SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 1, 2018

S. Litkowski
Orange Business Service
Y. Qu
Huawei
P. Sarkar
J. Tantsura
Individual

December 28, 2017

YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-08

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([I-D.ietf-spring-segment-routing]) configuration and operation. This YANG model is intended to be used on network elements to configure or operate segment routing. This document defines also generic containers that SHOULD be reused by IGP protocol modules to support segment routing.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 1, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 2
1.1. Tree diagram 3
2. Design of the Data Model 3
3. Configuration 5
4. IGP Control plane configuration 6
4.1. IGP interface configuration 6
4.1.1. Adjacency SID properties 7
4.1.1.1. Bundling 7
4.1.1.2. Protection 7
5. States 8
6. Notifications 8
7. YANG Module 8
8. Security Considerations 28
9. Acknowledgements 28
10. IANA Considerations 28
11. References 28
11.1. Normative References 28
11.2. Informative References 29
Authors' Addresses 29

1. Introduction

This document defines a YANG data model for segment routing configuration and operation. This document does not define the IGP extensions to support segment routing but defines generic groupings that SHOULD be reused by IGP extension modules. The reason of this design choice is to not require implementations to support all IGP extensions. For example, an implementation may support IS-IS extension but not OSPF.

1.1. Tree diagram

Tree diagrams used in this document follow the notation defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Design of the Data Model

As the module definition is just starting, it is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
  augment /rt:routing:
    +--rw segment-routing
       +--rw transport-type?      identityref
       +--ro node-capabilities
       |  +--ro transport-planes* [transport-plane]
       |  |  +--ro transport-plane    identityref
       |  +--ro readable-label-stack-depth?   uint8
       +--rw msd {msd}?
       |  +--rw node-msd?   uint8
       |  +--rw link-msd
       |     +--rw link-msds* [interface]
       |        +--rw interface    if:interface-ref
       |        +--rw msd?         uint8
       +--rw bindings
       |  +--rw mapping-server {mapping-server}?
       |  |  +--rw policy* [name]
       |  |     +--rw name    string
       |  |     +--rw ipv4
       |  |     |  +--rw mapping-entry* [prefix algorithm]
       |  |     |     +--rw prefix        inet:ipv4-prefix
       |  |     |     +--rw value-type?   enumeration
       |  |     |     +--rw start-sid     uint32
       |  |     |     +--rw range?        uint32
       |  |     |     +--rw algorithm     identityref
       |  |     +--rw ipv6
       |  |        +--rw mapping-entry* [prefix algorithm]
       |  |           +--rw prefix        inet:ipv6-prefix
       |  |           +--rw value-type?   enumeration
       |  |           +--rw start-sid     uint32
       |  |           +--rw range?        uint32
       |  |           +--rw algorithm     identityref
       |  +--rw connected-prefix-sid-map
       |  |  +--rw ipv4
       |  |  |  +--rw ipv4-prefix-sid* [prefix algorithm]
       |  |  |     +--rw prefix               inet:ipv4-prefix
       |  |  |     +--rw value-type?          enumeration
       |  |  |     +--rw start-sid            uint32
```

```
       |  |  |     +--rw range?              uint32
       |  |  |     +--rw algorithm           identityref
       |  |  |     +--rw last-hop-behavior?  enumeration {sid-last-hop-
behavior}?
       |  |  +--rw ipv6
       |  |     +--rw ipv6-prefix-sid* [prefix algorithm]
       |  |        +--rw prefix              inet:ipv6-prefix
       |  |        +--rw value-type?         enumeration
       |  |        +--rw start-sid           uint32
       |  |        +--rw range?              uint32
       |  |        +--rw algorithm           identityref
       |  |        +--rw last-hop-behavior?  enumeration {sid-last-hop-
behavior}?
       |  +--rw local-prefix-sid
       |     +--rw ipv4
       |     |  +--rw ipv4-prefix-sid-local* [prefix algorithm]
       |     |     +--rw prefix        inet:ipv4-prefix
       |     |     +--rw value-type?   enumeration
       |     |     +--rw start-sid     uint32
       |     |     +--rw range?        uint32
       |     |     +--rw algorithm     identityref
       |     +--rw ipv6
       |        +--rw ipv6-prefix-sid-local* [prefix algorithm]
       |           +--rw prefix        inet:ipv6-prefix
       |           +--rw value-type?   enumeration
       |           +--rw start-sid     uint32
       |           +--rw range?        uint32
       |           +--rw algorithm     identityref
       +--rw global-srgb
       |  +--rw srgb* [lower-bound upper-bound]
       |     +--rw lower-bound    uint32
       |     +--rw upper-bound    uint32
       +--rw srlb
       |  +--rw srlb* [lower-bound upper-bound]
       |     +--rw lower-bound    uint32
       |     +--rw upper-bound    uint32
       +--ro label-blocks*
       |  +--ro lower-bound?   uint32
       |  +--ro upper-bound?   uint32
       |  +--ro size?          uint32
       |  +--ro free?          uint32
       |  +--ro used?          uint32
       |  +--ro scope?         enumeration
       +--ro sid-list
          +--ro sid* [target sid source source-protocol binding-type]
             +--ro target             string
             +--ro sid                uint32
             +--ro algorithm?         uint8
```

