mpls
Internet-Draft
Intended status: Standards Track
Expires: 23 November 2025

N. Nainar
NVIDIA
M. Sankaranarayanan
J. Rajamanickam, Ed.
Cisco Systems, Inc.
C. Pignataro
North Carolina State University
W. Zheng
Huawei

22 May 2025

YANG Data Model for MPLS LSP Ping
draft-nainar-mpls-lsp-ping-yang-07

Abstract

This document describes the YANG data model for Multi-Protocol Label Switching (MPLS) LSP Ping. The model is based on YANG 1.1 as defined in RFC 7950 and conforms to the Network Management Datastore Architecture (NMDA) as described in RFC 8342.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 November 2025.

Table of Contents

1. Introduction

[RFC8029] describes the mechanism to detect any data-plane failures in MPLS Label Switched Paths (LSPs). The MPLS echo request is triggered from the head end node with different TLVs carrying control plane information such as Target FEC Stack that are used by the transit or the tail end node to validate the path and detect any failures.

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines the mechanism to manage network devices. YANG version 1 defined in [RFC6020] and version 1.1 defined in [RFC7950] is a modular data modeling language used to represent the data structure of the configuration and operational state of any device managed using NETCONF.

This document describes the YANG data model for Multi-Protocol Label Switching (MPLS) LSP Ping. The model is based on YANG 1.1 as defined in [RFC7950] and conforms to the Network Management Datastore Architecture (NMDA) as described in [RFC8342].

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

This document uses the terminologies defined in [RFC8029], [RFC7950], and so the readers are expected to be familiar with the terminologies.

1.3. Tree Diagrams

This document uses the graphical representation of the data models defined in [RFC8340].

2. Design of Data Model

2.1. Scope of Model

[RFC8029] describes the mechanism to detect any data-plane failures in MPLS Label Switched Paths (LSPs). [RFC6425] extends the mechanism further to P2MP MPLS LSPs. [RFC8287] extends the mechanism for Segment Routing with MPLS data plane.

The goal of this document is to produce a data model that provides a common user interface to the MPLS LSP Ping and allows the user to either configure and schedule the testing or to activate on-demand.

2.2. Module Hierarchy Organization

The module is currently defined in a way that can be used to instruct the echo parameters information that the initiator node must include in the payload and in the probe header. The module is defined to use RPC operations to execute LSP Ping and LSP Path tracing with multipath awareness and retrieve the result information.

The future version of the module will be updated to include ways to configure the testing parameters and schedule the testing on any node for continuous monitoring and use notification to receive any change in the monitoring status.

2.3. Optional Capabilities

This module includes the option to configure the MPLS OAM feature that is required in some vendor software to enable the capability. This is not a mandatory feature and so the module is compatible with nodes that does not require any such configuration. The structure of the configuration is as shown below:

```
module: ietf-mpls-lsp-ping
  augment /rt:routing/mpls:mpls:
    +--rw mpls-oam {mpls-oam}?
       +--rw enable?   boolean
```

2.4. RPC Operations

This module defines 3 different RPC operations as below:

* Continuity Check

* Single Path Discovery

* Multi Path Discovery

RPC "continuity-check" triggers LSP Ping from the initiator node to validate the path for any specific FEC defined in the Target FEC Stack and retrieves the response from the responder node as RPC output. The probe count object is used to control the number of probes sent. For each probe sent, a response is expected to be retrieved. The global-flags object can be used to control the FEC validation as defined in Section 6.2.3 of [RFC8029]..

RPC "single-path-discovery" triggers the LSP trace from the Initiator node to trace the path for any specific FEC defined in the Target FEC Stack and retrieves the response from each transit hops as RPC output. While the input parameters are similar to RPC continuity-check, this RPC will instruct the initiator node to send probes by sequentially incrementing the TTL of the probe.

RPC "multi-path-discovery" is similar to "single-path-discovery" with an additional ddmap-hash as an input parameter and retrieves the response from each equal cost multipath (ECMP) transit hops as RPC output.

2.5. Configuration and Notifications

```
module: ietf-mpls-lsp-ping
  augment /rt:routing/mpls:mpls:
    +--rw mpls-oam {mpls-oam}?
       +--rw enable boolean
```

This module includes the option to configure the MPLS OAM feature that is required in some vendor software to enable the capability. This is not a mandatory feature and so the module is compatible with nodes that does not require any such configuration. Leaf "enable" helps user to configure the MPLS OAM feature if required.

2.6. Augment Method

To be Updated.

