Internet Engineering Task Force (IETF)
Request for Comments: 9070
Category: Standards Track
ISSN: 2070-1721

K. Raza, Ed.
R. Asati
Cisco Systems
X. Liu
IBM Corporation
S. Easale
Juniper Networks
X. Chen
Huawei Technologies
H. Shah
Ciena Corporation
March 2022

YANG Data Model for MPLS LDP

Abstract

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

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

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9070.

Table of Contents

1. Introduction

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

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

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

* Configuration

* Operational State

* Executables (Actions)

* Notifications

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

1.1. Base and Extended

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

* Base

* Extended

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

The examples of a base feature include the configuration of LDP lsr-id, enabling LDP interfaces, setting passwords for LDP sessions, etc., whereas the examples of an extended feature include inbound/outbound label policies, IGP Sync [RFC5443], downstream on demand, etc. It is worth highlighting that LDP IPv6 [RFC7552] is also categorized as an extended feature.

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

2. Specification of Requirements

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

3. Overview

This document defines two new modules for LDP YANG support:

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

"ietf-mpls-ldp-extended"
: A module that specifies the extended LDP features and augments the base LDP module.

It is to be noted that the mLDP YANG data model [MPLS-MLDP-YANG] augments LDP base and extended modules to specify the mLDP-specific base and extended features.

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

* Read-write parameters for configuration (Section 5)

* Read-only parameters for operational state (Section 6)

* Notifications for events (Section 7)

* RPCs for executing commands to perform some action (Section 8)

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

The following diagram depicts high-level LDP YANG tree organization and hierarchy:

```
         +-- rw routing
            +-- rw control-plane-protocols
```

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


    rpcs:
       +-- x mpls-ldp-some_action
       +-- x . . . . .

    notifications:
       +--- n mpls-ldp-some_event
       +--- n ...
```

Figure 1: LDP YANG Tree Organization

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

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

* Multi-topology LDP [RFC7307] is beyond the scope of this document.

* This model does not cover any applications running on top of LDP, nor does it cover any Operations, Administration, and Maintenance (OAM) procedures for LDP.

* This model is a VRF-centric model. It is important to note that [RFC4364] defines VPN Routing and Forwarding (VRF) tables and default forwarding tables as different; however, from a YANG modeling perspective, this introduces unnecessary complications; hence, we are treating the default forwarding table as just another VRF.

* A "network-instance", as defined in [RFC8529], refers to a VRF instance (both default and non-default) within the scope of this model.

* This model supports two address families, namely, "ipv4" and "ipv6".

* This model assumes platform-wide label space (i.e., label space Id of zero). However, when upstream label assignment [RFC6389] is in use, an upstream assigned label is looked up in a Context-Specific Label Space as defined in [RFC5331].

* The label and peer policies (including filters) are defined using

prefix-set and neighbor-set, respectively, as defined in the routing-policy model [RFC9067].

* This model uses the terms LDP "neighbor/adjacency", "session", and "peer" with the following semantics:

  Neighbor/Adjacency: An LDP-enabled Label Switching Router (LSR) that is discovered through LDP discovery mechanisms.

  Session: An LDP neighbor with whom a TCP connection has been established.

  Peer: An LDP session that has successfully progressed beyond its initialization phase and is either already exchanging the bindings or is ready to do so.

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

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

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

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

4. The Complete Tree

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

```
module: ietf-mpls-ldp
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw mpls-ldp
       +--rw global
       |  +--rw capability
       |  |  +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
       |  |  |  +--rw ldp-ext:enabled?   boolean
       |  |  +--rw ldp-ext:typed-wildcard-fec
       |  |  |       {capability-typed-wildcard-fec}?
       |  |  |  +--rw ldp-ext:enabled?   boolean
       |  |  +--rw ldp-ext:upstream-label-assignment
       |  |          {capability-upstream-label-assignment}?
       |  |     +--rw ldp-ext:enabled?   boolean
       |  +--rw graceful-restart
       |  |  +--rw enabled?                  boolean
       |  |  +--rw reconnect-time?           uint16
       |  |  +--rw recovery-time?            uint16
       |  |  +--rw forwarding-holdtime?      uint16
       |  |  +--rw ldp-ext:helper-enabled?   boolean
       |  |          {graceful-restart-helper-mode}?
       |  +--rw lsr-id?
       |  |       rt-types:router-id
       |  +--rw address-families
```

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

```
|  |        |  +--ro ldp-ext:advertisement-type?
|  |        |  |       advertised-received
|  |        |  +--ro ldp-ext:peer
|  |        |     +--ro ldp-ext:lsr-id?           leafref
|  |        |     +--ro ldp-ext:label-space-id?   leafref
|  |        +--ro ldp-ext:fec-label* [fec]
|  |           +--ro ldp-ext:fec     inet:ipv6-prefix
|  |           +--ro ldp-ext:peer*
|  |                [lsr-id label-space-id advertisement-type]
|  |              +--ro ldp-ext:lsr-id                leafref
|  |              +--ro ldp-ext:label-space-id        leafref
|  |              +--ro ldp-ext:advertisement-type
|  |              |       advertised-received
|  |              +--ro ldp-ext:label?
|  |              |       rt-types:mpls-label
|  |              +--ro ldp-ext:used-in-forwarding?   boolean
|  +--rw ldp-ext:forwarding-nexthop
|  |       {forwarding-nexthop-config}?
|  |  +--rw ldp-ext:interfaces
|  |     +--rw ldp-ext:interface* [name]
|  |        +--rw ldp-ext:name              if:interface-ref
|  |        +--rw ldp-ext:address-family* [afi]
|  |           +--rw ldp-ext:afi            identityref
|  |           +--rw ldp-ext:ldp-disable?   boolean
|  +--rw ldp-ext:igp-synchronization-delay?   uint16
+--rw discovery
|  +--rw interfaces
|  |  +--rw hello-holdtime?   uint16
|  |  +--rw hello-interval?   uint16
|  |  +--rw interface* [name]
|  |     +--rw name
|  |     |       if:interface-ref
|  |     +--ro next-hello?                          uint16
|  |     +--rw address-families
|  |     |  +--rw ipv4!
|  |     |  |  +--rw enabled?                     boolean
|  |     |  |  +--ro hello-adjacencies
|  |     |  |  |  +--ro hello-adjacency* [adjacent-address]
|  |     |  |  |     +--ro adjacent-address
|  |     |  |  |     |       inet:ipv4-address
|  |     |  |  |     +--ro flag*               identityref
|  |     |  |  |     +--ro hello-holdtime
|  |     |  |  |     |  +--ro adjacent?     uint16
|  |     |  |  |     |  +--ro negotiated?   uint16
|  |     |  |  |     |  +--ro remaining?    uint16
|  |     |  |  |     +--ro next-hello?         uint16
|  |     |  |  |     +--ro statistics
|  |     |  |  |     |  +--ro discontinuity-time
|  |     |  |  |     |  |       yang:date-and-time
|  |     |  |  |     |  +--ro hello-received?
|  |     |  |  |     |  |       yang:counter64
|  |     |  |  |     |  +--ro hello-dropped?
|  |     |  |  |     |          yang:counter64
|  |     |  |  |     +--ro peer
|  |     |  |  |        +--ro lsr-id?           leafref
|  |     |  |  |        +--ro label-space-id?   leafref
|  |     |  |  +--rw ldp-ext:transport-address?   union
|  |     |  +--rw ldp-ext:ipv6!
|  |     |     +--rw ldp-ext:enabled?             boolean
|  |     |     +--ro ldp-ext:hello-adjacencies
|  |     |     |  +--ro ldp-ext:hello-adjacency*
|  |     |     |          [adjacent-address]
|  |     |     |     +--ro ldp-ext:adjacent-address
|  |     |     |     |       inet:ipv6-address
|  |     |     |     +--ro ldp-ext:flag*
|  |     |     |     |       identityref
```

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

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