```
            +--ro source             inet:ip-address
            +--ro used?              boolean
```

```
            +--ro source-protocol    -> /rt:routing/control-plane-protocols
            +                           /control-plane-protocol/name
            +--ro binding-type       enumeration
            +--ro scope?             enumeration

  notifications:
    +---n segment-routing-global-srgb-collision
    |  +--ro srgb-collisions*
    |     +--ro lower-bound?          uint32
    |     +--ro upper-bound?          uint32
    |     +--ro routing-protocol?     -> /rt:routing/control-plane-protocols
    |                                    /control-plane-protocol/name
    |     +--ro originating-rtr-id?   router-id
    +---n segment-routing-global-sid-collision
    |  +--ro received-target?       string
    |  +--ro new-sid-rtr-id?        router-id
    |  +--ro original-target?       string
    |  +--ro original-sid-rtr-id?   router-id
    |  +--ro index?                 uint32
    |  +--ro routing-protocol?      -> /rt:routing/control-plane-protocols
    |                                  /control-plane-protocol/name
    +---n segment-routing-index-out-of-range
       +--ro received-target?    string
       +--ro received-index?     uint32
       +--ro routing-protocol?   -> /rt:routing/control-plane-protocols
                                    /control-plane-protocol/name
```

3. Configuration

This module augments the "/rt:routing:" with a segment-routing container. This container defines all the configuration parameters related to segment-routing.

The segment-routing configuration is split in global configuration and interface configuration.

The global configuration includes :

- segment-routing transport type : The underlying transport type for segment routing. The version of the model limits the transport type to an MPLS dataplane. The transport-type is only defined once for a particular routing-instance and is agnostic to the control plane used. Only a single transport-type is supported in this version of the model.

- bindings : Defines prefix to SID mappings. The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. Two types of mappings are available :

  * Mapping-server : maps non local prefixes to a segment ID. Configuration of bindings does not automatically allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance (see Section 4). Multiple mapping policies may be defined.

  * Connected prefixes : maps connected prefixes to a segment ID. Advertisement of the mapping will be done by IGP when enabled for segment routing (see Section 4). The SID value can be expressed as an index (default), or an absolute value. The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".

- SRGB (Segment Routing Global Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels. The SRGB is also agnostic to the control plane used. So all routing-protocol instance will have to advertise the same SRGB.

- SRLB (Segment Routing Local Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels, reserved for lcoal SIDs.

4. IGP Control plane configuration

Support of segment-routing extensions for a particular IGP control plane is done by augmenting routing-protocol configuration with segment-routing extensions. This augmentation SHOULD be part of separate YANG modules in order to not create any dependency for implementations to support all protocol extensions.

This module defines groupings that SHOULD be used by IGP segment routing modules.

The "controlplane-cfg" grouping defines the generic global configuration for the IGP.

The "enabled" leaf enables segment-routing extensions for the routing-protocol instance.

The "bindings" container controls the routing-protocol instance's advertisement of local bindings and the processing of received bindings.

4.1. IGP interface configuration

The interface configuration is part of the "igp-interface-cfg" grouping and includes Adjacency SID properties.

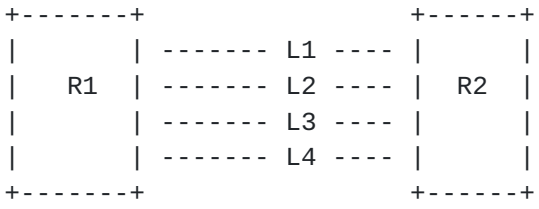
4.1.1. Adjacency SID properties

4.1.1.1. Bundling

This section is a first proposal on how to use S-bit in Adj-SID to create bundles. Authors would like to trigger discussion based on this first proposal.

In case of parallel IP links between routers, an additional Adjacency SID may be advertised representing more than one adjacency (i.e., a bundle of adjacencies). The "advertise-adj-group-sid" configuration controls whether or not an additional adjacency SID is advertised.

The "advertise-adj-group-sid" would be a list of "group-id". The "group-id" will permit to identify interfaces that must be bundled together.



In the figure above, R1 and R2 are interconnected by four links. A routing protocol adjacency is established on each link. Operator would like to create segment-routing Adj-SID that represent some bundles of links. We can imagine two different bundles : L1/L2 and L2/L3. To achieve this behavior, the service provider will configure a "group-id" X for both interfaces L1 and L2 and a "group-id" Y for both interfaces L3 and L3. This will result in R1 advertising an additional Adj-SID for each adjacency, for example a Adj-SID with S flag set and value of 400 will be added to L1 and L2. A Adj-SID with S flag set and value of 500 will be added to L3 and L4. As L1/L2 and L3/L4 does not share the same "group-id", a different SID value will be allocated.

4.1.1.2. Protection

The "advertise-protection" defines how protection for an interface is advertised. It does not control the activation or deactivation of protection. If the "single" option is used, a single Adj-SID will be advertised for the interface. If the interface is protected, the B-Flag for the Adj-SID advertisement will be set. If the "dual" option is used and if the interface is protected, two Adj-SIDs will be advertised for the interface adjacencies. One Adj-SID will always have the B-Flag set and the other will have the B-Flag clear. This

option is intended to be used in the case of traffic engineering where a path must use either protected segments or non-protected segments.

5. States

The operational states contains information reflecting the usage of allocated SRGB labels.

It also includes a list of all global SIDs, their associated bindings, and other information such as the source protocol and algorithm.

6. Notifications

The model defines the following notifications for segment-routing.

- segment-routing-global-srgb-collision: Rasied when a control plan advertised SRGB blocks have conflicts.
- segment-routing-global-sid-collision: Raised when a control plane advertised index is already associated with another target (in this version, the only defined targets are IPv4 and IPv6 prefixes).
- segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

7. YANG Module