2.7. The Complete Tree

Following is a complete tree representation of LSP Ping YANG module.

```
module: ietf-mpls-lsp-ping
  augment /rt:routing/mpls:mpls:
    +--rw mpls-oam {mpls-oam}?
       +--rw enable?   boolean

  rpcs:
    +---x continuity-check
    |  +---w input
    |  |  +---w echo-header-parameters
    |  |  |  +---w source-address?        ip-address
    |  |  |  +---w destination-address?   ip-loopback-address
    |  |  |  +---w traffic-class?         mpls-traffic-class
    |  |  |  +---w mpls-entropy-label?    mpls-entropy-label
    |  |  |  +---w header-mpls-ttl?       uint8
    |  |  |  +---w mpls-exp-label?        boolean
    |  |  +---w echo-payload-parameters
    |  |  |  +---w reply-tos-tlv?           boolean
    |  |  |  +---w reply-tos-value
    |  |  |  |  +---w reply-tos-value?   uint8
    |  |  |  +---w probe-size?              uint32
    |  |  |  +---w probe-sweep
    |  |  |  |  +---w min-probe-sweep?   uint16
    |  |  |  |  +---w max-probe-sweep?   uint16
    |  |  |  +---w target-fec-stack-type
    |  |  |  |  +---w target-fec-stack-type      identityref
```

```
|  |  |  |  +---w (target-fec-stack-value)?
|  |  |  |     +--:(ldp-ip-prefix)
|  |  |  |     |  +---w ldp-ip-prefix?       inet:ip-prefix
|  |  |  |     +--:(rsvp)
|  |  |  |     |  +---w tunnel-id?           uint32
|  |  |  |     +--:(vpn-ip-prefix)
|  |  |  |     |  +---w vrf-id?              uint32
|  |  |  |     |  +---w vpn-ip-prefix?       inet:ip-prefix
|  |  |  |     +--:(pw)
|  |  |  |     |  +---w pw-id?               uint32
|  |  |  |     |  +---w remote-pe-addr?      inet:ip-address
|  |  |  |     +--:(bgp-label-prefix)
|  |  |  |     |  +---w bgp-label-prefix?    inet:ip-prefix
|  |  |  |     +--:(generic-ip-prefix)
|  |  |  |     |  +---w generic-ip-prefix?   inet:ip-prefix
|  |  |  |     +--:(igp-ip-prefix)
|  |  |  |        +---w protocol?            identityref
|  |  |  |        +---w igp-ip-prefix?       inet:ip-prefix
|  |  |  +---w target-fec-type          target-fec-type
|  |  |  +---w reply-mode?              reply-mode
|  |  |  +---w return-ttl-tlv?          boolean
|  |  |  +---w return-ttl-value
|  |  |  |  +---w return-ttl-value?   uint8
|  |  |  +---w global-flags
|  |  |     +---w v-flag?   boolean
|  |  |     +---w t-flag?   boolean
|  |  |     +---w r-flag?   boolean
|  |  +---w echo-scheduling-parameters
|  |     +---w probe-interval
|  |     |  +---w min-probe-interval?   identityref
|  |     |  +---w max-probe-interval?   identityref
|  |     +---w probe-count?      uint32
|  |     +---w probe-timeout?    identityref
|  |     +---w output-info
|  |        +---w output-intf* [interface]
|  |        |  +---w interface   if:interface-ref
|  |        +---w nexthop?       inet:ip-address
|  +--ro output
|     +--ro response-list* [response-index]
|        +--ro response-index                 uint32
|        +--ro response-header-parameters
|        |  +--ro resp-source-address         ip-address
|        |  +--ro resp-destination-address    ip-address
|        |  +--ro resp-traffic-class          uint8
|        +--ro response-payload-parameters
|           +--ro reply-mode            reply-mode
|           +--ro return-code           return-code
|           +--ro return-sub-code       uint8
```

```
       |           +--ro seq-number            uint32
       |           +--ro timestamp-sent        yang:date-and-time
       |           +--ro timestamp-received    yang:date-and-time
       |           +--ro target-fec-type       target-fec-type
       +---x single-path-discovery
       |  +---w input
       |  |  +---w echo-header-parameters
       |  |  |  +---w source-address?        ip-address
       |  |  |  +---w destination-address?   ip-loopback-address
       |  |  |  +---w traffic-class?         mpls-traffic-class
       |  |  |  +---w mpls-entropy-label?    mpls-entropy-label
       |  |  |  +---w header-mpls-ttl?       uint8
       |  |  |  +---w mpls-exp-label?        boolean
       |  |  +---w echo-payload-parameters
       |  |  |  +---w reply-tos-tlv?           boolean
       |  |  |  +---w reply-tos-value
       |  |  |  |  +---w reply-tos-value?   uint8
       |  |  |  +---w probe-size?              uint32
       |  |  |  +---w probe-sweep
       |  |  |  |  +---w min-probe-sweep?   uint16
       |  |  |  |  +---w max-probe-sweep?   uint16
       |  |  |  +---w target-fec-stack-type
       |  |  |  |  +---w target-fec-stack-type      identityref
       |  |  |  |  +---w (target-fec-stack-value)?
       |  |  |  |     +--:(ldp-ip-prefix)
       |  |  |  |     |  +---w ldp-ip-prefix?       inet:ip-prefix
       |  |  |  |     +--:(rsvp)
       |  |  |  |     |  +---w tunnel-id?           uint32
       |  |  |  |     +--:(vpn-ip-prefix)
       |  |  |  |     |  +---w vrf-id?              uint32
       |  |  |  |     |  +---w vpn-ip-prefix?       inet:ip-prefix
       |  |  |  |     +--:(pw)
       |  |  |  |     |  +---w pw-id?               uint32
       |  |  |  |     |  +---w remote-pe-addr?      inet:ip-address
       |  |  |  |     +--:(bgp-label-prefix)
       |  |  |  |     |  +---w bgp-label-prefix?    inet:ip-prefix
       |  |  |  |     +--:(generic-ip-prefix)
       |  |  |  |     |  +---w generic-ip-prefix?   inet:ip-prefix
       |  |  |  |     +--:(igp-ip-prefix)
       |  |  |  |        +---w protocol?            identityref
       |  |  |  |        +---w igp-ip-prefix?       inet:ip-prefix
       |  |  |  +---w target-fec-type          target-fec-type
       |  |  |  +---w reply-mode?              reply-mode
       |  |  |  +---w return-ttl-tlv?          boolean
       |  |  |  +---w return-ttl-value
       |  |  |  |  +---w return-ttl-value?   uint8
       |  |  |  +---w global-flags
       |  |  |     +---w v-flag?   boolean
```

```
        |  |  |     +---w t-flag?   boolean
        |  |  |     +---w r-flag?   boolean
        |  |  +---w echo-scheduling-parameters
        |  |     +---w probe-interval
        |  |     |  +---w min-probe-interval?   identityref
        |  |     |  +---w max-probe-interval?   identityref
        |  |     +---w probe-count?      uint32