```
|  |  |     +--rw ldp-ext:accept
|  |  |        +--rw ldp-ext:prefix-list?   prefix-list-ref
|  |  +--rw ldp-ext:ipv6!
|  |     +--ro ldp-ext:hello-adjacencies
|  |     |  +--ro ldp-ext:hello-adjacency*
|  |     |           [local-address adjacent-address]
|  |     |     +--ro ldp-ext:local-address
|  |     |     |       inet:ipv6-address
|  |     |     +--ro ldp-ext:adjacent-address
|  |     |     |       inet:ipv6-address
|  |     |     +--ro ldp-ext:flag*
|  |     |     |       identityref
|  |     |     +--ro ldp-ext:hello-holdtime
|  |     |     |  +--ro ldp-ext:adjacent?     uint16
|  |     |     |  +--ro ldp-ext:negotiated?   uint16
|  |     |     |  +--ro ldp-ext:remaining?    uint16
|  |     |     +--ro ldp-ext:next-hello?         uint16
|  |     |     +--ro ldp-ext:statistics
|  |     |     |  +--ro ldp-ext:discontinuity-time
|  |     |     |  |       yang:date-and-time
|  |     |     |  +--ro ldp-ext:hello-received?
|  |     |     |  |       yang:counter64
|  |     |     |  +--ro ldp-ext:hello-dropped?
|  |     |     |          yang:counter64
|  |     |     +--ro ldp-ext:interface?
|  |     |              if:interface-ref
|  |     +--rw ldp-ext:label-policy
|  |        +--rw ldp-ext:advertise
|  |        |  +--rw ldp-ext:prefix-list?   prefix-list-ref
|  |        +--rw ldp-ext:accept
|  |           +--rw ldp-ext:prefix-list?   prefix-list-ref
|  +--ro label-advertisement-mode
|  |  +--ro local?        label-adv-mode
|  |  +--ro peer?         label-adv-mode
|  |  +--ro negotiated?   label-adv-mode
|  +--ro next-keep-alive?              uint16
|  +--ro received-peer-state
|  |  +--ro graceful-restart
|  |  |  +--ro enabled?          boolean
|  |  |  +--ro reconnect-time?   uint16
|  |  |  +--ro recovery-time?    uint16
|  |  +--ro capability
|  |     +--ro end-of-lib
|  |     |  +--ro enabled?   boolean
|  |     +--ro typed-wildcard-fec
|  |     |  +--ro enabled?   boolean
|  |     +--ro upstream-label-assignment
|  |        +--ro enabled?   boolean
|  +--ro session-holdtime
|  |  +--ro peer?         uint16
|  |  +--ro negotiated?   uint16
|  |  +--ro remaining?    uint16
|  +--ro session-state?                enumeration
|  +--ro tcp-connection
|  |  +--ro local-address?    inet:ip-address
|  |  +--ro local-port?       inet:port-number
|  |  +--ro remote-address?   inet:ip-address
|  |  +--ro remote-port?      inet:port-number
|  +--ro up-time?
|  |       rt-types:timeticks64
|  +--ro statistics
|  |  +--ro discontinuity-time          yang:date-and-time
|  |  +--ro received
|  |  |  +--ro total-octets?          yang:counter64
|  |  |  +--ro total-messages?        yang:counter64
|  |  |  +--ro address?               yang:counter64
```

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

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

```
           +---w protocol-name?    leafref
           +---w lsr-id?           leafref
           +---w label-space-id?   leafref

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

Figure 2: Complete Tree

5. Configuration

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

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

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

```
        augment /rt:routing/rt:control-plane-protocols:
                  /rt:control-plane-protocol:
          +-- mpls-ldp
              +-- global
              |   +-- ...
              |   +-- ...
              |   +-- address-families
              |   |   +-- ipv4
              |   |       +-- . . .
              |   |       +-- . . .
              |   +-- capability
              |       +-- ...
              |       +-- ...
              +-- discovery
              |   +-- interfaces
              |   |   +-- ...
              |   |   +-- ...
              |   |   +-- interface* [interface]
              |   |       +-- ...
              |   |       +-- address-families
              |   |           +-- ipv4
```

```
            |   |              +-- ...
            |   |              +-- ...
            |   +-- targeted
            |       +-- ...
            |       +-- address-families
            |           +-- ipv4
            |               +-- target* [adjacent-address]
            |                   +-- ...
            |                   +-- ...
            +-- peers
                +-- ...
                +-- ...
                +-- peer* [lsr-id label-space-id]
                    +-- ...
                    +-- ...
```

Figure 3: Base Configuration Organization

The following is the high-level configuration organization for the extended LDP module:

```
augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol
  +-- mpls-ldp
      +-- global
      |   +-- ...
      |   +-- ...
      |   +-- address-families
      |   |   +-- ipv4
      |   |   |   +-- . . .
      |   |   |   +-- . . .
      |   |   |   +-- label-policy
      |   |   |       +-- ...
      |   |   |       +-- ...
      |   |   +-- ipv6
      |   |       +-- . . .
      |   |       +-- . . .
      |   |       +-- label-policy
      |   |           +-- ...
      |   |           +-- ...
      |   +-- capability
      |   |   +-- ...
      |   |   +-- ...
      |   +-- discovery
      |       +-- interfaces
      |       |   +-- ...
      |       |   +-- ...
      |       |   +-- interface* [interface]
      |       |       +-- ...
      |       |       +-- address-families
      |       |           +-- ipv4
      |       |           |   +-- ...
      |       |           |   +-- ...
      |       |           +-- ipv6
      |       |               +-- ...
      |       |               +-- ...
      |       +-- targeted
      |           +-- ...
      |           +-- address-families
      |               +-- ipv6
      |                   +- target* [adjacent-address]
      |                      +- ...
      |                      +- ...
      +-- forwarding-nexthop
      |   +-- ...
      |   +-- ...
```

```
        +-- peers
            +-- ...
            +-- ...
            +-- peer*
                +-- ...
                +-- ...
                +-- label-policy
                |   +-- ..
                +-- address-families
                    +-- ipv4
                    |   +-- ...
                    +-- ipv6
                        +-- ...
```

Figure 4: Extended Configuration Organization

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

5.1. Configuration Hierarchy

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

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

The following subsections briefly explain these configuration areas.

5.1.1. Global Parameters

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

5.1.2. Capabilities Parameters

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

5.1.3. Per-Address-Family Parameters

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

5.1.4. Hello Discovery Parameters

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

The "interfaces" container is used 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 the "interfaces" container. The model defines parameters to configure per-interface non-AF-related items as well as per-interface per-AF items. The example of the former is interface Hello timers, and an example of the latter is enabling hellos for a given AF under an interface.

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

5.1.5. Peer Parameters

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

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

apply to a given peer.

5.1.6. Forwarding Parameters

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

6. Operational State

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

The following are main areas for which LDP operational state is defined:

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

6.1. Adjacency State

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

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

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

```
            +--ro hello-adjacencies
               +--ro hello-adjacencies*
               |                  [local-address adjacent-address]
               +--ro local-address
                  +--ro adjacent-address
                     . . . .
                     . . . .
```

Figure 5: Adjacency State

6.2. Peer State

Peer-related state is presented under a peers tree. This is one of the core states 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.

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

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

Figure 6: Peer State

6.3. Bindings State

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

```
     FEC-Label bindings:
         FEC 203.0.113.1/32:
           advertised: local-label 16000
             peer 192.0.2.1:0
             peer 192.0.2.2:0
```

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

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

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

Figure 7: Example Bindings

Note that all local addresses are advertised to all peers; hence, there is 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.

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

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

Figure 8: Bindings State

6.4. Capabilities State

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

The following captures high-level tree hierarchy for LDP capabilities state.

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

Figure 9: Capabilities State

7. Notifications

This model defines a list of notifications to inform the 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 one Next Hop Label Forwarding Entry (NHLFE) with an outgoing label.

