A YANG Data Model for MPLS Static LSPs

draft-ietf-mpls-static-yang-05

Abstract

This document contains the specification for the MPLS Static Label Switched Paths (LSPs) YANG model. The model allows for the provisioning of static LSP(s) on LER(s) and LSR(s) devices along a LSP path without the dependency on any signaling protocol. The MPLS Static LSP model augments the MPLS base YANG model with specific data to configure and manage MPLS Static LSP(s).

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 August 19, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of
The MPLS Static LSP YANG model is defined in module "ietf-mpls-static" and augments the MPLS Base YANG model defined in module "ietf-mpls" in [I-D.saad-mpls-static-yang].
1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

The following terms are defined in [RFC6020]:

- augment,
- configuration data,
- data model,
- data node,
- feature,
- mandatory node,
- module,
- schema tree,
- state data,
- RPC operation.

1.2. Model Organization

The base MPLS Static LSP model covers the core features with the minimal set of configuration parameters needed to manage and operate MPLS Static LSPs.

Additional MPLS Static LSP parameters as well as optional feature(s) are grouped in a separate MPLS Static LSP extended model. The relationship between the MPLS base and other MPLS modules are shown in Figure 1.
1.3. MPLS Static LSPs Model Tree Diagram

The MPLS Static and extendend LSP tree diagram is shown in Figure 2.

module: ietf-mpls-static
augment /rt:routing/mpls:mpls:
  +++-rw static-lsps
    +++-rw static-lsp* [name]
      |  +++-rw name         -> ../config/name
      |  +++-rw config
      |  |  +++-rw name?      string
      |  |  +++-rw operation?  mpls-operations-type
      |  +++-ro state
      |  |  +++-ro name?      string
      |  |  +++-ro operation?  mpls-operations-type
      |  +++-rw (out-segment)?
      |     +++:(simple-path)
      |     |  +++-rw simple-path
      |     |     +++-rw config
      |     |     |  |  +++-rw next-hop?    inet:ip-address
      |     |     |  |  +++-rw outgoing-label? rt-types:mpls-label
      |     |     |  |  +++-rw outgoing-interface? if:interface-ref

Figure 1: Relationship between MPLS modules
+++ro state
  +++ro next-hop? inet:ip-address
  +++ro outgoing-label? rt-types:mpls-label
  +++ro outgoing-interface? if:interface-ref
+++:(multiple-paths)
  +++rw paths
    +++rw path* [path-index]
      +++rw path-index -> ../config/path-index
      +++rw config
        +++rw path-index? uint16
        +++rw backup-path-index? uint16
        +++rw next-hop? inet:ip-address
        +++rw outgoing-interface? if:interface-ref
        +++rw loadshare? uint16
        +++rw role? enumeration
      +++ro state
        +++ro path-index? uint16
        +++ro backup-path-index? uint16
        +++ro next-hop? inet:ip-address
        +++ro outgoing-interface? if:interface-ref
        +++ro loadshare? uint16
        +++ro role? enumeration
    +++rw outgoing-labels
      +++rw outgoing-labels* [index]
        +++rw index -> ../config/index
        +++rw config
          +++rw index? uint8
          +++rw label? rt-types:mpls-label
        +++ro state
          +++ro index? uint8
          +++ro label? rt-types:mpls-label
        +++rw mpls-static-ext:bandwidth? uint32
        +++rw mpls-static-ext:lsp-priority-setup? uint8
        +++rw mpls-static-ext:lsp-priority-hold? uint8
module: ietf-mpls-static-extended
augment /rt:routing/mpls:mpls:
  +++rw bidir-static-lsps
    +++rw bidir-static-lsp* [name]
      +++rw name string
      +++rw config
        +++rw forward-lsp? mpls-static:static-lsp-ref
        +++rw reverse-lsp? mpls-static:static-lsp-ref
      +++ro state
        +++ro forward-lsp? mpls-static:static-lsp-ref
        +++ro reverse-lsp? mpls-static:static-lsp-ref

Figure 2: MPLS Static LSP tree diagram
1.4. MPLS Static LSP YANG Module(s)