        |  |     +---w probe-timeout?    identityref
        |  |     +---w output-info
        |  |        +---w output-intf* [interface]
        |  |        |  +---w interface   if:interface-ref
        |  |        +---w nexthop?       inet:ip-address
        |  +--ro output
        |     +--ro response-list* [response-index]
        |        +--ro response-index                          uint32
        |        +--ro response-header-parameters
        |        |  +--ro resp-source-address         ip-address
        |        |  +--ro resp-destination-address    ip-address
        |        |  +--ro resp-traffic-class          uint8
        |        +--ro response-payload-parameters
        |        |  +--ro reply-mode            reply-mode
        |        |  +--ro return-code           return-code
        |        |  +--ro return-sub-code       uint8
        |        |  +--ro seq-number            uint32
        |        |  +--ro timestamp-sent        yang:date-and-time
        |        |  +--ro timestamp-received    yang:date-and-time
        |        |  +--ro target-fec-type       target-fec-type
        |        +--ro response-payload-optional-parameters
        |           +--ro ddmap
        |              +--ro ddmap-mtu?                     int16
        |              +--ro ddmap-downstream-address?      ip-address
        |              +--ro ddmap-downstream-intf-index?   int32
        |              +--ro ddmap-return-code?             return-code
        |              +--ro ddmap-return-subcode?          int16
        |              +--ro ddmap-label-stack* [label]
        |                 +--ro label       rt-types:mpls-label
        |                 +--ro protocol?   ddmap-protocol
        |
      +---x multi-path-discovery
         +---w input
         |  +---w echo-header-parameters
         |  |  +---w source-address?        ip-address
         |  |  +---w destination-address?   ip-loopback-address
         |  |  +---w traffic-class?         mpls-traffic-class
         |  |  +---w mpls-entropy-label?    mpls-entropy-label
         |  |  +---w header-mpls-ttl?       uint8
         |  |  +---w mpls-exp-label?        boolean
         |  +---w echo-payload-parameters
         |  |  +---w reply-tos-tlv?           boolean
```

```
          |  |  +---w reply-tos-value
          |  |  |  +---w reply-tos-value?   uint8
          |  |  +---w probe-size?              uint32
          |  |  +---w probe-sweep
          |  |  |  +---w min-probe-sweep?   uint16
          |  |  |  +---w max-probe-sweep?   uint16
          |  |  +---w target-fec-stack-type
          |  |  |  +---w target-fec-stack-type      identityref
          |  |  |  +---w (target-fec-stack-value)?
          |  |  |     +--:(ldp-ip-prefix)
          |  |  |     |  +---w ldp-ip-prefix?       inet:ip-prefix
          |  |  |     +--:(rsvp)
          |  |  |     |  +---w tunnel-id?           uint32
          |  |  |     +--:(vpn-ip-prefix)
          |  |  |     |  +---w vrf-id?              uint32
          |  |  |     |  +---w vpn-ip-prefix?       inet:ip-prefix
          |  |  |     +--:(pw)
          |  |  |     |  +---w pw-id?               uint32
          |  |  |     |  +---w remote-pe-addr?      inet:ip-address
          |  |  |     +--:(bgp-label-prefix)
          |  |  |     |  +---w bgp-label-prefix?    inet:ip-prefix
          |  |  |     +--:(generic-ip-prefix)
          |  |  |     |  +---w generic-ip-prefix?   inet:ip-prefix
          |  |  |     +--:(igp-ip-prefix)
          |  |  |        +---w protocol?            identityref
          |  |  |        +---w igp-ip-prefix?       inet:ip-prefix
          |  |  +---w target-fec-type          target-fec-type
          |  |  +---w reply-mode?              reply-mode
          |  |  +---w return-ttl-tlv?          boolean
          |  |  +---w return-ttl-value
          |  |  |  +---w return-ttl-value?   uint8
          |  |  +---w global-flags
          |  |     +---w v-flag?   boolean
          |  |     +---w t-flag?   boolean
          |  |     +---w r-flag?   boolean
          |  +---w echo-scheduling-parameters
          |  |  +---w probe-interval
          |  |  |  +---w min-probe-interval?   identityref
          |  |  |  +---w max-probe-interval?   identityref
          |  |  +---w probe-count?      uint32
          |  |  +---w probe-timeout?    identityref
          |  |  +---w output-info
          |  |     +---w output-intf* [interface]
          |  |     |  +---w interface    if:interface-ref
          |  |     +---w nexthop?       inet:ip-address
          |  +---w ddmap-hash
          |     +---w ddmap-hash?   multipath-hashtype
          +--ro output
```

```
            +--ro response-list* [response-index]
               +--ro response-index                          uint32
               +--ro response-header-parameters
               |  +--ro resp-source-address         ip-address
               |  +--ro resp-destination-address    ip-address
               |  +--ro resp-traffic-class          uint8
               +--ro response-payload-parameters
               |  +--ro reply-mode            reply-mode
               |  +--ro return-code           return-code
               |  +--ro return-sub-code       uint8
               |  +--ro seq-number            uint32
               |  +--ro timestamp-sent        yang:date-and-time
               |  +--ro timestamp-received    yang:date-and-time
               |  +--ro target-fec-type       target-fec-type
               +--ro response-payload-optional-parameters
                  +--ro ddmap
                     +--ro ddmap-mtu?                     int16
                     +--ro ddmap-downstream-address?      ip-address
                     +--ro ddmap-downstream-intf-index?   int32
                     +--ro ddmap-return-code?             return-code
                     +--ro ddmap-return-subcode?          int16
                     +--ro ddmap-label-stack* [label]
                        +--ro label       rt-types:mpls-label
                        +--ro protocol?   ddmap-protocol
```

