

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2019

S. Litkowski
Orange Business Service
Y. Qu
Huawei
A. Lindem
Cisco
P. Sarkar
Individual
J. Tantsura
Apstra
February 28, 2019

**YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-12**

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([RFC8402]) 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.

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 <http://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 September 1, 2019.

Copyright Notice

Copyright (c) 2019 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 (<http://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](#)
- [2.](#) Terminology and Notation [3](#)
 - [2.1.](#) Tree diagram [3](#)
 - [2.2.](#) Prefixes in Data Node Names [3](#)
- [3.](#) Design of the Data Model [3](#)
- [4.](#) Configuration [5](#)
- [5.](#) IGP Control plane configuration [6](#)
 - [5.1.](#) IGP interface configuration [7](#)
 - [5.1.1.](#) Adjacency SID properties [7](#)
 - [5.1.1.1.](#) Bundling [7](#)
 - [5.1.1.2.](#) Protection [8](#)
- [6.](#) States [8](#)
- [7.](#) Notifications [8](#)
- [8.](#) YANG Module [8](#)
- [9.](#) Security Considerations [25](#)
- [10.](#) Acknowledgements [25](#)
- [11.](#) IANA Considerations [25](#)
- [12.](#) References [25](#)
 - [12.1.](#) Normative References [25](#)
 - [12.2.](#) Informative References [26](#)
- Authors' Addresses [27](#)

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.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [[RFC8342](#)].

2. Terminology and Notation

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 [\[RFC2119\]](#).

2.1. Tree diagram

Tree diagrams used in this document follow the notation defined in [\[RFC8340\]](#).

2.2. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

3. 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 entropy-readable-label-depth?  uint8
      +--rw msd {max-sid-depth}?
        | +--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 entries
|  |  |  |  +--rw mapping-entry* [prefix algorithm]
|  |  |  |  |  +--rw prefix      inet:ip-prefix
|  |  |  |  |  +--rw value-type? enumeration
|  |  |  |  |  +--rw start-sid   uint32
|  |  |  |  |  +--rw range?     uint32
|  |  |  |  |  +--rw algorithm  identityref
|  +--rw connected-prefix-sid-map
|  |  +--rw connected-prefix-sid* [prefix algorithm]
|  |  |  +--rw prefix          inet:ip-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 local-prefix-sid* [prefix algorithm]
|  |  |  +--rw prefix          inet:ip-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

```

4. 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 :

- o 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.

- o 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 5](#)). 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 5](#)). 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".
- o 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.
- o SRLB (Segment Routing Local Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels, reserved for local SIDs.

5. 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.

5.1. IGP interface configuration

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

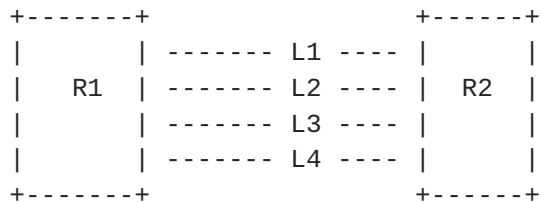
5.1.1. Adjacency SID properties

5.1.1.1. 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.

5.1.1.2. 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.

6. 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.

7. Notifications

The model defines the following notifications for segment-routing.

- o segment-routing-global-srgb-collision: Raised when a control plane advertised SRGB blocks have conflicts.
- o 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).
- o segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

8. YANG Module

The following RFCs and drafts are not referenced in the document text but are referenced in the ietf-segment-routing-common.yang and/or ietf-segment-routing.yang module: [[RFC6991](#)], [[RFC8294](#)], [[RFC8476](#)], and [[RFC8491](#)].

```
<CODE BEGINS> file "ietf-segment-routing-common@2019-02-28.yang"  
module ietf-segment-routing-common {
```



```
yang-version 1.1;
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 generic types and
  grouping for Segment Routing (SR) as described in RFC 8402.