```
<CODE BEGINS> file "ietf-segment-routing-common@2017-12-28.yang"
module ietf-segment-routing-common {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing-common";
  prefix sr-cmn;

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF SPRING - SPRING Working Group";

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

     Editor:    Stephane Litkowski
```

```
               <mailto:stephane.litkowski@orange.com>
     Editor:    Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>

     Author:    Acee Lindem
               <mailto:acee@cisco.com>
     Author:    Pushpasis Sarkar
               <mailto:pushpasis.ietf@gmail.com>
     Author:    Jeff Tantsura
               <jefftant.ietf@gmail.com>

    ";
  description
    "The YANG module defines a collection of types and groupings for
     Segment routing.

     Copyright (c) 2017 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

  reference "RFC XXXX";

  revision 2017-12-28 {
    description
      "
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-07-01 {
    description
      "
       *Conform to RFC6087BIS Appendix C
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-03-10 {
    description
      "
       * Add support of SRLB
```

```
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-28 {
    description
      "
       * Add support of MSD (Maximum SID Depth)
       * Update contact info
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-24 {
    description
      "Initial";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  feature sid-last-hop-behavior {
    description
      "Configurable last hop behavior.";
  }

  identity segment-routing-transport {
    description
      "Base identity for segment routing transport.";
  }

  identity segment-routing-transport-mpls {
    base segment-routing-transport;
    description
      "This identity represents MPLS transport for segment
       routing.";
  }

  identity segment-routing-transport-ipv6 {
    base segment-routing-transport;
    description
      "This identity represents IPv6 transport for segment
       routing.";
  }

  identity prefix-sid-algorithm {
    description
      "Base identity for prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-shortest-path {
    base prefix-sid-algorithm;
```