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

8. Action

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

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

9. YANG Specification

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

9.1. Base

This YANG module imports types defined in [RFC6991], [RFC8177], [RFC8294], [RFC8343], [RFC8344], [RFC8349], and [RFC9067].

```
<CODE BEGINS> file "ietf-mpls-ldp@2022-03-14.yang"
module ietf-mpls-ldp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
  prefix ldp;

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

```
  prefix rt;
  reference
    "RFC 8349: A YANG Data Model for Routing Management (NMDA
     version)";
}
import ietf-routing-types {
  prefix rt-types;
  reference
    "RFC 8294: Common YANG Data Types for the Routing Area";
}
import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343: A YANG Data Model for Interface Management";
}
import ietf-ip {
  prefix ip;
  reference
    "RFC 8344: A YANG Data Model for IP Management";
}
import ietf-key-chain {
  prefix key-chain;
  reference
    "RFC 8177: YANG Data Model for Key Chains";
}

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

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

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

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

   Author:   Santosh Easale
             <mailto:santosh_easale@berkeley.edu>

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

   Author:   Himanshu Shah
             <mailto:hshah@ciena.com>";
description
  "This YANG module defines the essential components for the
   management of Multiprotocol Label Switching (MPLS) Label
   Distribution Protocol (LDP).  It is also the base model to
   be augmented for Multipoint LDP (mLDP).

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

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

   This version of this YANG module is part of RFC 9070; see the
```

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

revision 2022-03-14 {
  description
    "Initial revision.";
  reference
    "RFC 9070: YANG Data Model for MPLS LDP";
}

/*
 * Typedefs
 */

typedef advertised-received {
  type enumeration {
    enum advertised {
      description
        "Advertised information.";
    }
    enum received {
      description
        "Received information.";
    }
  }
  description
    "Received or advertised.";
}

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

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

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

```
      value 2;
      description
        "Operational status changed to down.";
    }
  }
  description
    "Operational status event type for notifications.";
}

/*
 * Identities
 */

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

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

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

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

/*
 * Groupings
 */

grouping adjacency-state-attributes {
  description
    "The operational state attributes of an LDP Hello adjacency,
     which can used for basic and extended discoveries, in IPv4 and
     IPv6 address families.";
  leaf-list flag {
    type identityref {
      base adjacency-flag-base;
    }
    description
      "One or more flags to indicate whether the adjacency is
       actively created, passively accepted, or both.";
  }
  container hello-holdtime {
    description
      "Containing Hello holdtime state information.";
    leaf adjacent {
      type uint16;
      units "seconds";
      description
        "The holdtime value learned from the adjacent LSR.";
    }
    leaf negotiated {
      type uint16;
```

```
      units "seconds";
      description
        "The holdtime negotiated between this LSR and the adjacent
         LSR.";
    }
    leaf remaining {
      type uint16;
      units "seconds";
      description
        "The time remaining until the holdtime timer expires.";
    }
  }
  leaf next-hello {
    type uint16;
    units "seconds";
    description
      "The time when the next Hello message will be sent.";
  }
  container statistics {
    description
      "Statistics objects.";
    leaf discontinuity-time {
      type yang:date-and-time;
      mandatory true;
      description
        "The time on the most recent occasion at which any one or
         more of this interface's counters suffered a
         discontinuity.  If no such discontinuities have occurred
         since the last re-initialization of the local management
         subsystem, then this node contains the time the local
         management subsystem re-initialized itself.";
    }
    leaf hello-received {
      type yang:counter64;
      description
        "The number of Hello messages received.";
    }
    leaf hello-dropped {
      type yang:counter64;
      description
        "The number of Hello messages dropped.";
    }
  } // statistics
} // adjacency-state-attributes

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

```
      range "5..1200";
    }
    units "seconds";
    description
      "The interval between consecutive LDP link Hello messages
       used in basic LDP discovery.
       This leaf may be configured at the per-interface level or
       the global level, with precedence given to the value at the
       per-interface level.  If the leaf is not configured at
       either level, the default value at the global level is
       used.";
  }
} // basic-discovery-timers

grouping binding-address-state-attributes {
  description
    "Operational state attributes of an address binding, used in
     IPv4 and IPv6 address families.";
  leaf advertisement-type {
    type advertised-received;
    description
      "Received or advertised.";
  }
  container peer {
    when "../advertisement-type = 'received'" {
      description
        "Applicable for received address.";
    }
    description
      "LDP peer from which this address is received.";
    uses ldp-peer-ref-from-binding;
  }
} // binding-address-state-attributes

grouping binding-label-state-attributes {
  description
    "Operational state attributes for a FEC-Label binding, used in
     IPv4 and IPv6 address families.";
  list peer {
    key "lsr-id label-space-id advertisement-type";
    description
      "List of advertised and received peers.";
    uses ldp-peer-ref-from-binding {
      description
        "The LDP peer from which this binding is received, or to
         which this binding is advertised.
         The peer is identified by its LDP ID, which consists of
         the LSR Id and the label space Id.";
    }
    leaf advertisement-type {
      type advertised-received;
      description
        "Received or advertised.";
    }
    leaf label {
      type rt-types:mpls-label;
      description
        "Advertised (outbound) or received (inbound)
         label.";
    }
    leaf used-in-forwarding {
      type boolean;
      description
        "'true' if the label is used in forwarding.";
    }
  } // peer
```

```
} // binding-label-state-attributes

grouping graceful-restart-attributes-per-peer {
  description
    "Per-peer graceful restart attributes.
     On the local side, these attributes are configuration and
     operational state data.  On the peer side, these attributes
     are operational state data received from the peer.";
  container graceful-restart {
    description
      "Attributes for graceful restart.";
    leaf enabled {
      type boolean;
      description
        "Enable or disable graceful restart.
         This leaf may be configured at the per-peer level or the
         global level, with precedence given to the value at the
         per-peer level.  If the leaf is not configured at either
         level, the default value at the global level is used.";
    }
    leaf reconnect-time {
      type uint16 {
        range "10..1800";
      }
      units "seconds";
      description
        "Specifies the time interval that the remote LDP peer
         must wait for the local LDP peer to reconnect after the
         remote peer detects the LDP communication failure.
         This leaf may be configured at the per-peer level or the
         global level, with precedence given to the value at the
         per-peer level.  If the leaf is not configured at either
         level, the default value at the global level is used.";
    }
    leaf recovery-time {
      type uint16 {
        range "30..3600";
      }
      units "seconds";
      description
        "Specifies the time interval, in seconds, that the remote
         LDP peer preserves its MPLS forwarding state after
         receiving the Initialization message from the restarted
         local LDP peer.
         This leaf may be configured at the per-peer level or the
         global level, with precedence given to the value at the
         per-peer level.  If the leaf is not configured at either
         level, the default value at the global level is used.";
    }
  } // graceful-restart
} // graceful-restart-attributes-per-peer

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

```
  }
}

grouping ldp-peer-ref-absolute {
  description
    "An absolute reference to an LDP peer, by the LDP ID, which
     consists of the LSR Id and the label space Id.";
  leaf protocol-name {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
         + "rt:control-plane-protocol/rt:name";
    }
    description
      "The name of the LDP protocol instance.";
  }
  leaf lsr-id {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
         + "rt:control-plane-protocol"
         + "[rt:name=current()/../protocol-name]/"
         + "ldp:mpls-ldp/ldp:peers/ldp:peer/ldp:lsr-id";
    }
    description
      "The LSR Id of the peer, as a portion of the peer LDP ID.";
  }
  leaf label-space-id {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
         + "rt:control-plane-protocol"
         + "[rt:name=current()/../protocol-name]/"
         + "ldp:mpls-ldp/ldp:peers/"
         + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
         + "ldp:label-space-id";
    }
    description
      "The label space Id of the peer, as a portion of the peer
       LDP ID.";
  }
} // ldp-peer-ref-absolute