The MPLS Static LSP module is shown in Figure 3.

```yang
<CODE BEGINS> file "ietf-mpls-static@2017-07-02.yang"
module ietf-mpls-static {
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-static";
    prefix "mpls-static";
    import ietf-mpls {
        prefix mpls;
    }
    import ietf-routing {
        prefix "rt";
    }
    import ietf-routing-types {
        prefix "rt-types";
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix "if";
    }
    /* Import TE generic types */
    import ietf-te {
        prefix te;
    }
    organization "IETF MPLS Working Group";
    contact
        "WG Web: <http://tools.ietf.org/wg/mpls/>
        WG List: <mailto:mls@ietf.org>
        WG Chair: Loa Andersson
            <mailto:loa@pi.nu>
        WG Chair: Ross Callon
            <mailto:rcallon@juniper.net>
```
description
"This YANG module augments the 'ietf-routing' module with basic configuration and operational state data for MPLS static";

revision "2017-07-02" {
    description
    "Latest revision:
    - Addressed MPLS-RT review comments";
    reference "RFC 3031: A YANG Data Model for Static MPLS LSPs";
}

typedef static-lsp-ref {
    type leafref {
        path "/rt:routing/mpls:mpls/mpls-static:static-lsps/" + 
            "mpls-static:static-lsp/mpls-static:name";
    }
}
typedef mpls-operations-type {
  type enumeration {
    enum impose-and-forward {
      description
      "Operation impose outgoing label(s) and forward to
      next-hop";
    }
    enum pop-and-forward {
      description
      "Operation pop incoming label and forward to next-hop";
    }
    enum pop-impose-and-forward {
      description
      "Operation pop incoming label, impose one or more
      outgoing label(s) and forward to next-hop";
    }
    enum swap-and-forward {
      description
      "Operation swap incoming label, with outgoing label and
      forward to next-hop";
    }
    enum pop-and-lookup {
      description
      "Operation pop incoming label and perform a lookup";
    }
  }
}
description "MPLS operations types";

grouping path-basic_config {
  description "common definitions for statics";

  leaf next-hop {
    type inet:ip-address;
    description "next hop IP address for the LSP";
  }

  leaf outgoing-label {
    type rt-types:mpls-label;
    description
    "label value to push at the current hop for the
    LSP";
  }
}
leaf outgoing-interface {
  type if:interface-ref;
  description
    "The outgoing interface";
}

grouping path-outgoing-labels_config {
  description "Path outgoing labels grouping";
  leaf index {
    type uint8 {
      range "0..255";
    }
    description
      "Index of the label. Index 0 indicates top of the label stack";
  }
  leaf label {
    type rt-types:mpls-label;
    description
      "The outgoing MPLS labels to impose";
  }
}

grouping path-outgoing-labels {
  description "Path outgoing labels grouping";
  container outgoing-labels {
    description "List of outgoing labels";
    list outgoing-labels {
      key "index";
      description "Outgoing label list";
      leaf index {
        type leafref {
          path "../config/index";
        }
        description
          "Index of the label. Index 0 indicates top of the label stack";
      }
      container config {
        description
          "Configuration intended parameters";
        uses path-outgoing-labels_config;
      }
      container state {
        config false;
      }
  }
}
description
  "Configuration applied parameters and state";
uses path-outgoing-labels_config;
}
}
}

grouping path-properties_config {
  description
    "MPLS path properties";
  leaf path-index {
    type uint16;
    description
      "Path identifier";
  }
  leaf backup-path-index {
    type uint16;
    description
      "Backup path identifier";
  }
  leaf next-hop {
    type inet:ip-address;
    description
      "The address of the next-hop";
  }
  leaf outgoing-interface {
    type if:interface-ref;
    description
      "The outgoing interface";
  }
  leaf loadshare {
    type uint16;
    description
      "This value is used to compute a loadshare to perform un-equal
      load balancing when multiple outgoing path(s) are specified. A
      share is computed as a ratio of this number to the total under
      all configured path(s).";
  }
  leaf role {
    type enumeration {
      enum PRIMARY {
        description
          "";
      }
    }
  }
}


"Path as primary traffic carrying";
}
enum BACKUP {
    description
    "Path acts as backup";
}
enum PRIMARY_AND_BACKUP {
    description
    "Path acts as primary and backup simultaneously";
}
}

description
    "The MPLS path role";
}


grouping static-lsp-paths {
    description "Static LSP path grouping";
    choice out-segment {
        description "The MPLS out-segment type choice";
        case simple-path {
            container simple-path {
                description "Simple path container";
                container config {
                    description
                    "Holds the intended configuration";
                    uses path-basic_config;
                }
                container state {
                    config false;
                    description
                    "Holds the state and inuse configuration";
                    uses path-basic_config;
                }
            }
        }
        case multiple-paths {
            container paths {
                description "List of outgoing paths";
                list path {
                    key path-index;
                    description
                    "The list of MPLS paths associated with the FEC";
                    leaf path-index {
                        type leafref {
                            path ".../config/path-index";
                        }
                    description "Index of the path";
                }
            }
        }
    }
}
container config {
    description "Holds the intended configuration";
    uses path-properties_config;
}

container state {
    config false;
    description "Holds the state and inuse configuration";
    uses path-properties_config;
}

