Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2021

H. Bidgoli, Ed.
Nokia
D. Voyer
Bell Canada
R. Parekh
Cisco System
T. Saad
Juniper Networks
T. Kundu
Nokia

October 28, 2020

YANG Data Model for p2mp sr policy
draft-hb-spring-sr-p2mp-policy-yang-02

Abstract

SR P2MP policies are set of policies that enable architecture for P2MP service delivery.

This document defines a YANG data model for SR P2MP Policy Configuration and operation.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 1, 2021.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG data model for P2MP SR Policy configuration and operation.

2. Conventions used in this document

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

3. Design of the Data Model

```
submodule: router-p2mp-traffic-engineering (belongs-to root)
  +--rw p2mp-traffic-engiineering!
     +--rw p2mp-policy* [root-address tree-id]
     |  +--rw root-address        inet:ip-address
     |  +--rw tree-id             uint32
     |  +--rw p2mp-policy-name?   string
     |  +--rw admin-state?        enumeration
     |  +--ro oper-state?         enumeration
     |  +--rw leaf-list* [leaf-address]
     |  |  +--rw leaf-address    inet:ip-address
     |  |  +--rw admin-state?    enumeration
     |  +--rw candidate-path* [protocol-id originator discriminator]
```

```
|     +--rw protocol-id            enumeration
|     +--rw originator             inet:ip-address
|     +--rw discriminator          uint32
|     +--rw candidate-path-name?   string
|     +--rw admin-state?           enumeration
|     +--ro oper-state?            enumeration
|     +--rw preference?            uint32
|     +--rw constraints* [index]
|     |  +--rw index         uint32
|     |  +--rw attributes?   uint32
|     +--rw explicit-routing* [index]
|     |  +--rw index         uint32
|     |  +--rw attributes?   uint32
|     +--rw path-instances* [index]
|        +--rw index          uint32
|        +--rw instance-id?   -> ../../../../replication-segment
                                 /replication-id
|        +--ro oper-state?    enumeration
+--rw replication-segment* [node-address replication-id]
   +--rw node-address        inet:ipv4-address
   +--rw replication-id      uint32
   +--rw admin-state?        enumeration
   +--ro oper-state?         enumeration
   +--rw root-address?   inet:ipv4-address
   +--rw tree-id?        uint32
   +--rw instance-id?    uint32
   +--rw replication-sid?    uint32
   +--rw downstream-nodes* [downstream-index]
      +--rw downstream-index           uint32
      +--rw next-hop-address?          inet:ip-address
      +--rw next-hop-interface-name?   if:interface-ref
      +--rw protecting-next-hop?       boolean
      +--rw protect-nexthop-id?        uint32
      +--rw (label)?
         +--:(sid-list)
         |  +--rw sid-list* [index]
         |     +--rw index               uint32
         |     +--rw sid-segment-type?   uint32
         +--:(sr-policy)
         |  +--rw sr-policy* [replication-sid]
         |     +--rw replication-sid    uint32
         |     +--rw sr-policy?         string
         +--:(rsvp-te)
            +--rw rsvp-te* [replication-sid]
               +--rw replication-sid      uint32
               +--rw rsvp-te-tunnel-id?   uint32
```

4. Configuration

This Module augments the "/rt:routing:" with a treeSID container. This container defines all the configuration parameter related to treeSID and P2MP SR Policy for this particular routing. The P2MP SR policy contains replication policy which in order contain candidate-path and the next-hop-groups for each OIF in the replication Policy. IT should be noted that two disjoint replication policies can be connected via a SR Policy as per [draft-ietf-spring-sr-replication-segment].