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

grouping ldp-peer-ref-from-interface {
  description
    "A relative reference to an LDP peer, by the LDP ID, which
```

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

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

grouping peer-attributes {
  description
    "Peer configuration attributes, used in the per-peer setting
     can in the all-peer setting.";
  leaf session-ka-holdtime {
    type uint16 {
      range "45..3600";
    }
    units "seconds";
    description
      "The time interval after which an inactive LDP session
```

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

  grouping peer-authentication {
    description
      "Peer authentication container, used in the per-peer setting
       can in the all-peer setting.";
    container authentication {
      description
        "Containing authentication information.";
      choice authentication-type {
        description
          "Choice of authentication.";
        case password {
          leaf key {
            type string;
            description
              "This leaf specifies the authentication key.  The
               length of the key may be dependent on the
               cryptographic algorithm.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Cryptographic algorithm associated with key.";
          }
        }
      }
    }
  } // peer-authentication

  grouping peer-state-derived {
    description
      "The peer state information derived from the LDP protocol
       operations.";
    container label-advertisement-mode {
      config false;
      description
        "Label advertisement mode state.";
      leaf local {
        type label-adv-mode;
        description
          "Local Label Advertisement Mode.";
```

```
    }
    leaf peer {
      type label-adv-mode;
      description
        "Peer Label Advertisement Mode.";
    }
    leaf negotiated {
      type label-adv-mode;
      description
        "Negotiated Label Advertisement Mode.";
    }
  }
  leaf next-keep-alive {
    type uint16;
    units "seconds";
    config false;
    description
      "Time duration from now until sending the next KeepAlive
       message.";
  }
  container received-peer-state {
    config false;
    description
      "Operational state information learned from the peer.";
    uses graceful-restart-attributes-per-peer;
    container capability {
      description
        "Peer capability information.";
      container end-of-lib {
        description
          "Peer's end-of-lib capability.";
        leaf enabled {
          type boolean;
          description
            "'true' if peer's end-of-lib capability is enabled.";
        }
      }
      container typed-wildcard-fec {
        description
          "Peer's typed-wildcard-fec capability.";
        leaf enabled {
          type boolean;
          description
            "'true' if peer's typed-wildcard-fec capability is
              enabled.";
        }
      }
      container upstream-label-assignment {
        description
          "Peer's upstream label assignment capability.";
        leaf enabled {
          type boolean;
          description
            "'true' if peer's upstream label assignment is
             enabled.";
        }
      }
    } // capability
  } // received-peer-state
  container session-holdtime {
    config false;
    description
      "Session holdtime state.";
    leaf peer {
      type uint16;
      units "seconds";
```

```
      description
        "Peer holdtime.";
    }
    leaf negotiated {
      type uint16;
      units "seconds";
      description
        "Negotiated holdtime.";
    }
    leaf remaining {
      type uint16;
      units "seconds";
      description
        "Remaining holdtime.";
    }
  } // session-holdtime
  leaf session-state {
    type enumeration {
      enum non-existent {
        description
          "NON EXISTENT state.  Transport disconnected.";
      }
      enum initialized {
        description
          "INITIALIZED state.";
      }
      enum openrec {
        description
          "OPENREC state.";
      }
      enum opensent {
        description
          "OPENSENT state.";
      }
      enum operational {
        description
          "OPERATIONAL state.";
      }
    }
    config false;
    description
      "Representing the operational status of the LDP session.";
    reference
      "RFC 5036: LDP Specification, Sec. 2.5.4.";
  }
  container tcp-connection {
    config false;
    description
      "TCP connection state.";
    leaf local-address {
      type inet:ip-address;
      description
        "Local address.";
    }
    leaf local-port {
      type inet:port-number;
      description
        "Local port number.";
    }
    leaf remote-address {
      type inet:ip-address;
      description
        "Remote address.";
    }
    leaf remote-port {
      type inet:port-number;
```

```
      description
        "Remote port number.";
    }
  } // tcp-connection
  leaf up-time {
    type rt-types:timeticks64;
    config false;
    description
      "The number of time ticks (hundredths of a second) since the
       state of the session with the peer changed to
       OPERATIONAL.";
  }
  container statistics {
    config false;
    description
      "Statistics objects.";
    leaf discontinuity-time {
      type yang:date-and-time;
      mandatory true;
      description
        "The time on the most recent occasion at which any one or
         more of this interface's counters suffered a
         discontinuity.  If no such discontinuities have occurred
         since the last re-initialization of the local management
         subsystem, then this node contains the time the local
         management subsystem re-initialized itself.";
    }
    container received {
      description
        "Inbound statistics.";
      uses statistics-peer-received-sent;
    }
    container sent {
      description
        "Outbound statistics.";
      uses statistics-peer-received-sent;
    }
    leaf total-addresses {
      type uint32;
      description
        "The number of learned addresses.";
    }
    leaf total-labels {
      type uint32;
      description
        "The number of learned labels.";
    }
    leaf total-fec-label-bindings {
      type uint32;
      description
        "The number of learned label-address bindings.";
    }
  } // statistics
} // peer-state-derived