```
    description
      "The default behavior of prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-strict-spf {
    base prefix-sid-algorithm;
    description
      "This algorithm mandates that the packet is forwared
       according to ECMP-aware SPF algorithm.";
  }

  grouping srlr {
    description
      "Grouping for SR Label Range configuration.";
    leaf lower-bound {
        type uint32;
        description
          "Lower value in the block.";
      }
    leaf upper-bound {
        type uint32;
        description
          "Upper value in the block.";
      }
  }

  grouping srgb-cfg {
    description
      "Grouping for SR Label Range configuration.";
    list srgb {
      key "lower-bound upper-bound";
      ordered-by user;
      description
        "List of global blocks to be
         advertised.";
      uses srlr;
    }
  }

  grouping srlb-cfg {
    description
      "Grouping for SR Local Block range configuration.";
    list srlb {
      key "lower-bound upper-bound";
      ordered-by user;
      description
        "List of SRLBs.";
      uses srlr;
```

```
    }
  }

  grouping sid-value-type {
    description
      "Defines how the SID value is expressed.";
    leaf value-type {
      type enumeration {
        enum "index" {
          description
            "The value will be
             interpreted as an index.";
        }
        enum "absolute" {
          description
            "The value will become
             interpreted as an absolute
             value.";
        }
      }
      default "index";
      description
        "This leaf defines how value
         must be interpreted.";
    }
  }

  grouping ipv4-sid-cfg {
    description
      "This grouping defines cfg of prefix SID.";
    leaf prefix {
      type inet:ipv4-prefix;
      description
        "connected prefix sid.";
    }
    uses prefix-sid-attributes;
  }
  grouping ipv6-sid-cfg {
    description
      "This grouping defines cfg of prefix SID.";
    leaf prefix {
      type inet:ipv6-prefix;
      description
        "connected prefix sid.";
    }
    uses prefix-sid-attributes;
  }
```

```
  grouping last-hop-behavior {
    description
      "Defines last hop behavior";
    leaf last-hop-behavior {
      if-feature "sid-last-hop-behavior";
      type enumeration {
        enum "explicit-null" {
          description
            "Use explicit-null for the SID.";
        }
        enum "no-php" {
          description
            "Do no use PHP for the SID.";
        }
        enum "php" {
          description
            "Use PHP for the SID.";
        }
      }
      description
        "Configure last hop behavior.";
    }
  }

  grouping node-capabilities {
    description
      "Containing SR node capabilities.";
    container node-capabilities {
      config false;
      description
        "Shows the SR capability of the node.";
      list transport-planes {
        key "transport-plane";
        description
          "List of supported transport planes.";
        leaf transport-plane {
          type identityref {
            base segment-routing-transport;
          }
          description
            "Transport plane supported";
        }
      }
      leaf readable-label-stack-depth {
        type uint8;
        description
          "Number of MPLS labels that
           can be read in the stack.";
```

```
      }
    }
  }

  grouping prefix-sid-attributes {
    description
      "Containing SR attributes for a prefix.";
    uses sid-value-type;
    leaf start-sid {
      type uint32;
      mandatory true;
      description
        "Value associated with
         prefix. The value must
         be interpreted in the
         context of value-type.";
    }
    leaf range {
      type uint32;
      description
        "Describes how many SIDs could be
         allocated.";
    }
    leaf algorithm {
      type identityref {
        base prefix-sid-algorithm;
      }
      description
        "Prefix-sid algorithm.";
    }
  }
}
<CODE ENDS>
<CODE BEGINS> file "ietf-segment-routing@2017-12-28.yang"
module ietf-segment-routing {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing";
  prefix sr;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-routing {
    prefix rt;
  }
  import ietf-interfaces {
```

```
    prefix if;
  }
  import ietf-segment-routing-common {
    prefix sr-cmn;
  }

  organization
    "IETF SPRING Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/spring/>
     WG List:  <mailto:spring@ietf.org>

     Editor:    Stephane Litkowski
                <mailto:stephane.litkowski@orange.com>
     Editor:    Yingzhen Qu
                <mailto:yingzhen.qu@huawei.com>

     Author:    Acee Lindem
                <mailto:acee@cisco.com>
     Author:    Pushpasis Sarkar
                <mailto:pushpasis.ietf@gmail.com>
     Author:    Jeff Tantsura
                <jefftant.ietf@gmail.com>

    ";
  description
    "The YANG module defines a generic configuration model for
     Segment routing common across all of the vendor
     implementations.