5. Yang Data Model

```
<CODE BEGINS> file "ietf-p2mp-policy-@2019-07-04.yang"
  module ietf-tree-sid {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:p2mp-policy-segment";
     // replace with IANA namespace when assigned
     prefix tree-sid;

     import ietf-inet-types {
       prefix "inet";
     }

     import ietf-yang-types {
       prefix "yang";
     }

     import ietf-routing-types {
       prefix "rt-types";
     }

     import ietf-routing {
       prefix "rt";
     }

     import ietf-interfaces {
       prefix "if";
     }

     import ietf-ip {
       prefix ip;
     }


     organization
       "IETF SPRING Working Group";
```

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

            WG Chair: Bruno Decraene
                      <mailto:bruno.decraene@orange.com>

            WG Chair: Rob Shakir
                      <mailto:robjs@google.com>
         Editor: Hooman Bidgoli
                   <mailto:hooman.bidgoli@nokia.com>
         Editor: Tanmoy Kundu
                   <mailto:tanmoy.kundu@nokia.com>
         Editor: Daniel Voyer
                   <mainlto: daniel.voyer@bell.com>
        description
          "The module defines a collection of YANG definition for
                p2mp policy module.";

        revision 2019-10-30 {
           description
             "First draft.";
           reference
             "RFC XXXX: A YANG Data Model for TREE-SID";
       }

       submodule router-p2mp-traffic-engineering {

       belongs-to root                  { prefix "root"; }
       import ietf-inet-types          { prefix "inet"; }
        import ietf-interfaces                  { prefix "if"; }

       container p2mp-traffic-engineering {
           presence "Configure Tree SID P2PM segment parameters.";
           description
               "Create p2mp policies and their corresponding
                candidate paths for p2mp trees.";

                list p2mp-policy {
                        key "root-address tree-id";
                        uses p2mp-policy-key;
                        description
                   "Each p2mp-policy identifies one or more p2mp
                    LSP for on the root towards a set of leaf/leaves.";

                        unique "p2mp-policy-name";

                        leaf p2mp-policy-name {
```

```
                                type string;
                                description
                                        "P2MP policy name to be referenced by
mvpn pmsi.";
                        }

                        leaf admin-state {
                                type enumeration{
                                        enum down  { value 0; }
                                        enum up    { value 1; }
                                }
                                default down;
                                description
                                        "Administratively enable/disable Tree
SID p2mp policy.";
                        }
                        leaf oper-state {
                                type enumeration{
                                        enum down  { value 0; }
                                        enum up    { value 1; }
                                }
                                default down;
                                config false;
                                description
                                        "Tree SID p2mp policy operational state
based on users.";
                        }

                        list leaf-list {
                                key "leaf-address";
                                uses leaf-list-key;
                                description
                               "This list consists of one or more endpoint/s
                                for a p2mp-segment.";

                                leaf admin-state {
                                        type enumeration{
                                                enum down  { value 0; }
                                                enum up    { value 1; }
                                        }
                                default down;
                                description
                                        "Administratively enable/disable
                                            each leaf/endpoint";
                                }
                        }

                        list candidate-path {
```

```
                description
                    "Candidate path is p2mp tree representing a
                     unique path from root to a specific
```

```
                     endpoint using the tree-id constraint.";

                key "protocol-id originator discriminator";

                uses candidate-path-key;

                             leaf candidate-path-name {
                                     type string;
                                     description
                                             "A candidate path name
                                              equivalent to a PLSP";
                             }                  leaf admin-state {
                                     type enumeration{
                                             enum down  { value 0; }
                                             enum up    { value 1; }
                                     }
                                     default down;
                    description
                        "Administratively enable/disable Tree
                         SID candidate path.";
                }
                leaf oper-state {
                                     type enumeration{
                                             enum down  { value 0; }
                                             enum up    { value 1; }
                                     }
                                     default down;
                    config false;
                    description
                        "candidate-path operational state based
                         on ilm programming.";
                }

                leaf preference {
                    type uint32;
                    default 100;
                    description
                        "Preference determines the best preferred
                         candidate-path among list of candidate path
                         towards a leaf. Higher preference is
                         chosen.";
                }

                             list constraints {
                                     description
                                             "Set of constraints";
                                     key "index";
```