3. LSP Ping YANG Module

```
<CODE BEGIN> file "ietf-mpls-lsp-ping@2020-06-09.yang"
module ietf-mpls-lsp-ping-rev1d {
yang-version 1.1;
namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-lsp-ping-rev1d";
prefix "lsp-ping";

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

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

import ietf-interfaces {
```

```
    prefix if;
    reference
    "RFC 8343: YANG Interface Management";
  }

  import ietf-lime-time-types {
    prefix lime;
  }

  import ietf-routing {
    prefix rt;
    reference
    "RFC 8022: YANG Routing Management";
  }

  import ietf-mpls {
    prefix mpls;
    reference
    "RFC 8960: YANG Data Model for MPLS Base";
  }

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

  organization
    "IETF MPLS Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/mpls/>
     WG List:  <mailto:mpls@ietf.org>

    Editor:   Nagendra Kumar Nainar
              <mailto:nagendrakumar.nainar@gmail.com>
    Editor:   Madhan Sankaranarayanan
              <mailto:madsanka@cisco.com>;
    Editor:   Jaganbabu Rajamanickam
              <mailto:jrajaman@cisco.com>;
    Editor:   Carlos Pignataro
              <mailto:cpignata@gmail.com>
              <mailto:cmpignat@ncsu.edu>";

  description
    "This YANG module defines the configuration of MPLS LSP Ping.
    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.
```

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

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

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

   revision 2020-06-09 {
     description
       "Initial revision.";
     reference
       "To be Updated";
   }

   feature mpls-oam {
     description
       "MPLS OAM feature.";
   }

   typedef ipv4-loopback-address {
     type inet:ipv4-address {
       pattern '127.*';
     }
     description
     "This type represents an IPv4 Loopback address,
     which is in the range of 127.0.0.0 to 127.255.255.255.";
     reference
     "RFC 1212: Requirements for Internet Hosts
     -- Communication Layers.";
   }

   typedef ipv6-loopback-address {
     type inet:ipv6-address {
       pattern 'to-be-updated';
     }
```

```
    description
    "This type represents an IPv6 Loopback address,
    which is ::1/128";
    reference
    "RFC 4291: IP Version 6 Addressing Architecture.";
  }

  typedef ip-loopback-address {
    type union {
      type ipv4-loopback-address;
      type ipv6-loopback-address;
    }
    description
      "This type represents a version-neutral IP Loopback
      address.";
  }

  typedef ip-address {
    type union {
      type inet:ipv4-address;
      type inet:ipv6-address;
    }
    description
      "Version neutral IP Address";
  }

  typedef mpls-traffic-class {
    type uint8 {
      range 0..7;
    }
    description
      "MPLS Traffic Class (EXP) value within range 0 to 7.";
  }

  typedef mpls-entropy-label {
    type rt-types:mpls-label-general-use;

    description
      "This type represents the Entropy Label,
      which is outside the reserved range.";
    reference
      "RFC 6790: The Use of Entropy Labels in MPLS Forwarding";
  }

  typedef multipath-hashtype {
    type enumeration {
      enum mp-empty {
        value "0";
```

```
          description
            "No Multipath";
      }
      enum mp-ip-addr {
        value "2";
        description
          "IP Address based Multipath
          Information Encoding";
      }
      enum mp-ip-range {
        value "4";
        description
          "IP Address range based Multipath
          Information Encoding";
      }
      enum mp-ip-bitmask {
        value "8";
        description
          "Bit masked IP Address set for Multipath
          Information Encoding";
      }
      enum mp-label-bitmask {
        value "9";
        description
          "Bit masked Label set for Multipath
          Information Encoding";
      }
    }
    description
      "This type represents the multipath Data type to be used in
      the DDMAP for Multipath tracing.";
    reference
      "RFC 8029: Section 3.4.1.1.";
    }

    typedef reply-mode {
      type enumeration {
        enum do-not-reply {
          value "1";
          description
            "Do not Reply";
          reference
            "RFC8029: Section 3";
        }
        enum reply-udp {
          value "2";
          description
            "Reply via an IPv4/IPv6 UDP packet";
```

