the difference between number of injected probes and number of MPLS TTL expiration for a given test bundle would be equal to the number of packet drops observed. Similarly by calculating the time difference of time stamp from the moment of injection of the probe packet to its TTL expiration a good average of real RTT of a link can be calculated. The observation of such RTT times across number of test sequences will allow to model the aggregated queuing delays possibly allowing prediction of upcoming drops.

It needs to be also observed that the above tests are completely orthogonal and would co-exist with current mechanism like LSP Ping [RFC4379] or MPLS BFD [RFC5884]. The proposed in this document procedures do not intend to verify control plane or control plane to data plane correctness.

As stated already TTL handling or label lookup are both standards based and do not require any changes to the underlying hardware. However choice of MPLS special purpose label is proposed to simplify the test operation and remove significant new data plane requirements. The semantics of such label would be following:

When packet is received with MPLS Local Loop Label TTL must be decremented and if  $> 0$  the entire packet must be returned over the same interface over which it was received. If after decrementing TTL the resulting register value is 0 the probe packet should be dropped and local TTL-equal-zero error should be generated. If the probes header contain additional information (for example as described in LSP ping RFC) such header should be copied into the error message punted to the local control plane CPU for further processing.

If implementation allows such local error could be optionally logged and handled in data plane only reporting the aggregated results to the local RE/RP.

Such solutions has following advantages over possible alternative which would be use of regular IP packets:

- o The new mechanism with new type of MPLS label allows for defining a special handling which will not overlap with any of the possible interference with existing protocols ... for example ICMP traceroute
- o The new non transitive specification of mpls special purpose label allows for much stricter security and safer operation of the proposed testing model
- o The separate new label TTL expiration may be easily handled differently then general TTL expiration thus resulting in no data plane rate limiting or pacing
- o Use of signalling less special purpose MPLS LLL relaxes the solution from either additional control plane extensions or requirements to either extend opex with new static routes or to introduce new extensions to already established link local addresses.
- o The main principle of the tests are to be congruent with switching vectors of production traffic. Therefore it is important that probe packets use the same lookup LFIB structure as regular user data packets as far as egress and ingress line cards are concerned.

The MPLS LLL label is to be auto assigned to FECs by either automated association with numbered IP addresses for the given link or with link local addresses. The resolution to MAC address of L2 rewrites (with 8847 ethertype) would be resolved locally through corresponding L3 adjacency addresses.

#### 4.1. Operation of MPLS Special Purpose Loop Label

The following is considered as a high level description of proposed solution:

- o Two routers R1 and R2 connected together by link L1
- o The RTT between R1 and R2 on link L1 is 5ms
- o R1 and R2 have IP connectivity with each other on 10.10.10.0/30 numbered link. R1 has been configured with IP an address of 10.10.10.1 and R2 has been configured with an IP address of 10.10.10.2
- o A MPLS local loop label is allocated to match the corresponding FECs of the link addresses (or link local when applicable).

The following IPv4 probe packet has been injected from R1 towards the FEC of R2:

- o Source IP address: 10.10.10.1
- o Destination IP address: 10.10.10.100
- o TTL = 254
- o payload optional ... (to be discussed by WG)
- o Proposed format:
- o R1 sends probe of the following format:

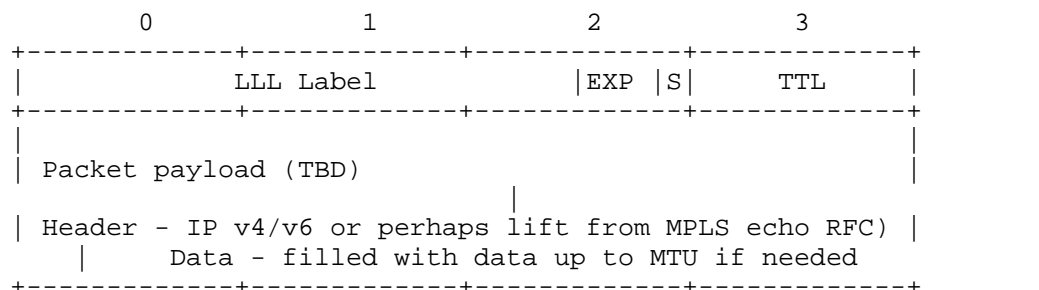


Figure 1: MPLS Label Stack Object

Label: 20 bits  
 Exp: Experimental Use, 3 bits  
 S: Bottom of Stack, 1 bit  
 TTL: Time to Live, 8 bits