  Copyright (c) 2019 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 2019-02-28 {
  description
    "Initial version";
  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
    "Shortest Path First (SPF) prefix-sid algorithm. This
    is the default algorithm.";
}

identity prefix-sid-algorithm-strict-spf {
  base prefix-sid-algorithm;
  description
    "This algorithm mandates that the packet is forwarded
    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 label range.";
  }
  leaf upper-bound {
    type uint32;
    description
      "Upper value in the label range.";
  }
}

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

grouping srlb {
  description
    "Grouping for SR Local Block range.";
  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 prefix-sid {
    description
        "This grouping defines cfg of prefix SID.";
    leaf prefix {
        type inet:ip-prefix;
        description
            "connected prefix sid.";
    }
    uses prefix-sid-attributes;
}

grouping ipv4-sid {
    description
        "Grouping for an IPv4 prefix SID.";
    leaf prefix {
        type inet:ipv4-prefix;
        description
            "Connected IPv4 prefix sid.";
    }
    uses prefix-sid-attributes;
}

grouping ipv6-sid {
    description
        "Grouping for an IPv6 prefix SID.";
    leaf prefix {
        type inet:ipv6-prefix;
        description
            "Connected ipv6 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 not use Penultimate Hop Popping (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 entropy-readable-label-depth {
    type uint8;
    description
      "Maximum label stack depth that a router can read.";
  }
}
}
```



```
grouping prefix-sid-attributes {
  description
    "Grouping for Segment Routing (SR) prefix attributes.";
  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
      "Indicates how many SIDs can be allocated.";
  }
  leaf algorithm {
    type identityref {
      base prefix-sid-algorithm;
    }
    description
      "Prefix-sid algorithm.";
  }
}
}
}
<CODE ENDS>
<CODE BEGINS> file "ietf-segment-routing@2019-02-28.yang"
module ietf-segment-routing {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing";
  prefix sr;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-routing {
    prefix rt;
  }
  import ietf-interfaces {
    prefix if;
  }
  import ietf-routing-types {
    prefix rt-types;
  }
  import ietf-segment-routing-common {
    prefix sr-cmn;
  }
}
```


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 generic configuration model for Segment routing common across all of the vendor implementations.

Copyright (c) 2019 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 2019-02-28 {

description

"Initial Version";

reference "RFC XXXX: YANG Data Model for Segment Routing.";

}

feature mapping-server {

description

"Support for Segment Routing Mapping Server (SRMS).";

}


```
feature protocol-srgb {
  description
    "Support for per-protocol Segment Routing Global Block
    (SRGB) configuration.";
}

feature max-sid-depth {
  description
    "Support for 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}';
  }
  description
    "This type defines IS-IS system-id using pattern,
    An example system-id is 0143.0438.AEF0";
}

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

grouping sr-controlplane {
  description
    "Defines protocol configuration.";
  container segment-routing {
    description
      "Segment Routing global configuration.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "Enables segment-routing protocol extensions.";
    }
  }
  container bindings {
    description
      "Control of binding advertisement and reception.";
    container advertise {
      description
        "Control advertisement of local mappings";
    }
  }
}
```



```
        in binding TLVs.";
    leaf-list policies {
        type string;
        description
            "List of binding advertisement policies.";
    }
}
leaf receive {
    type boolean;
    default "true";
    description
        "Allow the reception and usage of binding TLVs.";
}
}
}
}

grouping igp-interface {
    description
        "Grouping for IGP interface configuration.";
    container segment-routing {
        description
            "Container for SR interface configuration.";
        container adjacency-sid {
            description
                "Adjacency SID configuration.";
            list advertise-adj-group-sid {
                key "group-id";
                description
                    "Control advertisement of S flag. Enable advertisement
                    of 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 the interface is protected. In this case, will
        be advertised with backup flag set, the other will
        be advertised with the backup flag clear. In case
        protection is not configured, single Adj-SID will
        be advertised with the backup flag clear.";
    }
}
description
    "If set, the Adj-SID refers to a protected adjacency.";
}
}
}
}

grouping max-sid-depth {
    description
        "Maximum SID Depth (MSD)D configuration grouping.";
    leaf node-msd {
        type uint8;
        description
            "Node MSD is the lowest MSD supported by the node.";
    }
    container link-msd {
        description
            "MSD supported by an individual interface.";
        list link-msds {
            key "interface";
            description
                "List of link MSDs.";
            leaf interface {
                type if:interface-ref;
                description
                    "Reference to device interface.";
            }
            leaf msd {
                type uint8;
                description
                    "MSD supported by the interface.";
            }
        }
    }
}

augment "/rt:routing" {
    description
        "This augments routing data model (RFC 8349)
        with Segment Routing (SR).";
```