```
        reference
          "RFC8029: Section 3";
      }
      enum reply-udp-ra {
        value "3";
        description
          "Reply via an IPv4/IPv6 UDP packet with
          Router Alert";
        reference
          "RFC8029: Section 3";
      }
      enum reply-app-cc {
        value "4";
        description
          "Reply via application-level control
          channel";
        reference
          "RFC8029: Section 3";
      }
      enum reply-via-path {
        value "5";
        description
          "Reply via Specified Path";
        reference
          "RFC7110: Section 4.1";
      }
    }
    description
      "This type defines the Reply mode";
    }

    typedef return-code {
      type enumeration {
        enum no-return {
          value "0";
          description
            "No Return Code";
          reference
            "RFC 8029: Section 3.1";
        }
        enum malformed-echo {
          value "1";
          description
            "Malformed Echo Request Received";
          reference
            "RFC 8029: Section 3.1";
        }
        enum unknown-tlvs {
```

```
          value "2";
          description
            "One or more of the TLVs was not
            understood";
          reference
            "RFC 8029: Section 3.1";
        }
        enum egress-reply {
          value "3";
          description
            "Replying router is an egress for the
            FEC at stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum egress-nomap {
          value "4";
          description
            "Replying router has no mapping for the
            FEC at stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum dd-mismatch {
          value "5";
          description
            "Downstream Mapping Mismatch";
          reference
            "RFC 8029: Section 3.1";
        }
        enum unknown-upstream {
          value "6";
          description
            "Upstream Interface Index Unknown";
          reference
            "RFC 8029: Section 3.1";
        }
        enum reserved {
          value "7";
          description
            "Reserved";
          reference
            "RFC 8029: Section 3.1";
        }
        enum label-switched {
          value "8";
          description
            "Label switched at stack-depth <RSC>";
```

```
          reference
            "RFC 8029: Section 3.1";
        }
        enum label-switched-no-mpls {
          value "9";
          description
            "Label switched but no MPLS forwarding
            at stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum FEC-map-mismatch {
          value "10";
          description
            "Mapping for this FEC is not the given
            label at stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum no-label {
          value "11";
          description
            "No label entry at stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum protocol-mismatch {
          value "12";
          description
            "Protocol not associated with interface
            at FEC stack-depth <RSC>";
          reference
            "RFC 8029: Section 3.1";
        }
        enum premature-terminate {
          value "13";
          description
            "Premature termination of ping due to
             label stack shrinking to a single label";
          reference
            "RFC 8029: Section 3.1";
        }
        enum ddmap-return-code {
          value "14";
          description
            "See DDMAP TLV for meaning of Return Code
            and Return Subcode";
          reference
```

```
            "RFC 8029: Section 3.1";
        }
        enum label-switched-fec-change {
          value "15";
          description
            "Label switched with FEC change";
          reference
            "RFC 8029: Section 3.1";
        }
      }
      description
        "This defines the Return codes received in the
        Echo Response.";
      reference
        "RFC 8029: Section 3.1";
    }

    typedef ddmap-protocol {
      type enumeration {
        enum unknown {
          value "0";
          description
            "Unknown Signaling Protocol";
        }
        enum static {
          value "1";
          description
            "Static Signaling Protocol";
        }
        enum bgp {
          value "2";
          description
            "BGP Signaling Protocol";
        }
        enum ldp {
          value "3";
          description
            "LDP Signaling Protocol";
        }
        enum rsvp-te {
          value "4";
          description
            "RSVP-TE Signaling Protocol";
        }
      }
      description
        "This defines the Signaling Protocol
        received in the DDMAP.";
```

```
    reference
      "RFC 8029: Section 3.4.1.2";
  }

  identity igp-proto {
    description
      "IGP Protocol.";
  }

  identity igp-any {
    base igp-proto;
    description
      "Any IGP Protocol. The value is set to
      0 in the TLV.";
    reference
      "RFC8287: Section 9.2";
  }

  identity igp-ospfv2 {
    base igp-proto;
    description
      "OSPFv2 IGP Protocol. The value is set to
      1 in the TLV.";
    reference
      "RFC8287: Section 9.2";
  }

  identity igp-ospfv3 {
    base igp-proto;
    description
      "OSPFv3 IGP Protocol. The value is set to
       3 in the TLV.";
    reference
      "RFC8287: Section 9.2";
  }

  identity igp-isis {
    base igp-proto;
    description
      "ISIS IGP Protocol. The value is set to
      2 in the TLV.";
    reference
      "RFC8287: Section 9.2";
  }

  identity target-fec-type {
    description
    "Target FEC Stack TLV Type";
```

```
   }

   identity ldp-ip-prefix {
     base target-fec-type;
     description
       "LDP IPv4/IPv6 Prefix.";
   }

   identity rsvp {
     base target-fec-type;
     description
       "RSVP IPv4/IPv6 LSP.";
   }

   identity vpn-ip-prefix {
     base target-fec-type;
     description
       "VPN IPv4/IPv6 Prefix.";
   }

   identity pw {
     base target-fec-type;
     description
       "FEC 129 pseudowire IPv4/IPv6.";
   }

   identity bgp-label-prefix {
     base target-fec-type;
     description
       "BGP labeled IPv4/IPv6 Prefix.";
   }

   identity generic-ip-prefix {
     base target-fec-type;
     description
       "Generic IPv4/IPv6 Prefix.";
   }

   identity nil-fec {
     base target-fec-type;
     description
       "Nil FEC TLV.";
   }

   identity igp-ip-prefix {
     base target-fec-type;
     description
       "IGP IPv4/IPv6 Prefix Segment ID.";
```