Figure 1: MPLS LLL probe format



Such test packet would be resolved and encapsulated into MPLS LLL label and directed towards R2 with even TTL value.

Test sequence:

- o Packet arrives at R2 and TTL is decremented following MPLS LLL label lookup and reinjection towards R1
- o Packet keeps looping till TTL expires on R1.
- o Upon TTL expiration an ICMP TTL-exceeded error message is being logged on R1 (originator of probe sequence). The ICMP message contains the header information of the original packet is being sent to local control plane processor.
- o The local implementation may optionally optimize the accounting for the received vs missed in flight probe packets in the data plane layer and only report the aggregated sequence history
- o The analysis of the packets header would be logged in local or remote database and become a very valuable source of the link's behavioral metrics.

Observations:

- o A test probe packet has potential to be amplified up to 254 times
- o An ICMP TTL exceeded message is indicative of successful test - healthy link
- o No ICMP TTL message implies that the original packet was lost while it was getting looped between two routers. So, No ICMP TTL message means that test for a specific probe has failed. Let's note that this would have no bearing on the local control plane of either end of the test. Any reaction associated and triggered by the test results would be driven by controller (residing together or separate from participating routers).
- o Ability to send multiple packets of different sizes on the link with inherently controlled TTL loop can result in expected burst of control/probe traffic on the link under test
- o Such probe burst can be programmed to get to a certain % of the link speed for a short time

Based on fine tuned testing scenario allowing to fill the bandwidth up to a certain % of link capacity the count of packets originally sent by router R1 should be the same as the number of ICMP TTL expired messages. If the count of packets originally sent by router is the same as the number of ICMP TTL expired messages then the test is successful. If however the number of ICMP TTL expired messages is less than the count of packets originally sent by the router then the test is unsuccessful proving potential problems with the link.

A test probe packet with even initial TTL value will generate a TTL time expired ICMP message on the originating router. A test probe

packet with odd initial TTL value will generate a TTL time expired ICMP message on the neighboring router. It is RECOMMENDED that the test probe is sent with even initial TTL value. So, ICMP messages are not traversing the link under test.

It is RECOMMENDED that a special payload structure is built for these test probes with sequence numbers. When the TTL expires and an ICMP message is generated, the IP header + 64 bits from original packet gets copied to ICMP message. [ RFC 792 ]. This can be used for associating the ICMP message and the test.

#### 4.2. Comparison with stated test requirements

Analysis of the proposed solution against the actual new test methodology requirements:

- o Provides means to potentially fill up the part of link bandwidth very rapidly because of inherent amplification due to high initial TTL value. The fill level of the test traffic is a function of: Initial packet size (higher the packet size the higher the fill level), Initial TTL value (higher the TTL value, higher the multiplicative factor for packets and hence higher the fill level), Initial number of packets sent (the more the packets sent the more the fill level).
- o Test can be run together with production traffic. There is no impact on production traffic neither there is any requirement to stop production traffic in order to perform the test.
- o The amplification of the packets and looping happens as a part of inherent forwarding in the routers. This solution does not require a special process in software or hardware to send the test probes between the two routers.
- o The link is not required to be taken offline for testing. This testing can co-exist with production traffic.
- o This mechanism is light-weight and does not require a lot of protocol programming or significant enhancements.