     Copyright (c) 2017 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

  reference "RFC XXXX";

  revision 2017-12-28 {
    description
      "";
```

```
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2017-07-01 {
    description
      "
       * Implement NMDA model
       *Conform to RFC6087BIS Appendix C
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  revision 2017-03-10 {
    description
      "
       * Change global-sid-list to sid-list and add a leaf scope
       * Added support of SRLB
       * Added support of local sids
       * fixed indentations
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-28 {
    description
      "
       * Add support of MSD (Maximum SID Depth)
       * Update contact info
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-10-24 {
    description
      "
       * Moved common SR types and groupings to a seperate module
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-07-07 {
    description
      "
       * Add support of prefix-sid algorithm configuration
       * change routing-protocols to control-plane-protocols
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-03-17 {
    description
      "
       * Add notification segment-routing-global-srgb-collision
```

```
       * Add router-id to segment-routing-global-sid-collision
       * Remove routing-instance
       * Add typedef router-id
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-10-17 {
    description
      "
       * Add per-protocol SRGB config feature
       * Move SRBG config to a grouping
      ";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-06-22 {
    description
      "
       * Prefix SID config moved to
       connected-prefix-sid-map in global SR cfg
       rather than IGP.
      ";
    reference "draft-litkowski-spring-sr-yang-01";
  }
  revision 2015-04-23 {
    description
      "
       * Node flag deprecated from prefixSID
       * SR interface cfg moved to protocol
       * Adding multiple binding policies for SRMS
      ";
    reference "";
  }
  revision 2015-02-27 {
    description
      "Initial";
    reference "draft-litkowski-spring-sr-yang-00";
  }

  feature mapping-server {
    description
      "Support of SRMS.";
  }

  feature protocol-srgb {
    description
      "Support per-protocol srgb configuration.";
  }
```

```
  feature msd {
    description
      "Support of signaling MSD (Maximum SID Depth) in IGP.";
  }

  typedef system-id {
    type string {
      pattern "[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.[0-9A-Fa-f]{4}\\.00";
    }
    description
      "This type defines ISIS system id using pattern,
       system id looks like : 0143.0438.AeF0.00";
  }

  typedef router-id {
    type union {
      type system-id;
      type yang:dotted-quad;
    }
    description
      "OSPF/BGP router id or ISIS system ID.";
  }

  grouping controlplane-cfg {
    description
      "Defines protocol configuration.";
    container segment-routing {
      description
        "segment routing global config.";
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enables segment-routing
           protocol extensions.";
      }
      container bindings {
        description
          "Control of binding advertisement
           and reception.";
        container advertise {
          description
            "Authorize the advertise
             of local mappings in binding TLV.";
          leaf-list policies {
            type string;
            description
              "List of policies to be advertised.";
```

```
        }
      }
      leaf receive {
        type boolean;
        default "true";
        description
          "Authorize the reception and usage
           of binding TLV.";
      }
    }
  }
}

grouping igp-interface-cfg {
  description
    "Grouping for IGP interface cfg.";
  container segment-routing {
    description
      "container for SR interface cfg.";
    container adjacency-sid {
      description
        "Defines the adjacency SID properties.";
      list advertise-adj-group-sid {
        key "group-id";
        description
          "Control advertisement of S flag.
           Enable to advertise a common Adj-SID
           for parallel links.";
        leaf group-id {
          type uint32;
          description
            "The value is an internal value to identify
             a group-ID. Interfaces with the same
             group-ID will be bundled together.";
        }
      }
      leaf advertise-protection {
        type enumeration {
          enum "single" {
            description
              "A single Adj-SID is associated
               with the adjacency and reflects
               the protection configuration.";
          }
          enum "dual" {
            description
              "Two Adj-SIDs will be associated
               with the adjacency if interface
```