uses path-outgoing-labels;
}

grouping in-segment_config {
    description "In-segment grouping";
    choice type {
        description "Basic FEC choice";
        case ip-prefix {
            leaf ip-prefix {
                type inet:ip-prefix;
                description "An IP prefix";
            }
        }
        case mpls-label {
            leaf incoming-label {
                type rt-types:mpls-label;
                description "label value on the incoming packet";
            }
        }
        case tunnel {
            leaf tunnel {
                type te:tunnel-ref;
                description "TE tunnel FEC mapping";
            }
        }
    }
    leaf incoming-interface {
        type if:interface-ref;
        description "Optional incoming interface if FEC is restricted to traffic incoming on a specific interface";
    }
}
grouping in-segment {
  description "In-segment grouping";
  container in-segment {
    description "MPLS incoming segment";
    container config {
      description "Holds the intended configuration";
      uses in-segment_config;
    }
    container state {
      config false;
      description "Holds the state and inuse configuration";
      uses in-segment_config;
    }
  }
}

grouping static-lsp-top_config {
  description "Static LSP configuration grouping";
  leaf name {
    type string;
    description "name to identify the LSP";
  }
  leaf operation {
    type mpls-operations-type;
    description "The MPLS operation to be executed on the incoming packet";
  }
}

grouping static-lsp-top {
  description "common definitions for static LSPs";
  container config {
    description "Holds the intended configuration";
    uses static-lsp-top_config;
  }
  container state {
    config false;
    description "Holds the state and inuse configuration";
    uses static-lsp-top_config;
  }
}
augment "/rt:routing/mpls:mpls" {
  description "Augmentations for MPLS Static LSPs";
  container static-lsps {
    description "Statically configured LSPs, without dynamic signaling";
    list static-lsp {
      key name;
      description "list of defined static LSPs";
      leaf name {
        type leafref {
          path "../config/name";
        }
        description "name to identify the LSP";
      }
      uses static-lsp-top;
      uses static-lsp-paths;
    }
  }
}

<CODE ENDS>

Figure 3: MPLS Static LSP YANG module

The extended MPLS Static LSP module is shown in Figure 4.

<CODE BEGINS> file "ietf-mpls-static-extended@2017-07-02.yang"
module ietf-mpls-static-extended {
  prefix "mpls-static-ext";
  import ietf-mpls {
    prefix "mpls";
  }
  import ietf-routing {
    prefix "rt";
  }
  import ietf-mpls-static {
    prefix "mpls-static";
  }
  organization "IETF MPLS Working Group";

This module contains the Extended MPLS YANG data model.

revision 2017-03-10 {
description "Latest revision of MPLS extended yang module.";
reference "RFC2205";
}

/* RSVP features */
feature bandwidth {
  description
    "Indicates support for static LSP bandwidth allocation";
}

grouping static-lsp-extended_config {
  description
    "Configuration parameters for MPLS extended parameters";
  leaf bandwidth {
    type uint32;
    description
      "bandwidth in Mbps, e.g., using offline calculation";
  }
  leaf lsp-priority-setup {
    type uint8 {
      range "0..7";
    }
    description "LSP setup priority";
  }
  leaf lsp-priority-hold {
    type uint8 {
      range "0..7";
    }
    description "LSP hold priority";
  }
}

grouping bidir-static-lsp_config {
  description "common definitions for static LSPs";
  leaf forward-lsp {
    type mpls-static:static-lsp-ref;
    description
      "Reference to a configured static forward LSP";
  }
  leaf reverse-lsp {
    type mpls-static:static-lsp-ref;
    description
      "Reference to a configured static reverse LSP";
  }
}

grouping bidir-static-lsp {

description "grouping for top level list of static LSPs";
container config {
  description
  "Holds the intended configuration";
  uses bidir-static-lsp_config;
}
container state {
  config false;
  description
  "Holds the state and inuse configuration";
  uses bidir-static-lsp_config;
}

augment "/rt:routing/mpls:mpls/mpls-static:static-lsps" {
  description
  "RSVP signaling all interfaces configuration extensions";
  uses static-lsp-extended_config;
}

augment "/rt:routing/mpls:mpls" {
  description "Augmentations for MPLS Static LSPs";
  container bidir-static-lsps {
    description
    "Statically configured LSPs, without dynamic signaling";
    list bidir-static-lsp {
      key name;
      description "list of defined static LSPs";

      leaf name {
        type string;
        description "name to identify the LSP";
      }
      uses bidir-static-lsp;
    }
  }
}

<CODE ENDS>

Figure 4: Extended MPLS Static LSP YANG module

2. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.
This document registers two YANG modules in the YANG Module Names registry [RFC6020].


3. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

4. Normative References


Authors’ Addresses

Tarek Saad
Cisco Systems, Inc.
Email: tsaad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Email: skraza@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.
Email: rgandhi@cisco.com

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com
Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net