```
   }

   identity igp-adj-prefix {
     base target-fec-type;
     description
       "IGP Adjacency Segment ID.";
   }

   grouping global-flags {
     container global-flags {
       leaf v-flag {
         type boolean;
         default false;
         description
           "Section 3 of RFC8029 - The V (Validate
           FEC Stack) flag is used if the
           FEC stack should be validated";
       }
       leaf t-flag {
         type boolean;
         default false;
         description
           "Section 3 of RFC8029 - The T flag
           is set if the response is expected
           only if TTL expires";
       }
       leaf r-flag {
         type boolean;
         default false;
         description
           "Section 3 of RFC8029 - The R flag
           is set if the responder should
           return the reverse-path FEC information.";
       }
     }
   }

   grouping echo-header-parameters {
     container echo-header-parameters {
       leaf source-address {
         type ip-address;
         description
           "Specifies the Source IP address in
           the Echo Request header.";
       }
       leaf destination-address {
         type ip-loopback-address;
         description
```

```
            "Specifies the Destination IP address
            in the Echo Request header.";
        }
        leaf traffic-class {
          type mpls-traffic-class;
          description
            "Specifies the MPLS traffic class
            in the Echo Request header.";
        }
        leaf mpls-entropy-label {
          type mpls-entropy-label;
          description
            "Specifies the Entropy Label to be
            inserted along with the Label Stack
            for the Echo Request header.";
        }
        leaf header-mpls-ttl {
          type uint8;
          default 255;
          description
            "Specifies the TTL value for the
            MPLS Label in the Echo Request header.";
        }
        leaf mpls-exp-label {
          type boolean;
          description
            "This optional attribute is used to
            force the insertion of MPLS Explicit
            Null in the Label Stack for the Echo
            Request header.";
        }
      }
    }

    grouping echo-payload-parameters {
      container echo-payload-parameters {
        leaf reply-tos-tlv {
          type boolean;
          default false;
          description
            "This optional attribute is used to
            instruct the Initiator to include
            Reply-TOS TLV.";
        }//leaf reply-tos-tlv
        container reply-tos-value {
          when "../reply-tos-tlv = 'true'" {
          description
            "Reply TOS Value is set ONLY
```

```
            when Reply-TOS TLV is required
            in the Echo Request.";
          }
          leaf reply-tos-value {
            type uint8 {
              range "0..63";
              }
            description
              "TOS value for the return packet.";
           }//leaf ddmap-hash
        }//container reply-tos-value
        leaf probe-size {
          type uint32 {
            range 1..15000;
          }
        }//leaf probe-size
        container probe-sweep {
          when "../probe-size = 'false'" {
            description
              "Probe sweep should be used only if a fixed
              probe size is not defined.";
          }
          leaf min-probe-sweep {
            type uint16 {
              range 72..18000;
            }
            description
              "This define the minimum size of
              the probe packet.";
            }
            leaf max-probe-sweep {
              type uint16 {
                range "72..18000";
              }
              description
                "This defines the maximum size of
                the probe packet.";
            }
            description
              "This instructs the initiator to send
              a sweep of probe packets at varying size
              between the minimum and maximum value
              defined.";
         }
         container target-fec-stack-type {
          leaf target-fec-stack-type {
            type identityref {
              base target-fec-type;
```

```
          }
          mandatory true;
            description
              "Target FEC STack to define the FEC to
              be included in the Echo Request";
        }

        choice target-fec-stack-value {
          description
            "Target FEC Stack Value";
          case ldp-ip-prefix {
            leaf ldp-ip-prefix {
              type inet:ip-prefix;
              description
                "LDP IPv4/IPv6 Prefix.";
            }
          }
          case rsvp {
            leaf tunnel-id {
              type uint32;
              description
                "RSVP Tunnel ID.";
            }
          }
          case vpn-ip-prefix {
            leaf vrf-id {
              type uint32;
              description
                "VPN ID.";
            }
            leaf vpn-ip-prefix {
              type inet:ip-prefix;
              description
                "VPN IP Prefix";
            }
          }
          case pw {
            leaf pw-id {
              type uint32;
              description
                "Pseudowire ID.";
            }
            leaf remote-pe-addr {
              type inet:ip-address;
              description
                "PW Remote PE Address.";
            }
          }
```

```
            case bgp-label-prefix {
              leaf bgp-label-prefix {
                type inet:ip-prefix;
                description
                  "BGP IPv4/IPv6 Prefix.";
              }
            }
            case generic-ip-prefix {
              leaf generic-ip-prefix {
                type inet:ip-prefix;
                description
                  "Generic IPv4/IPv6 Prefix.";
              }
            }
            case igp-ip-prefix {
              leaf protocol {
                type identityref {
                  base igp-proto;
                }
              description
                "IGP Protocol ID.";
              }
              leaf igp-ip-prefix {
                type inet:ip-prefix;
                description
                  "IGP IPv4/IPv6 Prefix.";
              }
            }
          }
        }
        leaf reply-mode {
          type reply-mode;
            description
              "XYZ";
            reference
              "RFC 8029: Section 3.";
        }

        leaf return-ttl-tlv {
          type boolean;
          default "false";
            description
              "'Time to Live' TLV to be included
              in the Echo Response.";
            reference
              "RFC 7394: Section 3.";
        }
```

```
      container return-ttl-value {
        when "../return-ttl-tlv = 'true'" {
          description
            "When TTL TLV is included in the
            Echo Request, the value mentioned
            in this field should be included in
            the value field of the TLV.";
          }
        leaf return-ttl-value {
          type uint8;
            description
              "Return TTL value";
        }//leaf return-ttl-value
      }//container return-ttl-value

      uses global-flags;

