

Inter-Domain Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 26, 2022

K. Deevi
K. Raza
Cisco
K. Majumdar
Commscope
B. Decraene
Orange
Z. Jiang
Tencent
January 26, 2022

YANG data model for BGP Segment Routing TE Extensions
draft-deevi-idr-bgp-srte-yang-03

Abstract

This document defines a YANG data model that can be used to configure and manage Segment Routing TE extensions in BGP.

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 26, 2022.

Copyright Notice

Copyright (c) 2022 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

Internet-Draft Yang Model for BGP Segment Routing TE January 2022

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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

| | | |
|----------------------|--|--------------------|
| 1. | Introduction | 2 |
| 1.1. | Requirements Language | 3 |
| 2. | BGP Segment Routing Traffic Engineering Yang model | 3 |
| 2.1. | Overview | 3 |
| 2.2. | SR Policy | 4 |
| 2.3. | Automatic Steering | 4 |
| 3. | Yang Tree | 5 |
| 3.1. | SR Policy | 5 |
| 3.2. | Automatic Steering | 8 |
| 4. | Yang Module | 9 |
| 5. | Contributors | 25 |
| 6. | IANA Considerations | 26 |
| 7. | Security Considerations | 26 |
| 8. | Acknowledgements | 26 |
| 9. | References | 26 |
| 9.1. | Normative References | 26 |
| 9.2. | Informative References | 27 |
| | Authors' Addresses | 28 |

[1.](#) Introduction

YANG [[RFC6020](#)] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [[RFC6241](#)]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces

(e.g. ReST) [[RFC8040](#)] and encodings other than XML (e.g. JSON) [[RFC7951](#)] are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines the YANG model for Segment Routing TE specific extensions in BGP.

[1.1.](#) Requirements Language

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

[2.](#) BGP Segment Routing Traffic Engineering Yang model

[2.1.](#) Overview

Segment Routing (SR), as defined in [[RFC8402](#)], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain.

When applied to ipv6 data-plane (i.e. SRv6), the ordered set of instructions are realized via SRv6 SIDs. The various functions and behaviors corresponding to network programming using SRv6 are specified in [[I-D.ietf-spring-srv6-network-programming](#)].

This document defines Yang model for the Segment Routing TE extensions applicable for BGP as following:

- o BGP signaled SR Policy as described in [[I-D.ietf-idr-segment-routing-te-policy](#)].
- o Automatic Steering as described in [[I-D.ietf-spring-segment-routing-policy](#)] and [[I-D.ietf-idr-segment-routing-te-policy](#)].

The Yang extensions proposed in this model augment the base BGP model defined in [[I-D.ietf-idr-bgp-model](#)].

Note: Base BGP model does not have a common structure for BGP RIB. The placeholder containers defined in this model can be removed once base BGP model has the BGP RIB structure.

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [[RFC8342](#)]. The operational state data is combined with the associated configuration data in the same hierarchy [[RFC8407](#)]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all

"config true" (rw) and "config false" (ro) nodes defined in the schema.

[2.2.](#) SR Policy

Architecture for SR Policies is described in [[I-D.ietf-spring-segment-routing-policy](#)]. BGP Signaled SR Policies are described in the [[I-D.ietf-idr-segment-routing-te-policy](#)]. Following Yang extensions for SR Policy configuration and state data are applicable:

- o Addition of identities extending the BGP-AFI-SAFI base identity. This is to add two new address families namely IPv4 SR-policy and IPv6 SR-policy, as described in [[I-D.ietf-idr-segment-routing-te-policy](#)].
- o BGP Signaled SR Policy candidate paths. These refer to the explicit candidate paths signaled via BGP as SAFI NLRIs, state of which is applicable in the context of BGP speaker process. This is modeled by adding SR Policy address family specific container under generic BGP afi-safi list entry defined in the base BGP model [[I-D.ietf-idr-bgp-model](#)].
- o On Demand SR Policy candidate paths. These refer to the dynamic candidate paths as described in [[I-D.ietf-spring-segment-routing-policy](#)]. There are two parts to this in the context of BGP. A set of authorized SR Policy colors for on demand policy triggers, and the actual instantiated candidate paths per BGP next-hop. New containers and lists are

added under BGP global mode to model this information.