```
                is protected. In this case
                one will be enforced with
                backup flag set, the other
                will be enforced to backup flag unset.
                In case, protection is not configured,
                a single Adj-SID will be advertised
                with backup flag unset.";
            }
          }
          description
            "If set, the Adj-SID refers to an
             adjacency being protected.";
        }
      }
    }
  }

  grouping msd-cfg {
    description
      "MSD configuration grouping.";
    leaf node-msd {
      type uint8;
      description
        "Node MSD is the lowest MSD supported by the node.";
    }
    container link-msd {
      description
        "Link MSD is a number represetns the particular link MSD value.";
      list link-msds {
        key "interface";
        description
          "List of link MSDs.";
        leaf interface {
          type if:interface-ref;
          description
            "Name of the interface.";
        }
        leaf msd {
          type uint8;
          description
            "SID depth of the interface associated with the link.";
        }
      }
    }
  }

  augment "/rt:routing" {
    description
```

```
      "This augments routing-instance
       configuration with segment-routing.";
    container segment-routing {
      description
        "segment routing global config.";
      leaf transport-type {
        type identityref {
          base sr-cmn:segment-routing-transport;
        }
        default "sr-cmn:segment-routing-transport-mpls";
        description
          "Dataplane to be used.";
      }
      uses sr-cmn:node-capabilities;
      container msd {
        if-feature "msd";
        description
          "MSD configuration.";
        uses msd-cfg;
      }
      container bindings {
        description
          "List of bindings.";
        container mapping-server {
          if-feature "mapping-server";
          description
            "Configuration of mapping-server
             local entries.";
          list policy {
            key "name";
            description
              "Definition of mapping policy.";
            leaf name {
              type string;
              description
                "Name of the mapping policy.";
            }
            container ipv4 {
              description
                "IPv4 mapping entries.";
              list mapping-entry {
                key "prefix algorithm";
                description
                  "Mapping entries.";
                uses sr-cmn:ipv4-sid-cfg;
              }
            }
            container ipv6 {
```

```
            description
              "IPv6 mapping entries.";
            list mapping-entry {
              key "prefix algorithm";
              description
                "Mapping entries.";
              uses sr-cmn:ipv6-sid-cfg;
            }
          }
        }
      }
      container connected-prefix-sid-map {
        description
          "Prefix SID configuration.";
        container ipv4 {
          description
            "Parameters associated with IPv4 prefix SID";
          list ipv4-prefix-sid {
            key "prefix algorithm";
            description
              "List of prefix SID
               mapped to IPv4 local prefixes.";
            uses sr-cmn:ipv4-sid-cfg;
            uses sr-cmn:last-hop-behavior;
          }
        }
        container ipv6 {
          description
            "Parameters associated with IPv6 prefix SID";
          list ipv6-prefix-sid {
            key "prefix algorithm";
            description
              "List of prefix SID
               mapped to IPv6 local prefixes.";
            uses sr-cmn:ipv6-sid-cfg;
            uses sr-cmn:last-hop-behavior;
          }
        }
      }
      container local-prefix-sid {
        description
          "Local sid configuration.";
        container ipv4 {
          description
            "List of local ipv4 sids.";
          list ipv4-prefix-sid-local {
            key "prefix algorithm";
            description
```

```
              "List of local prefix-sid.";
            uses sr-cmn:ipv4-sid-cfg;
          }
        }
        container ipv6 {
          description
            "List of local ipv6 sids.";
          list ipv6-prefix-sid-local {
            key "prefix algorithm";
            description
              "List of local prefix-sid.";
            uses sr-cmn:ipv6-sid-cfg;
          }
        }
      }
    }
    container global-srgb {
      description
        "Global SRGB configuration.";
      uses sr-cmn:srgb-cfg;
    }
    container srlb {
      description
        "SR Local Block configuration.";
      uses sr-cmn:srlb-cfg;
    }

    list label-blocks {
      config false;
      description
        "List of labels blocks currently
         in use.";
      leaf lower-bound {
        type uint32;
        description
          "Lower bound of the label block.";
      }
      leaf upper-bound {
        type uint32;
        description
          "Upper bound of the label block.";
      }
      leaf size {
        type uint32;
        description
          "Number of indexes in the block.";
      }
      leaf free {
```