    }//container echo-payload-parameters
  }//grouping echo-payload-parameters

  grouping echo-scheduling-parameters {
    container echo-scheduling-parameters {
      container probe-interval {
        leaf min-probe-interval {
          type identityref {
            base lime:time-unit-type;
          }
          default "lime:milliseconds";
          description
            "This defines the minimum leaf interval.";
        }
        leaf max-probe-interval {
          type identityref {
            base lime:time-unit-type;
          }
          default "lime:milliseconds";
          description
            "This defines the maximum leaf interval.";
        }
        description
          "To be Added.";
      }//container probe-interval

      leaf probe-count {
        type uint32;
        default "5";
        description
          "This defines the number of probe counts.";
```

```
      }

      leaf probe-timeout {
        type identityref {
            base lime:time-unit-type;
          }
          default "lime:seconds";
          //range "0..3600";
          description
            "This defines the probe timeout
            interval in Seconds.";
      }

      container output-info {
        list output-intf {
          key "interface";
          leaf interface {
            type if:interface-ref;
            description
              "Specifies the Egress interface to
            send the probe out.";
          }
          description
            "List of outgoing interfaces";
        }

        leaf nexthop {
          type inet:ip-address;
          description
            "Specifies the next hop address to
            send the probe out.";
        }
      }
    }//container echo-scheduling-parameters
  }//grouping echo-scheduling-parameters

  grouping response-header-parameters {
    container response-header-parameters {
      leaf resp-source-address {
        type ip-address;
        mandatory true;
        description
          "Specifies the Source IP address in
          the Echo Response header.";
      }

      leaf resp-destination-address {
        type ip-address;
```

```
        mandatory true;
        description
          "Specifies the Destination IP address in
          the Echo Response header.";
      }

      leaf resp-traffic-class {
        type uint8 {
          range "0..63";
        }
        mandatory true;
        description
          "Specifies the TOS/DSCP in the Echo
          Response header.";
      }

    }//container response-header
  }//grouping response-header

  grouping response-payload-parameters {
    container response-payload-parameters {
      leaf reply-mode {
        type reply-mode;
        mandatory true;
        description
          "XYZ";
        reference
          "RFC 8029: Section 3.";
      }

      leaf return-code {
        type return-code;
        mandatory true;
        description
          "Return Code received in the Echo
          Reply Payload.";
      }

      leaf return-sub-code {
        type uint8;
        mandatory true;
        description
          "Return Sub Code received in the Echo
          Reply Payload.";
      }

      leaf seq-number {
        type uint32;
```

```
          mandatory true;
          description
            "Sequence Number received in the Echo
            Reply Payload.";
        }

        leaf timestamp-sent {
          type yang:date-and-time;
          mandatory true;
          description
            "Timestamp Sent is the time of day in
            64-bit NTP timestamp format
            when MPLS Echo Request is sent.";
        }

        leaf timestamp-received {
          type yang:date-and-time;
          mandatory true;
          description
            "Timestamp Received is the time of day in
            64-bit NTP timestamp format
            when MPLS Echo Response is sent.";
        }

        leaf target-fec-type {
          type target-fec-type;
          mandatory true;
            description
              "Target FEC Stack to define the FEC to be
              included in the Echo Request.";
        }
      }//container response-payload-parameters

    }//grouping response-payload-parameters

    grouping response-payload-optional-parameters {
      container response-payload-optional-parameters {
          container ddmap {
              leaf ddmap-mtu {
                  type int16;
                  description
                      "This is used to carry the MTU from the DDMAP
                      received in teh Echo Response Payload.";
              }

              leaf ddmap-downstream-address {
                  type ip-address;
                  description
```

```
                    "This is used to carry the Downstream
                    Address from the DDMAP received in the
                    Echo Response Payload.";
            }

            leaf ddmap-downstream-intf-index {
                type int32 {
                    range "1..2147483647";
                }
                description
                    "This is used to carry the Downstream
                    Interface Address from the DDMAP
                    received in the Echo Response Payload.";
            }

            leaf ddmap-return-code {
                type return-code;
                description
                    "This is used to carry the Return Code
                    from the DDMAP received in the Echo
                    Response Payload.";
            }

            leaf ddmap-return-subcode {
                type int16;
                description
                    "This is used to carry the Return Sub Code
                        from the DDMAP received in the Echo
                        Response Payload.";
            }

            list ddmap-label-stack {
                key "label";
                description
                    "This is used to carry the Label Stack
                    from the DDMAP received in the Echo
                    Response Payload.";
                leaf label {
                    type rt-types:mpls-label;
                }

                leaf protocol {
                    type ddmap-protocol;
                }
            }
        }//container ddmap
    } //response-payload-optional-parameters
    reference
```

```
        "RFC 8029: Section 3.4 -- DDMAP.";
    } //grouping resp-payload-optional-parameters

    /* Configuration */

    augment "/rt:routing/mpls:mpls" {
        if-feature mpls-oam;
        description
          "RFC8029: MPLS OAM Feature Augmentation";
        container mpls-oam {
          leaf enable {
            type boolean;
            description
              "Enable MPLS OAM";
          }
        }
    }