- o SR Policy state in the context of BGP speaker. This represents the state SR Policies (regardless of method of instantiation per candidate path). The SR Policy state is maintained in the context of BGP speaker process to realize the Automatic Steering of overlay routes. Automatic Steering extensions are described in the next section.

Note: The common parameters and datatypes for the SR Policy, currently defined in this model, should be imported from SR Policy Manager model, once available.

[2.3.](#) Automatic Steering

Automatic Steering (AS) refers to the ability to forward traffic over a SR Policy on the head-end, as described in [\[I-D.ietf-spring-segment-routing-policy\]](#). When a BGP route is received with the color extended community and if the color value

matches the color of an authorized SR Policy installed on the head-end, the route is programmed to resolve over SR Policy in forwarding. Automatic Steering information associated with the BGP routes is modeled as state information per route.

TBD: The configuration parameters for Automatic Steering are yet to be added as an augmentation to the BGP route policy model. Such as, extensions for opaque color extended community in BGP policy model, and the Color Only (CO) flags controlling the Automatic Steering behavior as described in [\[I-D.ietf-idr-segment-routing-te-policy\]](#).

[3.](#) Yang Tree

[3.1.](#) SR Policy

On Demand Nexthop (ODN) policies triggered by BGP

```
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
+--rw segment-routing
  +--rw on-demand-policies
    | +--ro authorized-colors
    | | +--ro colors* [color]
```

```

| |      +---ro color      uint32
| +---ro installed-policies
|       +---ro sr-policy* [color end-point]
|         +---ro color      uint32
|         +---ro end-point  inet:ip-address
+---ro policy-state
    +---ro sr-policy* [color end-point]
        +---ro color      uint32
        +---ro end-point  inet:ip-address
        +---ro policy-state? enumeration
        +---ro binding-sid? sid-type
        +---ro steering-disabled? empty
        +---ro ref-count?  uint32

```

BGP Signaled Explicit SR Policies under ipv4 and ipv6 SR-Policy SAFI

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp

```

+---rw ipv4-srpolicy
    +---ro explicit-policies
        +---ro sr-policy* [distinguisher color end-point]
            +---ro distinguisher      uint32
            +---ro color              uint32
            +---ro end-point          inet:ip-address
            +---ro preference?        uint32
            +---ro explicit-binding-sid
            | +---ro binding-sid?    sid-type

```

```

| +---ro strict?          boolean
| +---ro drop-on-invalid? boolean
+---ro usable?            boolean
+---ro registered?        boolean
+---ro segment-lists
    +---ro segment-list* [weight]
        +---ro weight      uint32
        +---ro segments
            +---ro segment* [index]
                +---ro index      uint32
                +---ro type?      segment-type
            +---ro segment-types
                +---ro segment-type-1
                | +---ro sid-value? rt-types:mpls-label
                +---ro segment-type-2

```

```

|   +---ro sid-value?      srv6-types:srv6-sid
+---ro segment-type-3
|   +---ro ipv4-address?   inet:ipv4-address
|   +---ro algorithm?      uint8
+---ro segment-type-4
|   +---ro ipv6-address?   inet:ipv6-address
|   +---ro algorithm?      uint8
+---ro segment-type-5
|   +---ro ipv4-address?   inet:ipv4-address
|   +---ro interface-identifier?  uint32
+---ro segment-type-6
|   +---ro local-ipv4-address?   inet:ipv4-address
|   +---ro remote-ipv4-address?  inet:ipv4-address
+---ro segment-type-7
|   +---ro local-ipv6-address?   inet:ipv6-ad
|   +---ro local-interface-identifier?  uint32
|   +---ro remote-ipv6-address?   inet:ipv6-ad
|   +---ro remote-interface-identifier?  uint32
+---ro segment-type-8
|   +---ro local-ipv6-address?   inet:ipv6-address
|   +---ro remote-ipv6-address?  inet:ipv6-address
+---ro segment-type-9
|   +---ro ipv6-address?   inet:ipv6-address
|   +---ro algorithm?      uint8
+---ro segment-type-10
|   +---ro local-ipv6-address?   inet:ipv6-ad
|   +---ro local-interface-identifier?  uint32
|   +---ro remote-ipv6-address?   inet:ipv6-ad
|   +---ro remote-interface-identifier?  uint32
+---ro segment-type-11
|   +---ro local-ipv6-address?   inet:ipv6-address
|   +---ro remote-ipv6-address?  inet:ipv6-address
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp

```

```

+---rw ipv6-srpolicy
+---ro explicit-policies
+---ro sr-policy* [distinguisher color end-point]
|   +---ro distinguisher      uint32
|   +---ro color              uint32
|   +---ro end-point          inet:ip-address
|   +---ro preference?        uint32
+---ro explicit-binding-sid

```

```

|   +---ro binding-sid?          sid-type
|   +---ro strict?              boolean
|   +---ro drop-on-invalid?     boolean
+---ro usable?                  boolean
+---ro registered?              boolean
+---ro segment-lists
    +---ro segment-list* [weight]
        +---ro weight           uint32
        +---ro segments
            +---ro segment* [index]
                +---ro index           uint32
                +---ro type?          segment-type
                +---ro segment-types
                    +---ro segment-type-1
                        |   +---ro sid-value?    rt-types:mpls-label
                    +---ro segment-type-2
                        |   +---ro sid-value?    srv6-types:srv6-sid
                    +---ro segment-type-3
                        |   +---ro ipv4-address?  inet:ipv4-address
                        |   +---ro algorithm?     uint8
                    +---ro segment-type-4
                        |   +---ro ipv6-address?  inet:ipv6-address
                        |   +---ro algorithm?     uint8
                    +---ro segment-type-5
                        |   +---ro ipv4-address?  inet:ipv4-address
                        |   +---ro interface-identifier?  uint32
                    +---ro segment-type-6
                        |   +---ro local-ipv4-address?  inet:ipv4-address
                        |   +---ro remote-ipv4-address?  inet:ipv4-address
                    +---ro segment-type-7
                        |   +---ro local-ipv6-address?  inet:ipv6-ad
                        |   +---ro local-interface-identifier?  uint32
                        |   +---ro remote-ipv6-address?  inet:ipv6-ad
                        |   +---ro remote-interface-identifier?  uint32
                    +---ro segment-type-8
                        |   +---ro local-ipv6-address?  inet:ipv6-address
                        |   +---ro remote-ipv6-address?  inet:ipv6-address
                    +---ro segment-type-9
                        |   +---ro ipv6-address?  inet:ipv6-address
                        |   +---ro algorithm?     uint8

```

```

|   +--ro local-ipv6-address?      inet:ipv6-ad
|   +--ro local-interface-identifier?  uint32
|   +--ro remote-ipv6-address?      inet:ipv6-ad
|   +--ro remote-interface-identifier?  uint32
+--ro segment-type-11
    +--ro local-ipv6-address?      inet:ipv6-address
    +--ro remote-ipv6-address?      inet:ipv6-address

```

3.2. Automatic Steering

Yang Tree for Automatic Steering with example of ipv4-unicast SAFI

```

module: ietf-bgp-srte
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
+--ro routes
  +--ro route* [prefix neighbor add-path-id]
    +--ro prefix          union
    +--ro neighbor        inet:ip-address
    +--ro add-path-id      uint32
    +--ro automatic-steering
      +--ro color?        -> /rt:routing/control-plane-protocols/control
      +--ro end-point?    -> /rt:routing/control-plane-protocols/control
      +--ro co-flag?      enumeration
      +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control
  augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
  +--ro routes
    +--ro route* [prefix neighbor add-path-id]
      +--ro prefix          union
      +--ro neighbor        inet:ip-address
      +--ro add-path-id      uint32
      +--ro automatic-steering
        +--ro color?        -> /rt:routing/control-plane-protocols/control
        +--ro end-point?    -> /rt:routing/control-plane-protocols/control
        +--ro co-flag?      enumeration
        +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control
    augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
    +--ro routes
      +--ro route* [prefix neighbor add-path-id]
        +--ro prefix          union
        +--ro neighbor        inet:ip-address
        +--ro add-path-id      uint32
        +--ro automatic-steering
          +--ro color?        -> /rt:routing/control-plane-protocols/control
          +--ro end-point?    -> /rt:routing/control-plane-protocols/control
          +--ro co-flag?      enumeration
          +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control