```
                                        leaf index {
                                                type uint32;
                                                description
                                                        "Key index";
                                        }

                                        leaf attributes {
                                                type uint32;
                                                description
                                                        "Hooman to fill this, I
am not sure";
                                        }
                                }

                                list explicit-routing {
                                        description
                                                "Set of explicit routing";
                                        key "index";

                                        leaf index {
                                                type uint32;
                                                description
                                                        "Key index";
                                        }

                                        leaf attributes {
                                                type uint32;
                                                description
                                                        "Hooman to fill this, I
am not sure";
                                        }
                                }

                                list path-instances {
                                        description
                                                "Set of calculated path
                                                    given the constraints
                                                    and explicit routing per
                                                candidate path";
                                        key "index";

                                        leaf index {
                                                type uint32;
                                                description
                                                        "Key index";
                                        }

                                        leaf instance-id {
```

```
                                        type leafref {
                                        path "../../../../replication-
segment/replication-id";
```

```
                                        }
                                    }

                                    leaf oper-state {
                                        type enumeration{
                                            enum down  { value 0; }
                                            enum up    { value 1; }
                                        }
                                        default down;
                                        config false;
                                        description
                                            "Operational state of
this replication segment.";
                                    }

                                }
                            }
                        }

                        list replication-segment {
                            description
                           "A replication segment specifies the forwarding
                            information for one path-instance in a
                            candidate path.";

                        key "node-address replication-id";
                            uses replication-segment-key;

                            leaf admin-state {
                                type enumeration{
                                    enum down  { value 0; }
                                    enum up    { value 1; }
                                }
                                default down;
                                description
                                    "Administratively enable/disable Tree
SID replication segment.";
                            }

                            leaf oper-state {
                                type enumeration{
                                    enum down  { value 0; }
                                    enum up    { value 1; }
                                }
                                default down;
                                config false;
                                description
                                    "Replication segment operational
```

```
                                state based on SID programming.";
                        }
```

```
                container service-info {

                        leaf root-address {
                                type inet:ipv4-address;
                                description
                                        "Root address of the tree.";
                        }

                        leaf tree-id {
                                type uint32;
                                description
                                        "Tree ID uniquely identifies a
tunnel in the root,
                                        this also represent a specific
constraint. Also known as
                                        color and/or p2mp-id";
                        }

                        leaf instance-id {
                                type uint32;
                                description
                                        "Each LSP instance within a
root, tree-id.";
                        }
                }

                leaf replication-sid {
                        type uint32;
                        description
                                "incoming sid to identify this
                                 replication segment. Either
                                 BSID or MPLS label";
                }

                list downstream-nodes {
                        description
                                "Identifies each nexthop in a candidate
path.";
                        key "downstream-index";

                        uses downstream-key;

                        leaf next-hop-address {
                                type inet:ip-address;
                                description
                                        "Nexthop address of the
destination.";
                        }
```

```
                                leaf next-hop-interface-name {
                                        type if:interface-ref;
                                        description
                                                "Next hop out going
interface.";
```

```
                                }

                                leaf protecting-next-hop {
                                        type boolean;
                                        default false;
                                        description
                                                "True if this is a protect
nexthop.";
                                }
                                leaf protect-nexthop-id {
                                        type uint32;
                                        description
                                                "Nexthop protection id.";
                                }

                                choice label {
                                        list sid-list {
                                                key index;
                                                uses sid-list-key;
                                                description
                                                        "Out going label for
this nexthop.";

                                                leaf sid-segment-type {
                                                        type uint32;
                                                        description
                                                                "as defined in
draft-ietf-spring-segment-routing-policy section 4

                                                                 note if 2
replication segments are directly connected then

                                                                 they can be
steered via adjacency SID or IP Nexthop";
                                                }
                                        }

                                        list sr-policy {
                                                key replication-sid;
                                                uses sr-policy-key;
                                                description
                                                        "sr-policy sid for this
nexthop.";