```
container segment-routing {
  description
    "Segment Routing global configuration.";
  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 "max-sid-depth";
    description
      "MSD configuration.";
    uses max-sid-depth;
  }
  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
          "List mapping-server policies.";
        leaf name {
          type string;
          description
            "Name of the mapping policy.";
        }
        container entries {
          description
            "IPv4/IPv6 mapping entries.";
          list mapping-entry {
            key "prefix algorithm";
            description
              "Mapping entries.";
            uses sr-cmn:prefix-sid;
          }
        }
      }
    }
  }
  container connected-prefix-sid-map {
    description
```



```
    "Prefix SID configuration.";
  list connected-prefix-sid {
    key "prefix algorithm";
    description
      "List of prefix SID mapped to IPv4/IPv6
       local prefixes.";
    uses sr-cmn:prefix-sid;
    uses sr-cmn:last-hop-behavior;
  }
}
container local-prefix-sid {
  description
    "Local sid configuration.";
  list local-prefix-sid {
    key "prefix algorithm";
    description
      "List of local IPv4/IPv6 prefix-sids.";
    uses sr-cmn:prefix-sid;
  }
}
}
container global-srgb {
  description
    "Global SRGB configuration.";
  uses sr-cmn:srgb;
}
container srlb {
  description
    "Segment Routing Local Block (SRLB) configuration.";
  uses sr-cmn:srlb;
}

list label-blocks {
  config false;
  description
    "List of label 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 free indexes in the block.";
  }
  leaf used {
    type uint32;
    description
      "Number of indexes in use 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
      "SID 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 that owns the binding.";
}
leaf used {
  type boolean;
  description
    "Indicates if the binding is install in the
    forwarding plane.";
}
leaf source-protocol {
  type leafref {
    path "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol/rt:name";
  }
  description
    "Routing 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
      "SID scoping.";
  }
}
}
}
}

notification segment-routing-global-srgb-collision {
  description
    "This notification is sent when SRGB blocks received from
    routers 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 for SRGB collision.";
    }
    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 s mapping where the SID is already used.
    The notification generation must be throttled with at least
    a 5 second gap between notifications.";
  leaf received-target {
```



```
    type string;
    description
      "Target received in the router advertisement that caused
       the SID collision.";
  }
  leaf new-sid-rtr-id {
    type router-id;
    description
      "Router ID that advertised the conflicting SID.";
  }
  leaf original-target {
    type string;
    description
      "Target already available in the database with the same SID
       as the received target.";
  }
  leaf original-sid-rtr-id {
    type router-id;
    description
      "Router-ID for the router that originally advertised the
       conflicting SID, i.e., the instance in the database.";
  }
  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 for conflicting SID.";
  }
}
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 between
     notifications.";
  leaf received-target {
    type string;
    description
      "Target received in the router advertisement with
       the out-of-range index.";
```



```
    }
    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 for out-of-range indexd.";
    }
  }
}
<CODE ENDS>
```

9. Security Considerations

TBD.

10. 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.

11. IANA Considerations

TBD.

12. References

12.1. 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>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", [RFC 8294](#), DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", [RFC 8342](#), DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", [RFC 8343](#), DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", [RFC 8349](#), DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", [RFC 8402](#), DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8476] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", [RFC 8476](#), DOI 10.17487/RFC8476, December 2018, <<https://www.rfc-editor.org/info/rfc8476>>.
- [RFC8491] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", [RFC 8491](#), DOI 10.17487/RFC8491, November 2018, <<https://www.rfc-editor.org/info/rfc8491>>.

12.2. Informative References

- [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>>.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Huawei

Email: yingzhen.qu@huawei.com

Acee Lindem
Cisco
301 Mindenhall Way
Cary, NC 27513
US

Email: acee@cisco.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