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

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

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

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

```
description
  "LDP augmentation to routing control plane protocol
   configuration and state.";
container mpls-ldp {
  must "not (../../rt:control-plane-protocol"
     + "[derived-from-or-self(rt:type, 'ldp:mpls-ldp')]"
     + "[rt:name!=current()/../rt:name])" {
    description
      "Only one LDP instance is allowed.";
  }
  description
    "Containing configuration and operational data for the LDP
     protocol.";
  container global {
    description
      "Global attributes for LDP.";
    container capability {
      description
        "Containing the LDP capability data.  The container is
         used for augmentations.";
      reference
        "RFC 5036: LDP Specification, Sec. 1.5.";
    }
    container graceful-restart {
      description
        "Attributes for graceful restart.";
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enable or disable graceful restart.";
      }
      leaf reconnect-time {
        type uint16 {
          range "10..1800";
        }
        units "seconds";
        default "120";
        description
          "Specifies the time interval that the remote LDP peer
           must wait for the local LDP peer to reconnect after
           the remote peer detects the LDP communication
           failure.";
      }
      leaf recovery-time {
        type uint16 {
          range "30..3600";
        }
        units "seconds";
        default "120";
        description
          "Specifies the time interval, in seconds, that the
           remote LDP peer preserves its MPLS forwarding state
           after receiving the Initialization message from the
           restarted local LDP peer.";
      }
      leaf forwarding-holdtime {
        type uint16 {
          range "30..3600";
        }
        units "seconds";
        default "180";
        description
          "Specifies the time interval, in seconds, before the
           termination of the recovery phase.";
      }
```

```
} // graceful-restart
leaf lsr-id {
  type rt-types:router-id;
  description
    "Specifies the value to act as the LDP LSR Id.
     If this attribute is not specified, LDP uses the router
     ID as determined by the system.";
}
container address-families {
  description
    "Per-address-family configuration and operational state.
     The address family can be either IPv4 or IPv6.";
  container ipv4 {
    presence "Present if IPv4 is enabled, unless the
              'enabled' leaf is set to 'false'.";
    description
      "Containing data related to the IPv4 address family.";
    leaf enabled {
      type boolean;
      default "true";
      description
        "'false' to disable the address family.";
    }
    leaf label-distribution-control-mode {
      type enumeration {
        enum independent {
          description
            "Independent label distribution control.";
        }
        enum ordered {
          description
            "Ordered label distribution control.";
        }
      }
      config false;
      description
        "Label distribution control mode.";
      reference
        "RFC 5036: LDP Specification, Sec. 2.6.";
    }
    // ipv4 bindings
    container bindings {
      config false;
      description
        "LDP address and label binding information.";
      list address {
        key "address";
        description
          "List of address bindings learned by LDP.";
        leaf address {
          type inet:ipv4-address;
          description
            "The IPv4 address learned from an Address
             message received from or advertised to a peer.";
        }
        uses binding-address-state-attributes;
      }
      list fec-label {
        key "fec";
        description
          "List of FEC-label bindings learned by LDP.";
        leaf fec {
          type inet:ipv4-prefix;
          description
            "The prefix FEC value in the FEC-Label binding,
             learned in a Label Mapping message received from
```

```
              or advertised to a peer.";
          }
          uses binding-label-state-attributes;
        }
      } // bindings
    } // ipv4
  } // address-families
} // global
container discovery {
  description
    "Neighbor-discovery configuration and operational state.";
  container interfaces {
    description
      "A list of interfaces for LDP Basic Discovery.";
    reference
      "RFC 5036: LDP Specification, Sec. 2.4.1.";
    uses basic-discovery-timers {
      refine "hello-holdtime" {
        default "15";
      }
      refine "hello-interval" {
        default "5";
      }
    }
    list interface {
      key "name";
      description
        "List of LDP interfaces used for LDP Basic Discovery.";
      uses ldp-interface-ref;
      leaf next-hello {
        type uint16;
        units "seconds";
        config false;
        description
          "Time to send the next Hello message.";
      }
      container address-families {
        description
          "Container for address families.";
        container ipv4 {
          presence "Present if IPv4 is enabled, unless the
                    'enabled' leaf is set to 'false'.";
          description
            "IPv4 address family.";
          leaf enabled {
            type boolean;
            default "true";
            description
              "Set to false to disable the address family on
               the interface.";
          }
          container hello-adjacencies {
            config false;
            description
              "Containing a list of Hello adjacencies.";
            list hello-adjacency {
              key "adjacent-address";
              config false;
              description
                "List of Hello adjacencies.";
              leaf adjacent-address {
                type inet:ipv4-address;
                description
                  "Neighbor address of the Hello adjacency.";
              }
              uses adjacency-state-attributes;
```

```
            uses ldp-peer-ref-from-interface;
          }
        }
      } // ipv4
    } // address-families
  } // interface
} // interfaces
container targeted {
  description
    "A list of targeted neighbors for extended discovery.";
  leaf hello-holdtime {
    type uint16 {
      range "15..3600";
    }
    units "seconds";
    default "45";
    description
      "The time interval for which an LDP targeted Hello
       adjacency is maintained in the absence of targeted
       Hello messages from an LDP neighbor.";
  }
  leaf hello-interval {
    type uint16 {
      range "5..3600";
    }
    units "seconds";
    default "15";
    description
      "The interval between consecutive LDP targeted Hello
       messages used in extended LDP discovery.";
  }
  container hello-accept {
    description
      "LDP policy to control the acceptance of extended
       neighbor-discovery Hello messages.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "'true' to accept; 'false' to deny.";
    }
  }
  container address-families {
    description
      "Container for address families.";
    container ipv4 {
      presence "Present if IPv4 is enabled.";
      description
        "IPv4 address family.";
      container hello-adjacencies {
        config false;
        description
          "Containing a list of Hello adjacencies.";
        list hello-adjacency {
          key "local-address adjacent-address";
          description
            "List of Hello adjacencies.";
          leaf local-address {
            type inet:ipv4-address;
            description
              "Local address of the Hello adjacency.";
          }
          leaf adjacent-address {
            type inet:ipv4-address;
            description
              "Neighbor address of the Hello adjacency.";
```

```
            }
            uses adjacency-state-attributes;
            uses ldp-peer-ref-from-target;
          }
        }
        list target {
          key "adjacent-address";
          description
            "Targeted discovery params.";
          leaf adjacent-address {
            type inet:ipv4-address;
            description
              "Configures a remote LDP neighbor for the
               extended LDP discovery.";
          }
          leaf enabled {
            type boolean;
            default "true";
            description
              "'true' to enable the target.";
          }
          leaf local-address {
            type inet:ipv4-address;
            description
              "The local address used as the source address to
               send targeted Hello messages.
               If the value is not specified, the
               transport address is used as the source
               address.";
          }
        } // target
      } // ipv4
    } // address-families
  } // targeted
} // discovery
container peers {
  description
    "Peers configuration attributes.";
  uses peer-authentication;
  uses peer-attributes {
    refine "session-ka-holdtime" {
      default "180";
    }
    refine "session-ka-interval" {
      default "60";
    }
  }
  list peer {
    key "lsr-id label-space-id";
    description
      "List of peers.";
    leaf lsr-id {
      type rt-types:router-id;
      description
        "The LSR Id of the peer, used to identify the globally
         unique LSR.  This is the first four octets of the LDP
         ID.  This leaf is used together with the leaf
         'label-space-id' to form the LDP ID.";
      reference
        "RFC 5036: LDP Specification, Sec. 2.2.2.";
    }
    leaf label-space-id {
      type uint16;
      description
        "The label space Id of the peer, used to identify a
         specific label space within the LSR.  This is the last
```

```
             two octets of the LDP ID.  This leaf is used together
             with the leaf 'lsr-id' to form the LDP ID.";
          reference
            "RFC 5036: LDP Specification, Sec. 2.2.2.";
        }
        uses peer-authentication;
        container address-families {
          description
            "Per-vrf per-af params.";
          container ipv4 {
            presence "Present if IPv4 is enabled.";
            description
              "IPv4 address family.";
            container hello-adjacencies {
              config false;
              description
                "Containing a list of Hello adjacencies.";
              list hello-adjacency {
                key "local-address adjacent-address";
                description
                  "List of Hello adjacencies.";
                leaf local-address {
                  type inet:ipv4-address;
                  description
                    "Local address of the Hello adjacency.";
                }
                leaf adjacent-address {
                  type inet:ipv4-address;
                  description
                    "Neighbor address of the Hello adjacency.";
                }
                uses adjacency-state-attributes;
                leaf interface {
                  type if:interface-ref;
                  description
                    "Interface for this adjacency.";
                }
              }
            }
          } // ipv4
        } // address-families
        uses peer-state-derived;
      } // list peer
    } // peers
  } // container mpls-ldp
}

/*
 * RPCs
 */

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

rpc mpls-ldp-clear-hello-adjacency {
```

```
  description
    "Clears the Hello adjacency.";
  input {
    container hello-adjacency {
      description
        "Link adjacency or targeted adjacency.  If this is not
         provided, then all Hello adjacencies are cleared.";
      leaf protocol-name {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "The name of the LDP protocol instance.";
      }
      choice hello-adjacency-type {
        description
          "Adjacency type.";
        case targeted {
          container targeted {
            presence "Present to clear targeted adjacencies.";
            description
              "Clear targeted adjacencies.";
            leaf target-address {
              type inet:ip-address;
              description
                "The target address.  If this is not provided, then
                 all targeted adjacencies are cleared.";
            }
          }
        }
        case link {
          container link {
            presence "Present to clear link adjacencies.";
            description
              "Clear link adjacencies.";
            leaf next-hop-interface {
              type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                   + "rt:control-plane-protocol/mpls-ldp/"
                   + "discovery/interfaces/interface/name";
              }
              description
                "Interface connecting to a 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 an interface is specified.";
              }
              description
                "IP address of a next hop.  If this is not
                 provided, then adjacencies to all next hops on the
                 given interface are cleared.";
            }
          }
        }
      } // hello-adjacency-type
    } // hello-adjacency
  } // input
} // mpls-ldp-clear-hello-adjacency

rpc mpls-ldp-clear-peer-statistics {
```