```

```
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
+--ro routes
  +--ro route* [prefix neighbor add-path-id]
    +--ro prefix          union
    +--ro neighbor        inet:ip-address
    +--ro add-path-id      uint32
    +--ro automatic-steering
      +--ro color?        -> /rt:routing/control-plane-protocols/control
      +--ro end-point?    -> /rt:routing/control-plane-protocols/control
      +--ro co-flag?      enumeration
      +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
+--ro routes
  +--ro route* [rd prefix neighbor add-path-id]
    +--ro rd              rt-types:route-distinguisher
    +--ro prefix          union
    +--ro neighbor        inet:ip-address
    +--ro add-path-id      uint32
    +--ro automatic-steering
      +--ro color?        -> /rt:routing/control-plane-protocols/control
      +--ro end-point?    -> /rt:routing/control-plane-protocols/control
      +--ro co-flag?      enumeration
      +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/bgp
+--ro routes
  +--ro route* [rd prefix neighbor add-path-id]
    +--ro rd              rt-types:route-distinguisher
    +--ro prefix          union
    +--ro neighbor        inet:ip-address
    +--ro add-path-id      uint32
    +--ro automatic-steering
      +--ro color?        -> /rt:routing/control-plane-protocols/control
      +--ro end-point?    -> /rt:routing/control-plane-protocols/control
      +--ro co-flag?      enumeration
      +--ro binding-sid?  -> /rt:routing/control-plane-protocols/control
```

...

[4.](#) Yang Module

<CODE BEGINS> file "ietf-bgp-srte@2019-07-07.yang"

```
module ietf-bgp-srte {
```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-srte";
```

```
prefix "ietf-bgp-srte";

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

import ietf-routing {
  prefix "rt";
}

import ietf-inet-types {
  prefix inet;
}

import ietf-bgp {
  prefix bgp;
}

import ietf-bgp-types {
  prefix bgp-types;
}

import ietf-srv6-types {
  prefix srv6-types;
}

organization
  "IETF Inter-Domain Routing Working Group";

contact
  "Inter-Domain working group - idr@ietf.org";

description
  "This YANG module defines a data model to configure and
  manage segment routing extensions in BGP."
```

Terms and Acronyms

AF : Address Family

BGP (bgp) : Border Gateway Protocol

SR : Segment Routing

SID : Segment Identifier

SRv6 : Segment Routing with IPv6 Data plane

VPN : Virtual Private Network

VRF : Virtual Routing and Forwarding

";

```
revision 2018-06-26 {  
  description  
    "Initial revision" ;  
  reference "";  
}
```

```
//  
// New identities and typedefs for SR extensions  
//
```

```
// SR Policy SAFI identities  
identity IPV4_SRPOLICY {  
  base bgp-types:afi-safi-type;  
  description  
    "IPv4 SR Policy (AFI,SAFI = 1,73)";  
  reference "TBD";  
}
```

```
identity IPV6_SRPOLICY {  
  base bgp-types:afi-safi-type;  
  description  
    "IPv6 SR Policy (AFI,SAFI = 2,73)";  
  reference "TBD";  
}
```

```
typedef segment-type {
```

```

type enumeration {
    enum segment-type-1 {
        value 1;
        description "SR-MPLS Label";
    }
    enum segment-type-2 {
        value 2;
        description "SRv6 SID";
    }
    enum segment-type-3 {
        value 3;
        description "IPv4 Prefix with optional SR Algorithm";
    }
    enum segment-type-4 {
        value 4;
        description "IPv6 Global Prefix with optional SR Algorithm for SR-MPLS";
    }
}

```

```

}
enum segment-type-5 {
    value 5;
    description "IPv4 Prefix with Local Interface ID";
}
enum segment-type-6 {
    value 6;
    description "IPv4 Addresses for link endpoints as Local, Remote pair";
}
enum segment-type-7 {
    value 7;
    description "IPv6 Prefix and Interface ID for link endpoints as Local,
        Remote pair for SR-MPLS";
}
enum segment-type-8 {
    value 8;
    description "IPv6 Addresses for link endpoints as Local, Remote pair for
        SR-MPLS";
}
enum segment-type-9 {
    value 9;
    description "IPv6 Global Prefix with optional SR Algorithm for SRv6";
}
enum segment-type-10 {
    value 10;
}

```