#### 4.3. Probe size and rate calculation

Initial packet size and rate are important to determine the test fill level for the link. The test packet loops the same number of times as the original TTL value of the original packet. The time it takes for the original packet to come back to the original router is the RTT (Round Trip Time) value between two routers.

Under the assumptions that: RTT of link under test is 1ms, link speed 1 Gb/s, packet size of test packet is 1536 bytes, TTL on original packet is 254, then the test packets would loop on the link for next 254ms.

Under the above assumptions it is easy to calculate that in order fill the 1 Gb/s link to 100% 81 such probe packets need to be injected into any link under test. Likewise in order to fill the link to 20% of its capacity 16 probe packets are required.

Link Simmering - To be able to set non user impacting graceful link removal from IGP topology and conduct full bandwidth test then return the link to the topology.

## 5. MPLS Special Purpose Swap-to-Drop and Drop Labels

While tests described in the former sections are addressing most of the link health monitoring cases there is additional class of tests which may require quite different data plane pattern characteristics. Specifically there is requirement for injection of large number of full MTU echo probes which will only be able to be received by peer's ingress line card and returned once to the sender without even processing the TTL field.

For such purpose this document also defines two additional types of Special Purpose MPLS labels: Swap-to-Drop and Drop Labels.

The semantics of Swap-to-Drop label requires the network device which received packets with Swap-to-Drop label to decrement TTL and swap the label with Drop label and return it over the same interface over which it arrived.

The semantics of Drop Label means to drop the packet received with such special purpose label while incrementing either global interface drop counters or defining the new counters dedicated to logging the received packets with drop label.

Similar to MPLS Special purpose loop label the swap-to-drop as well as drop labels are local to the link. Packets containing such labels should be never switched via a network node.

## 6. Probe's QOS marking

Since injected test packets are regular IP packets encapsulated in MPLS they can be marked with any class of service inserted into EXP field. As a result the test probes similar to actual data will be processed based on the real QoS configuration and will be subject to treatment defined for a given packet class.

That allows both prioritization as well as de-prioritization of a given set of test probes.

## 7. Bandwidth Considerations for link under test

The payload of the test packets can be of any IP protocols.

The link fill levels is also a function of Inter-packet gap of the test and the RTT of that link. Deterministic fill levels can only be derived by accounting for RTT of the link under test.

## 8. I2RS and YANG modelling

It is expected that link testing methodology described in this document will be accessible by I2RS channel as well as YANG models will be defined for both setting and retrieval of the data.

## 9. IANA Considerations

This document requires IANA to define new Special purpose mpls label or extended special purpose MPLS label values subject to WG recommendation.

The following special purpose labels are to be allocated:

- o Local Loop Label
- o Swap-to-Drop Label
- o Drop Label

## 10. Security Considerations

While the proposed mechanism does not define any new protocols nor protocol extensions of already existing specifications it does relay on the TTL-expiry notifications.

Such notifications must be enabled and must not be limited in any way for the specific class of probe packets.

It is highly recommended that test destinations addresses to be not routeable beyond their locally attached devices.

## 11. Contributors

Authors would like to thank Truman Boyes and Leo Pang for their valuable input.

## 12. Acknowledgments

## 13. References

### 13.1. Normative References

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

### 13.2. Informative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<http://www.rfc-editor.org/info/rfc792>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <<http://www.rfc-editor.org/info/rfc3927>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <<http://www.rfc-editor.org/info/rfc4379>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<http://www.rfc-editor.org/info/rfc5884>>.
- [RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <<http://www.rfc-editor.org/info/rfc7274>>.

### Authors' Addresses

Partha Turaga  
Bloomberg LP  
731 Lexington Ave  
New York City, NY 10022  
USA

Email: [pturaga@bloomberg.net](mailto:pturaga@bloomberg.net)

Robert Raszuk  
Bloomberg LP  
731 Lexington Ave  
New York City, NY 10022  
USA

Email: robert@raszuk.net

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](mailto: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