    /* RPC */

    rpc continuity-check {
        description
          "Triggers LSP Ping from the Initiator and return
          the response.";

        input {
          uses echo-header-parameters {
            description
              "This grouping defines the parameters
              to be set in the probe header.";
          }
          uses echo-payload-parameters {
            description
              "This grouping defines the parameters
              to be set in the probe payload.";
          }
          uses echo-scheduling-parameters {
            description
              "This grouping defines the scheduling
              parameters to be used by the initiator.";
          }
        }

        output {
          list response-list {
            key "response-index";
            description
              "Continuity Check Response List.";
```

```
          leaf response-index {
            type uint32;
            mandatory true;
          }
          uses response-header-parameters {
            description
              "This grouping defines the parameters
              from the received echo response header.";
          }

          uses response-payload-parameters {
            description
              "This grouping defines the parameters
              from the received echo response payload.";
          }
        }//list response-list
      }//output
   }//rpc continuity-check

   rpc single-path-discovery {
       description
         "Triggers LSP trace from the Initiator and return
         the response.";

       input {
         uses echo-header-parameters {
           description
             "This grouping defines the parameters to be
             set in the probe header.";
         }
         uses echo-payload-parameters {
           description
             "This grouping defines the parameters to be
             set in the probe payload.";
         }
         uses echo-scheduling-parameters {
           description
             "This grouping defines the scheduling
             parameters to be used by the initiator.";
         }
       }
       output {
         list response-list {
           key "response-index";
           description
             "Single Path Discovery Response List.";
           leaf response-index {
             type uint32;
```

```
            mandatory true;
          }
          uses response-header-parameters {
            description
              "This grouping defines the parameters
              from the received echo response header.";
          }

          uses response-payload-parameters {
            description
              "This grouping defines the parameters
              from the received echo response payload.";
          }

          uses response-payload-optional-parameters {
            description
              "This grouping defines the optional
              parameters from the received Echo
              Response.";
          }

        }//list response-list
      }//output
   }//rpc single-path-discovery

   rpc multi-path-discovery {
     description
       "Triggers LSP trace with ddmap-hash from the Initiator and return
       the response.";

     input {
       uses echo-header-parameters {
         description
           "This grouping defines the parameters to be
           set in the probe header.";
       }
       uses echo-payload-parameters {
         description
           "This grouping defines the parameters to be
           set in the probe payload.";
       }
       uses echo-scheduling-parameters {
         description
           "This grouping defines the scheduling
           parameters to be used by the initiator.";
       }
       container ddmap-hash {
         leaf ddmap-hash {
```

```
            type multipath-hashtype;
              description
                "Hashkey type for the DDMAP";
          }//leaf ddmap-hash
        }
      }
      output {
        list response-list {
          key "response-index";
          description
            "Multi Path Discovery Response List.";
          leaf response-index {
            type uint32;
            mandatory true;
          }
          uses response-header-parameters {
            description
              "This grouping defines the parameters
              from the received echo response header.";
          }

          uses response-payload-parameters {
            description
              "This grouping defines the parameters
              from the received echo response payload.";
          }

          uses response-payload-optional-parameters {
            description
              "This grouping defines the optional
              parameters from the received Echo
              Response.";
          }

        }//list response-list
      }//output
    }//rpc multi-path-discovery
    }

    <CODE ENDS>
```

4. IANA Considerations

To be Updated.

5. Security Considerations

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

The NETCONF Configuration Access Control Model (NACM) [RFC8341] [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.

This module defines RPC operations that may be considered sensitive or vulnerable in some network environments. It is therefore important to control the access to these operations. These are the operations and their sensitivity/vulnerability:

* Continuity Check: Generates OAM probe for continuity check and retrieves the resulting data from the Initiator node.

* Single Path Discovery: Generates OAM probe for single path tracing and retrieves the resulting data from various transit nodes.

* Multi Path Discovery: Generates OAM probe for equal cost multipath tracing and retrieves the resulting data from various transit nodes.

These operations are used to retrieve the data from the device that needs to execute the OAM command. Unauthorized source access to some sensitive information in the above data may be used for network reconnaissance or lead to denial-of-service attacks on both the local device and the network.

6. Acknowledgement

Some part of the YANG model was inspired by the previous model developed by Lianshu Zheng, Guangying Zheng, Greg Mirsky, Reshad Rahman, Faisal Iqbal and the authors would like to acknowledge them.

The authors also would like to thank Loa Andersson for his help with forming this team and help organizing the work.

7. Contributors

The following are key contributors to this document:

Reshad Rahman, Cisco Systems, Inc.

Zafar Ali, Cisco Systems, Inc.

8. References

8.1. Normative References

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

[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>.

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

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <https://www.rfc-editor.org/info/rfc8029>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <https://www.rfc-editor.org/info/rfc8287>.

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

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

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

[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>.

8.2. Informative References

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

Authors' Addresses

Nagendra Kumar Nainar
NVIDIA
Research Triangle Park, NC 27709
United States of America
Email: nagendrakumar.nainar@gmail.com

Madhan Sankaranarayanan
Cisco Systems, Inc.
Manjanam Sansel, Kurunji Nagar
Chinnamanur 625515
Tamil Nadu
India

Email: madsanka@cisco.com

Jaganbabu Rajamanickam (editor)
Cisco Systems, Inc.
Canada
Email: jrajaman@cisco.com

Carlos Pignataro
North Carolina State University
United States of America
Email: cpignata@gmail.com, cmpignat@ncsu.edu

Walker Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: zhengguangying@huawei.com