```

        description "IPv6 Prefix and Interface ID for link endpoints as Local,
            Remote pair for SRv6";
    }
    enum segment-type-11 {
        value 11;
        description "IPv6 Addresses for link endpoints as Local, Remote pair fo
            SRv6";
    }
}
description "SR segment type";
}

// Sid type union
typedef sid-type {
    type union {
        type rt-types:mpls-label;
        type srv6-types:srv6-sid;
    }
    description "Type definition for Segment Identifier. This is
        a union type which can be either a SR MPLS SID in the
        form of a label, or a SRv6 SID in the form of
        an IPv6 address.";
    reference "TBD";
}

```

```

}

//
// SR Policy Related Groupings
//
//Color and Endpoint of the SR Policy
grouping sr-policy-color-endpoint {
    description "Common grouping for SR Policy Color and
        Endpoint";
    leaf color {
        type uint32;
        description "Color of the policy";
    }

    leaf end-point {
        type inet:ip-address;
        description "Endpoint of the policy";
    }
}

```

```

}
// Authorized colors for On Demand SR Policy programming
grouping sr-odn-auth-colors {
  description
    "Authorized colors for On Demand (dynamic) SR Policies
    towards BGP nexthops";
  container authorized-colors {
    config false;
    description
      "Authorized colors for On Demand (dynamic) SR policies
      towards BGP nexthops";
    list colors {
      key "color";
      description "List of SR Policy Colors";
      leaf color {
        type uint32;
        description "Color value";
      }
    }
  }
}

grouping sr-policy-cmn-state {
  description "Common state parameters applicable to
  SR Policies";
  leaf policy-state {
    type enumeration {
      enum UP {
        description "SR Policy state UP";
      }
    }
  }
}

```

```

    enum DOWN {
      description "SR Policy state DOWN";
    }
  }
  description "SR Policy forwarding state";
}

leaf binding-sid {
  type sid-type;
  description "Binding SID of the SR Policy";
}

```

```

leaf steering-disabled {
    type empty;
    description "This attribute is set if steering
                is disabled on this SR policy";
}

leaf ref-count {
    type uint32;
    description "Count of routes steering over this policy";
}
}

//
// SR Policy State grouping
//
grouping sr-policy-state {
    description "SR Policy State";
    container policy-state {
        config false;
        description "SR Policy State";
        list sr-policy {
            key "color end-point";
            description "List of SR Policies";

            uses sr-policy-color-endpoint;

            // State of the SR Policy in BGP
            uses sr-policy-cmn-state;
        }
    }
}

grouping sr-exp-policy-cp-state {
    description "State of BGP signaled SR Policy (explicit)
                candidate paths";
    container explicit-policies {

```

```

    config false;
    description "BGP signaled explicit SR Policies";
    list sr-policy {
        key "distinguisher color end-point";

```

```

description "List of BGP signaled explicit SR Policies";
leaf distinguisher {
    type uint32;
    description "Distinguisher of the SR Policy
                candidate path";
}

uses sr-policy-color-endpoint;

leaf preference {
    type uint32;
    description "Preference of the SR Policy candidate path";
}

container explicit-binding-sid {
    description "Explicitly supplied Binding SID
                for this policy";
    leaf binding-sid {
        type sid-type;
        description "Binding SID value";
    }
    leaf strict {
        type boolean;
        description "Boolean indicating that the node
                    must use only the supplied Binding SID
                    for this SR Policy.
                    reference: TBD";
    }
    leaf drop-on-invalid {
        type boolean;
        description "Boolean to indicate drop upon invalid
                    policy, behavior. This overwrites the
                    default behavior of fallback to IGP path
                    , when SR Policy is (or becomes) invalid.
                    reference: TBD";
    }
}

leaf usable {
    type boolean;
    description "Boolean to indicate that the SR Policy is
                usable on this node.
                reference: TBD";
}

```

```
    leaf registered {
      type boolean;
      description "Boolean to indicate that the SR policy
                  is registered with policy manager to
                  install the corresponding forwarding entry";
    }

    uses segment-lists;
    // TODO: Segment Lists and other parameters from SR Policy model
    //       to be imported here.
  }
}

grouping segment-lists {
  description
    "Segment lists grouping";
  container segment-lists {
    description "Segment-lists properties";

    list segment-list {
      key "weight";
      description "Segment-list";
      leaf weight {
        type uint32;
        description "Segment-list weight";
      }
      container segments {
        description
          "Segments for given segment list";

        list segment {
          key "index";
          description "Segment/hop at the index";
          uses segment-properties;
        }
      }
    }
  }
}

grouping segment-properties {
  description "Segment properties grouping";
  leaf index {
    type uint32;
    description "Segment index";
  }
}
```

}