                                                leaf sr-policy {
                                                type string;
                                                description
                                                        "SR policy name to be
referrenced by outloing label";
                                                }
```

```
                                        }

                                        list rsvp-te {
                                                key replication-sid;
                                                uses sr-policy-key;
                                                description
                                                        "RSVP TE Tunnel for
this nexthop.";
```

```
                                        leaf rsvp-te-tunnel-id {
                                        type uint32;
                                        description
                                                "rsvp-te-tunnel-id";
                                        }
                                    }
                                }
                            }
                    }
            }
            // --------------------------- GROUPINGS ---------------------------
            grouping p2mp-policy-key {

                    leaf root-address {
                            type inet:ip-address;
                            description
                                    "Root address of the tree.";
                    }

                    leaf tree-id {
                            type uint32;
                            description
                                    "Tree ID uniquely indentifies a tunnel in the
root,
                                    this also represent a spefic constraint. Also
known as
                                    color and/or p2mp-id";
                    }

            }

            grouping leaf-list-key {
                    leaf leaf-address{
                            type inet:ip-address;
                            description
                                    "leaf address of the this p2mp-tree";
                    }
            }

            grouping candidate-path-key {

                    leaf protocol-id {
                            type enumeration {
                                    enum pcep                       { value 10; }
                                    enum bgp-sr-policy  { value 20; }
                                    enum configuration  { value 30; }

                            }
```

```
                    description
                            "Protocol-Origin of a candidate path is an 8-
bit value
```

```
                                which identifies the component or protocol that
originates
                                or signals the candidate path.";
                }

                leaf originator {
                        type inet:ip-address;
                        description
                                "128 bit value, IPv4 address are encoded in
lower 32 bit.";
                }

                leaf discriminator {
                        type uint32;
                        description
                        "The Discriminator is a 32 bit value associated
                         with a candidate path that uniquely identifies
                         it within the context of an SR Policy from a
                         specific Protocol-Origin";
                }
        }

        grouping replication-segment-key {
                        leaf node-address {
                                type inet:ipv4-address;
                                description
                                        "Node address for which this
replication policy is used.";
                                }

                                leaf replication-id {
                                type uint32;
                                description
                                        "Each LSP per candidate path.";
                                }
        }

        grouping downstream-key {
                leaf downstream-index {
                        type uint32;
                        description
                                "Nexthop group or downstream replication node
index";
                }
        }

        grouping sid-list-key {
                leaf index {
```

```
                type uint32 {
                        range 1..2;
                }
        }
```

```
        }

        grouping sr-policy-key {
                leaf replication-sid {
                        type uint32 {
                                range 1..12;
                        }
                        description
                                "Index for push-label.";
                }
        }
   }
```

6. IANA Consideration

None

7. Security Considerations

TBD

8. Acknowledgments

9. References

9.1. Normative References

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

9.2. Informative References

[draft-ietf-pim-sr-p2mp-policy]
"D. Yoyer, C. Filsfils, R.Prekh, H.bidgoli, Z. Zhang, "draft-voyer-pim-sr-p2mp-policy"", October 2019.

[draft-ietf-spring-segment-routing-policy]
.

[draft-ietf-spring-sr-replication-segment]
"D. Yoyer, C. Filsfils, R.Prekh, H.bidgoli, Z. Zhang, "draft-voyer-pim-sr-p2mp-policy "draft-voyer-spring-sr-replication-segment"", July 2020.

Authors' Addresses

Hooman Bidgoli (editor)
Nokia
Ottawa
Canada

Email: hooman.bidgoli@nokia.com

Daniel Voyer
Bell Canada
Montreal
Canada

Email: daniel.yover@bell.ca

Rishabh Parekh
Cisco System
San Jose
USA

Email: riparekh@cisco.com

Tarek Saad
Juniper Networks
Ottawa
Canada

Email: tsaad@juniper.com

Tanmoy Kundu
Nokia
Mountain View
US

Email: tanmoy.kundu@nokia.com