```
          type uint32;
          description
            "Number of indexes free in the block.";
        }
        leaf used {
          type uint32;
          description
            "Number of indexes used in the block.";
        }
        leaf scope {
          type enumeration {
            enum "global" {
              description
                "Global sid.";
            }
            enum "local" {
              description
                "Local sid.";
            }
          }
          description
            "Scope of this label block.";
        }
      }
      container sid-list {
        config false;
        description
          "List of prefix and SID associations.";
        list sid {
          key "target sid source source-protocol binding-type";
          ordered-by system;
          description
            "Binding.";
          leaf target {
            type string;
            description
              "Defines the target of the binding.
               It can be a prefix or something else.";
          }
          leaf sid {
            type uint32;
            description
              "Index associated with the prefix.";
          }
          leaf algorithm {
            type uint8;
            description
              "Algorithm to be used for the prefix
```

```
            SID.";
        }
        leaf source {
          type inet:ip-address;
          description
            "IP address of the router than own
             the binding.";
        }
        leaf used {
          type boolean;
          description
            "Defines if the binding is used
             in forwarding plane.";
        }
        leaf source-protocol {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
               + "rt:control-plane-protocol/rt:name";
          }
          description
            "Rtg protocol that owns the binding";
        }
        leaf binding-type {
          type enumeration {
            enum "prefix-sid" {
              description
                "Binding is learned from
                 a prefix SID.";
            }
            enum "binding-tlv" {
              description
                "Binding is learned from
                 a binding TLV.";
            }
          }
          description
            "Type of binding.";
        }
        leaf scope {
          type enumeration {
            enum "global" {
              description
                "Global sid.";
            }
            enum "local" {
              description
                "Local sid.";
            }
```

```
            }
            description
              "The sid is local or global.";
          }
        }
      }
    }
  }

  notification segment-routing-global-srgb-collision {
    description
      "This notification is sent when received SRGB blocks from
       a router conflict.";
    list srgb-collisions {
      description
        "List of SRGB blocks that conflict.";
      leaf lower-bound {
        type uint32;
        description
          "Lower value in the block.";
      }
      leaf upper-bound {
        type uint32;
        description
          "Upper value in the block.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference that received the event.";
      }
      leaf originating-rtr-id {
        type router-id;
        description
          "Originating router id of this SRGB block.";
      }
    }
  }
  notification segment-routing-global-sid-collision {
    description
      "This notification is sent when a new mapping is learned
       , containing mapping
       where the SID is already used.
       The notification generation must be throttled with at least
       a 5 second gap. ";
```

```
    leaf received-target {
      type string;
      description
        "Target received in the controlplane that
                          caused SID collision.";
    }
    leaf new-sid-rtr-id {
      type router-id;
      description
        "Router Id that advertising the conflicting SID.";
    }
    leaf original-target {
      type string;
      description
        "Target already available in database that have the same SID
         as the received target.";
    }
    leaf original-sid-rtr-id {
      type router-id;
      description
        "Original router ID that advertised the conflicting SID.";
    }
    leaf index {
      type uint32;
      description
        "Value of the index used by two different prefixes.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/"
           + "rt:control-plane-protocol/rt:name";
      }
      description
        "Routing protocol reference that received the event.";
    }
  }
  notification segment-routing-index-out-of-range {
    description
      "This notification is sent when a binding
       is received, containing a segment index
       which is out of the local configured ranges.
       The notification generation must be throttled with at least
       a 5 second gap. ";
    leaf received-target {
      type string;
      description
        "Target received in the controlplane
                          that caused SID collision.";
```

```
    }
    leaf received-index {
      type uint32;
      description
        "Value of the index received.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols/"
           + "rt:control-plane-protocol/rt:name";
      }
      description
        "Routing protocol reference that received the event.";
    }
  }
}
<CODE ENDS>
```

8. Security Considerations

TBD.

9. Acknowledgements

Authors would like to thank Derek Yeung, Acee Lindem, Greg Hankins, Hannes Gredler, Uma Chunduri, Jeffrey Zhang, Shradda Hedge, Les Ginsberg for their contributions.

10. IANA Considerations

TBD.

11. References

11.1. Normative References

[I-D.ietf-isis-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-04 (work in progress), June 2017.

[I-D.ietf-ospf-segment-routing-msd]
Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling MSD (Maximum SID Depth) using OSPF", draft-ietf-ospf-segment-routing-msd-05 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

11.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-04 (work in progress), December 2017.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Individual

Email: jefftant.ietf@gmail.com