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

/*
 * Notifications
 */

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

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

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

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

Figure 10: LDP Base Module

9.2. Extended

This YANG module imports types defined in [RFC5036], [RFC6991], [RFC8349], [RFC8177], [RFC8343], and [RFC9067].

```
  <CODE BEGINS> file "ietf-mpls-ldp-extended@2022-03-14.yang"
  module ietf-mpls-ldp-extended {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended";
    prefix ldp-ext;

    import ietf-inet-types {
      prefix inet;
```

```
  reference
    "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix rt;
  reference
    "RFC 8349: A YANG Data Model for Routing Management (NMDA
     version)";
}
import ietf-key-chain {
  prefix key-chain;
  reference
    "RFC 8177: YANG Data Model for Key Chains";
}
import ietf-mpls-ldp {
  prefix ldp;
  reference
    "RFC 9070: YANG Data Model for MPLS LDP";
}
import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343: A YANG Data Model for Interface Management";
}
import ietf-routing-policy {
  prefix rt-pol;
  reference
    "RFC 9067: A YANG Data Model for Routing Policy";
}

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

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

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

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

   Author:   Santosh Easale
             <mailto:santosh_easale@berkeley.edu>

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

   Author:   Himanshu Shah
             <mailto:hshah@ciena.com>";
description
  "This YANG module defines the extended components for the
   management of Multiprotocol Label Switching (MPLS) Label
   Distribution Protocol (LDP).  It is also the model to
   be augmented for extended Multipoint LDP (mLDP).

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

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Revised BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
```

```
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC 9070; see the
   RFC itself for full legal notices.";

revision 2022-03-14 {
  description
    "Initial revision.";
  reference
    "RFC 9070: YANG Data Model for MPLS LDP";
}

/*
 * Features
 */

feature capability-end-of-lib {
  description
    "This feature indicates that the system allows for the
     configuration of LDP end-of-lib capability.";
}

feature capability-typed-wildcard-fec {
  description
    "This feature indicates that the system allows for the
     configuration of LDP typed-wildcard-fec capability.";
}

feature capability-upstream-label-assignment {
  description
    "This feature indicates that the system allows for the
     configuration of LDP upstream label assignment capability.";
}

feature forwarding-nexthop-config {
  description
    "This feature indicates that the system allows for controlling
     MPLS forwarding on an LDP interface.";
}

feature graceful-restart-helper-mode {
  description
    "This feature indicates that the system supports graceful
     restart helper mode.  We call an LSR to be operating in GR
     helper mode when it advertises 0 as its FT Reconnect Timeout
     in the FT Session TLV.
     Please refer to Section 2 of RFC 3478 for details.";
}

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

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

feature per-interface-timer-config {
  description
    "This feature indicates that the system allows for the
```

```
     configuration of interface Hello timers at the per-interface
     level.";
}

feature per-peer-admin-down {
  description
    "This feature indicates that the system allows for the
     administrative disabling of a peer.";
}

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

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

feature policy-label-assignment-config {
  description
    "This feature indicates that the system allows for the
     configuration of policies to assign labels according to
     certain prefixes.";
}

feature policy-ordered-label-config {
  description
    "This feature indicates that the system allows for the
     configuration of ordered label policies.";
}

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

feature session-downstream-on-demand-config {
  description
    "This feature indicates that the system allows for the
     configuration of session downstream on demand.";
}

/*
 * Typedefs
 */

typedef neighbor-list-ref {
  type leafref {
    path "/rt-pol:routing-policy/rt-pol:defined-sets/"
       + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
  }
  description
    "A type for a reference to a neighbor address list.
     The string value is the name identifier for uniquely
     identifying the referenced address list, which contains a list
     of addresses that a routing policy can applied.";
  reference
    "RFC 9067: A YANG Data Model for Routing Policy";
}
```

```
typedef prefix-list-ref {
  type leafref {
    path "/rt-pol:routing-policy/rt-pol:defined-sets/"
       + "rt-pol:prefix-sets/rt-pol:prefix-set/rt-pol:name";
  }
  description
    "A type for a reference to a prefix list.
     The string value is the name identifier for uniquely
     identifying the referenced prefix set, which contains a list
     of prefixes that a routing policy can applied.";
  reference
    "RFC 9067: A YANG Data Model for Routing Policy";
}

typedef peer-list-ref {
  type leafref {
    path "/rt-pol:routing-policy/rt-pol:defined-sets/"
       + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
  }
  description
    "A type for a reference to a peer address list.
     The string value is the name identifier for uniquely
     identifying the referenced address list, which contains a list
     of addresses that a routing policy can applied.";
  reference
    "RFC 9067: A YANG Data Model for Routing Policy";
}

/*
 * Identities
 */
/*
 * Groupings
 */

grouping address-family-ipv4-augment {
  description
    "Augmentation to address family IPv4.";
  uses policy-container;
  leaf transport-address {
    type inet:ipv4-address;
    description
      "The transport address advertised in LDP Hello messages.
       If this value is not specified, the LDP LSR Id is used as
       the transport address.";
    reference
      "RFC 5036: LDP Specification, Sec. 3.5.2.";
  }
}

grouping authentication-keychain-augment {
  description
    "Augmentation to authentication to add key-chain.";
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
      "key-chain name.
       If not specified, no key chain is used.";
  }
}

grouping capability-augment {
  description
    "Augmentation to capability.";
  container end-of-lib {
    if-feature "capability-end-of-lib";
```

```
    description
      "Configure end-of-lib capability.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "'true' to enable end-of-lib capability.";
    }
  }
  container typed-wildcard-fec {
    if-feature "capability-typed-wildcard-fec";
    description
      "Configure typed-wildcard-fec capability.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "'true' to enable typed-wildcard-fec capability.";
    }
  }
  container upstream-label-assignment {
    if-feature "capability-upstream-label-assignment";
    description
      "Configure upstream label assignment capability.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "'true' to enable upstream label assignment.";
    }
  }
} // capability-augment

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

grouping global-forwarding-nexthop-augment {
  description
    "Augmentation at the global level for controlling MPLS
     forwarding on LDP interfaces.";
  container forwarding-nexthop {
    if-feature "forwarding-nexthop-config";
    description
      "Configuration for controlling MPLS forwarding on LDP
       interfaces.";
    container interfaces {
      description
        "Containing a list of interfaces on which forwarding can be
         disabled.";
      list interface {
        key "name";
```

```
        description
          "List of LDP interfaces on which forwarding can be
           disabled.";
        uses ldp:ldp-interface-ref;
        list address-family {
          key "afi";
          description
            "Per-vrf per-af params.";
          leaf afi {
            type identityref {
              base rt:address-family;
            }
            description
              "Address family type value.";
          }
          leaf ldp-disable {
            type boolean;
            default "false";
            description
              "'true' to disable LDP forwarding on the interface.";
          }
        }
      } // interface
    } // interfaces
  } // forwarding-nexthop
} // global-forwarding-nexthop-augment

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

grouping interface-address-family-ipv4-augment {
  description
    "Augmentation to interface address family IPv4.";
  leaf transport-address {
    type union {
      type enumeration {
        enum use-global-transport-address {
          description
            "Use the transport address set at the global level
             common for all interfaces for this address family.";
        }
        enum use-interface-address {
          description
            "Use interface address as the transport address.";
        }
      }
      type inet:ipv4-address;
    }
    default "use-global-transport-address";
    description
      "IP address to be advertised as the LDP transport address.";
  }
}

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

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

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

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

```
        "Label advertisement acceptance policies.";
      leaf prefix-list {
        type prefix-list-ref;
        description
          "Applies the prefix list to filer incoming label
           advertisements.
           If the value is not specified, no prefix filter
           is applied.";
      }
    }
  }
} // peer-af-policy-container

grouping peer-augment {
  description
    "Augmentation to each peer list entry.";
  leaf admin-down {
    if-feature "per-peer-admin-down";
    type boolean;
    default "false";
    description
      "'true' to disable the peer.";
  }
  uses ldp:graceful-restart-attributes-per-peer {
    if-feature "per-peer-graceful-restart-config";
  }
  uses ldp:peer-attributes {
    if-feature "per-peer-session-attributes-config";
  }
}

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

```
    }
    container prefer-ipv4 {
      presence "Present if IPv4 is preferred for transport
                connection establishment, subject to the
                'peer-list' in this container.";
      description
        "Uses IPv4 as the preferred address family for transport
         connection establishment, subject to the 'peer-list' in
         this container.
         If this container is not present, as a default, IPv6 is
         the preferred address family for transport connection
         establishment.";
      leaf peer-list {
        type peer-list-ref;
        description
          "The name of a peer ACL, to be applied to the IPv4
           transport connections.
           If this value is not specified, no filter is applied,
           and the IPv4 is preferred for all peers.";
      }
    }
  }
} // peers-augment

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

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

/*
 * Configuration and state data nodes
 */
// Forwarding nexthop augmentation to the global tree

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

// global/address-families/ipv6

augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
      + "ldp:address-families" {
  description
    "Global IPv6 augmentation.";
  container ipv6 {
    presence "Present if IPv6 is enabled, unless the 'enabled'
              leaf is set to 'false'.";
    description
      "Containing data related to the IPv6 address family.";
    leaf enabled {
      type boolean;
      default "true";
      description
        "'false' to disable the address family.";
    }
    uses policy-container;
    leaf transport-address {
      type inet:ipv6-address;
      mandatory true;
```

```
      description
        "The transport address advertised in LDP Hello messages.";
    }
    leaf label-distribution-control-mode {
      type enumeration {
        enum independent {
          description
            "Independent label distribution control.";
        }
        enum ordered {
          description
            "Ordered label distribution control.";
        }
      }
      config false;
      description
        "Label distribution control mode.";
      reference
        "RFC 5036: LDP Specification, Sec. 2.6.";
    }
    // ipv6 bindings
    container bindings {
      config false;
      description
        "LDP address and label binding information.";
      list address {
        key "address";
        description
          "List of address bindings learned by LDP.";
        leaf address {
          type inet:ipv6-address;
          description
            "The IPv6 address learned from an Address
             message received from or advertised to a peer.";
        }
        uses ldp:binding-address-state-attributes;
      }
      list fec-label {
        key "fec";
        description
          "List of FEC-label bindings learned by LDP.";
        leaf fec {
          type inet:ipv6-prefix;
          description
            "The prefix FEC value in the FEC-Label binding,
             learned in a Label Mapping message received from
             or advertised to a peer.";
        }
        uses ldp:binding-label-state-attributes;
      }
    } // bindings
  } // ipv6
}

// discovery/interfaces/interface/address-families/ipv6

augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
      + "ldp:interfaces/ldp:interface/"
      + "ldp:address-families" {
  description
    "Interface IPv6 augmentation.";
  container ipv6 {
    presence "Present if IPv6 is enabled, unless the 'enabled'
              leaf is set to 'false'.";
    description
```

```
      "IPv6 address family.";
    leaf enabled {
      type boolean;
      default "true";
      description
        "'false' to disable the address family on the interface.";
    }
    container hello-adjacencies {
      config false;
      description
        "Containing a list of Hello adjacencies.";
      list hello-adjacency {
        key "adjacent-address";
        config false;
        description
          "List of Hello adjacencies.";
        leaf adjacent-address {
          type inet:ipv6-address;
          description
            "Neighbor address of the Hello adjacency.";
        }
        uses ldp:adjacency-state-attributes;
        uses ldp:ldp-peer-ref-from-interface;
      }
    }
  } // ipv6
}

// discovery/targeted/address-families/ipv6

augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
      + "ldp:targeted/ldp:address-families" {
  description
    "Targeted discovery IPv6 augmentation.";
  container ipv6 {
    presence "Present if IPv6 is enabled.";
    description
      "IPv6 address family.";
    container hello-adjacencies {
      config false;
      description
        "Containing a list of Hello adjacencies.";
      list hello-adjacency {
        key "local-address adjacent-address";
        config false;
        description
          "List of Hello adjacencies.";
        leaf local-address {
          type inet:ipv6-address;
          description
            "Local address of the Hello adjacency.";
        }
        leaf adjacent-address {
          type inet:ipv6-address;
          description
            "Neighbor address of the Hello adjacency.";
        }
        uses ldp:adjacency-state-attributes;
        uses ldp:ldp-peer-ref-from-target;
      }
    }
    list target {
      key "adjacent-address";
      description
        "Targeted discovery params.";
```

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

// /peers/peer/state/address-families/ipv6

augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/ldp:mpls-ldp/ldp:peers/"
      + "ldp:peer/ldp:address-families" {
  description
    "Peer state IPv6 augmentation.";
  container ipv6 {
    presence "Present if IPv6 is enabled.";
    description
      "IPv6 address family.";
    container hello-adjacencies {
      config false;
      description
        "Containing a list of Hello adjacencies.";
      list hello-adjacency {
        key "local-address adjacent-address";
        description
          "List of Hello adjacencies.";
        leaf local-address {
          type inet:ipv6-address;
          description
            "Local address of the Hello adjacency.";
        }
        leaf adjacent-address {
          type inet:ipv6-address;
          description
            "Neighbor address of the Hello adjacency.";
        }
        uses ldp:adjacency-state-attributes;
        leaf interface {
          type if:interface-ref;
          description
            "Interface for this adjacency.";
        }
      }
    }
  } // ipv6
}

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

```
 */

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

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

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

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

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

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

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

augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
      + "ldp:targeted/ldp:hello-accept" {
  description
    "Targeted discovery augmentation.";
  leaf neighbor-list {
    if-feature "policy-targeted-discovery-config";
```

```
      type neighbor-list-ref;
      description
        "The name of a neighbor ACL, used to accept Hello messages
         from LDP peers as permitted by the neighbor-list policy.
         If this value is not specified, targeted Hello messages
         from any source are accepted.";
    }
  }

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

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

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

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

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

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

Figure 11: LDP Extended Module

10. Security Considerations

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

10.1. YANG Data Model

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

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

10.1.1. Writable Nodes

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

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

/mpls-ldp/discovery/interfaces/interface: Adding LDP on any unprotected interface could allow an LDP Hello adjacency to be formed with an unauthorized and malicious neighbor. Once a Hello adjacency is formed, a peer session could progress with this neighbor.

/mpls-ldp/discovery/targeted/hello-accept: Allowing acceptance of targeted-hellos could open LDP to DoS attacks related to incoming targeted hellos from malicious sources.

/mpls-ldp/peers/authentication: Allowing a peer session establishment is typically controlled via LDP authentication where a proper and secure authentication password/key management is warranted.

/mpls-ldp/peers/peer/authentication: Same as above.

10.1.2. Readable Nodes

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

The exposure of LDP databases (such as Hello adjacencies, peers, address bindings, and FEC-Label bindings) beyond the scope of the LDP admin domain may be undesirable. The relevant subtrees and data

nodes are as follows:

* /mpls-ldp/global/address-families/ipv4/bindings/address

* /mpls-ldp/global/address-families/ipv6/bindings/address

* /mpls-ldp/global/address-families/ipv4/bindings/fec-label

* /mpls-ldp/global/address-families/ipv6/bindings/fec-label

* /mpls-ldp/discovery/interfaces/interface/address-families/ipv4/hello-adjacencies

* /mpls-ldp/discovery/interfaces/interface/address-families/ipv6/hello-adjacencies

* /mpls-ldp/discovery/targeted/address-families/ipv4/hello-adjacencies

* /mpls-ldp/discovery/targeted/address-families/ipv6/hello-adjacencies

* /mpls-ldp/peers

The configuration for LDP peer authentication is supported via the key-chain specification [RFC8177] or via direct specification of a key associated with a crypto algorithm (such as MD5). The relevant subtrees and data nodes are as follows:

* /mpls-ldp/peers/authentication

* /mpls-ldp/peers/peer/authentication

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

10.1.3. RPC Operations

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

* mpls-ldp-clear-peer

* mpls-ldp-clear-hello-adjacency

10.1.4. Notifications

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

11. IANA Considerations

Per this document, the following URIs have been registered in the IETF "XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-mpls-ldp

Registrant: The IESG
XML: N/A

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

Per this document, the following YANG modules have been registered in the "YANG Module Names" registry [RFC6020]:

Name: ietf-mpls-ldp
Namespace: urn:ietf:params:xml:ns:yang:ietf-mpls-ldp
Prefix: ldp
Reference: RFC 9070

Name: ietf-mpls-ldp-extended
Namespace: urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended
Prefix: ldp-ext
Reference: RFC 9070

12. Normative References

[RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <https://www.rfc-editor.org/info/rfc3478>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <https://www.rfc-editor.org/info/rfc5036>.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, DOI 10.17487/RFC5331, August 2008, <https://www.rfc-editor.org/info/rfc5331>.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <https://www.rfc-editor.org/info/rfc5443>.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <https://www.rfc-editor.org/info/rfc5561>.

[RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class (FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010, <https://www.rfc-editor.org/info/rfc5918>.

[RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas, "Signaling LDP Label Advertisement Completion", RFC 5919, DOI 10.17487/RFC5919, August 2010, <https://www.rfc-editor.org/info/rfc5919>.

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <https://www.rfc-editor.org/info/rfc5920>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010,

<https://www.rfc-editor.org/info/rfc6020>.

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <https://www.rfc-editor.org/info/rfc6389>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7552] Asati, R., Pignataro, C., Raza, K., Manral, V., and R. Papneja, "Updates to LDP for IPv6", RFC 7552, DOI 10.17487/RFC7552, June 2015, <https://www.rfc-editor.org/info/rfc7552>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of

Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <https://www.rfc-editor.org/info/rfc8529>.

[RFC9067] Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy", RFC 9067, DOI 10.17487/RFC9067, October 2021, <https://www.rfc-editor.org/info/rfc9067>.

13. Informative References

[MPLS-MLDP-YANG] Raza, K., Ed., Liu, X., Esale, S., Andersson, L., Tantsura, J., and S. Krishnaswamy, "YANG Data Model for MPLS mLDP", Work in Progress, Internet-Draft, draft-ietf-mpls-mldp-yang-10, 11 November 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-mpls-mldp-yang-10>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <https://www.rfc-editor.org/info/rfc7307>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Data Tree Example

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

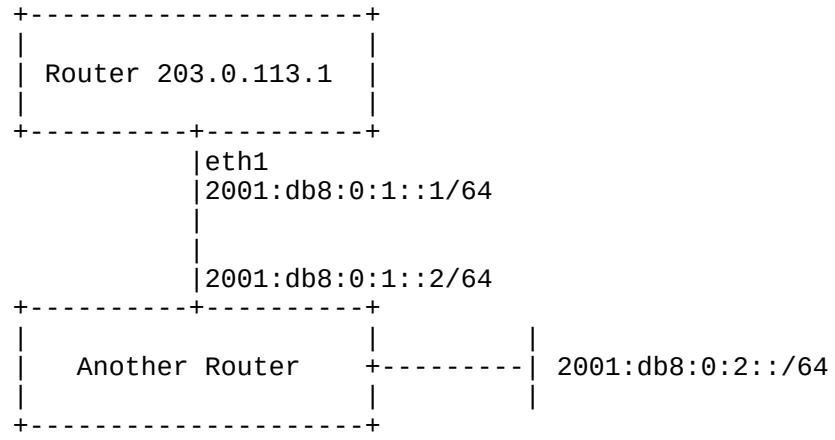

Figure 12: Example Topology

The configuration instance data tree for Router 203.0.113.1 in Figure 12 could be as follows:


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


Figure 13: Example Configuration Data in JSON

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


```
{
```

```
"ietf-interfaces:interfaces": {
  "interface": [
    {
      "name": "eth1",
      "description": "An interface with LDP enabled.",
      "type": "iana-if-type:ethernetCsmacd",
      "phys-address": "00:00:5e:00:53:01",
      "oper-status": "up",
      "statistics": {
        "discontinuity-time": "2018-09-10T15:16:27-05:00"
      },
      "ietf-ip:ipv6": {
        "forwarding": true,
        "mtu": 1500,
        "address": [
          {
            "ip": "2001:db8:0:1::1",
            "prefix-length": 64,
            "origin": "static",
            "status": "preferred"
          },
          {
            "ip": "fe80::200:5eff:fe00:5301",
            "prefix-length": 64,
            "origin": "link-layer",
            "status": "preferred"
          }
        ],
        "neighbor": [
          {
            "ip": "2001:db8:0:1::2",
            "link-layer-address": "00:00:5e:00:53:02",
            "origin": "dynamic",
            "is-router": [null],
            "state": "reachable"
          },
          {
            "ip": "fe80::200:5eff:fe00:5302",
            "link-layer-address": "00:00:5e:00:53:02",
            "origin": "dynamic",
            "is-router": [null],
            "state": "reachable"
          }
        ]
      }
    }
  ]
},
"ietf-routing:routing": {
  "router-id": "203.0.113.1",
  "interfaces": {
    "interface": [
      "eth1"
    ]
  },
  "control-plane-protocols": {
    "control-plane-protocol": [
      {
        "type": "ietf-mpls-ldp:mpls-ldp",
        "name": "ldp-1",
        "ietf-mpls-ldp:mpls-ldp": {
          "global": {
            "address-families": {
              "ietf-mpls-ldp-extended:ipv6": {
                "enabled": true,
                "transport-address": "2001:db8:0:1::1"
```

```
    }
  }
},
"discovery": {
  "interfaces": {
    "interface": [
      {
        "name": "eth1",
        "address-families": {
          "ietf-mpls-ldp-extended:ipv6": {
            "enabled": true,
            "hello-adjacencies": {
              "hello-adjacency": [
                {
                  "adjacent-address":
                  "fe80::200:5eff:fe00:5302",
                  "flag": ["adjacency-flag-active"],
                  "hello-holdtime": {
                    "adjacent": 15,
                    "negotiated": 15,
                    "remaining": 9
                  },
                  "next-hello": 3,
                  "statistics": {
                    "discontinuity-time":
                    "2018-09-10T15:16:27-05:00"
                  },
                  "peer": {
                    "lsr-id": "203.0.113.2",
                    "label-space-id": 0
                  }
                }
              ]
            }
          }
        }
      }
    ]
  }
},
"peers": {
  "peer": [
    {
      "lsr-id": "203.0.113.2",
      "label-space-id": 0,
      "label-advertisement-mode": {
        "local": "downstream-unsolicited",
        "peer": "downstream-unsolicited",
        "negotiated": "downstream-unsolicited"
      },
      "next-keep-alive": 5,
      "session-holdtime": {
        "peer": 180,
        "negotiated": 180,
        "remaining": 78
      },
      "session-state": "operational",
      "tcp-connection": {
        "local-address": "fe80::200:5eff:fe00:5301",
        "local-port": 646,
        "remote-address": "fe80::200:5eff:fe00:5302",
        "remote-port": 646
      },
      "up-time": 3438100,
      "statistics": {
        "discontinuity-time": "2018-09-10T15:16:27-05:00"
```

```
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    }
  }
```

Figure 14: Example Operational Data in JSON

Acknowledgments

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

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

Contributors

Danial Johari
Cisco Systems
Email: dajohari@cisco.com

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

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

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

Reshad Rahman
Email: reshad@yahoo.com

Stephane Litkowski
Cisco Systems
Email: slitkows@cisco.com

Authors' Addresses

Kamran Raza (editor)
Cisco Systems
Canada
Email: skraza@cisco.com

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

Xufeng Liu
IBM Corporation
United States of America
Email: xufeng.liu.ietf@gmail.com

Santosh Easale
Juniper Networks
United States of America
Email: santosh_easale@berkeley.edu

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

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