```
leaf type {
  type segment-type;
  description "Segment type";
}
container segment-types {
  description "Types of segments";
  container segment-type-1 {
    description
      "Segment declared by MPLS label";
    leaf sid-value {
      type rt-types:mpls-label;
      description "MPLS label value";
    }
  }
  container segment-type-2 {
    description
      "Segment declared by SRv6 SID value";
    leaf sid-value {
      type srv6-types:srv6-sid;
      description "SRv6 SID value";
    }
  }
  container segment-type-3 {
    description
      "Segment declared by IPv4 Prefix with optional SR Algorithm";
    leaf ipv4-address {
      type inet:ipv4-address;
      description "Segment IPv4 address";
    }
    leaf algorithm {
      type uint8;
      description "Prefix SID algorithm identifier";
    }
  }
  container segment-type-4 {
    description
      "Segment declared by IPv6 Global Prefix with optional
      SR Algorithm for SR-MPLS";
    leaf ipv6-address {
      type inet:ipv6-address;
```

```

        description "Segment IPv6 address";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-5 {

```

```

    description
        "Segment declared by IPv4 Prefix with Local Interface ID";
    leaf ipv4-address {
        type inet:ipv4-address;
        description "Node IPv4 address";
    }
    leaf interface-identifier {
        type uint32;
        description "local interface identifier";
    }
}
container segment-type-6 {
    description
        "Segment declared by IPv4 Addresses for link endpoints
        as Local, Remote pair";
    leaf local-ipv4-address {
        type inet:ipv4-address;
        description "Segment local IPv4 adjacency address";
    }
    leaf remote-ipv4-address {
        type inet:ipv4-address;
        description "Segment remote IPv4 adjacency address";
    }
}
container segment-type-7 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for
        link endpoints as Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Local link IPv6 address";
    }
    leaf local-interface-identifier {

```

```

        type uint32;
        description "Local interface identifier";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Remote link IPv6 address";
    }
    leaf remote-interface-identifier {
        type uint32;
        description "Remote interface identifier";
    }
}
container segment-type-8 {
    description
        "Segment declared by IPv6 Addresses for link endpoints as

```

```

        Local, Remote pair for SR-MPLS";
    leaf local-ipv6-address {
        type inet:ipv6-address;
        description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
        type inet:ipv6-address;
        description "Segment remote IPv6 adjacency address";
    }
}
container segment-type-9 {
    description
        "Segment declared by IPv6 Global Prefix with optional
        SR Algorithm for SRv6";
    leaf ipv6-address {
        type inet:ipv6-address;
        description "Segment IPv6 prefix";
    }
    leaf algorithm {
        type uint8;
        description "Prefix SID algorithm identifier";
    }
}
container segment-type-10 {
    description
        "Segment declared by IPv6 Prefix and Interface ID for

```

```

    link endpoints as Local, Remote pair for SRv6";
  leaf local-ipv6-address {
    type inet:ipv6-address;
    description "Local link IPv6 address";
  }
  leaf local-interface-identifier {
    type uint32;
    description "Local interface identifier";
  }
  leaf remote-ipv6-address {
    type inet:ipv6-address;
    description "Remote link IPv6 address";
  }
  leaf remote-interface-identifier {
    type uint32;
    description "Remote interface identifier";
  }
}
container segment-type-11 {
  description
    "Segment declared by IPv6 Addresses for link endpoints as
    Local, Remote pair for SRv6";
}

```

```

    leaf local-ipv6-address {
      type inet:ipv6-address;
      description "Segment local IPv6 adjacency address";
    }
    leaf remote-ipv6-address {
      type inet:ipv6-address;
      description "Segment remote IPv6 adjacency address";
    }
  }
}
grouping sr-odn-policies {
  description "SR On Demand (dynamic) SR Policies";
  container installed-policies {
    config false;
    description "BGP triggered On Demand (dynamic) SR Policies
      corresponding to the BGP nexthops";
  }
  list sr-policy {
    key "color end-point";
  }
}

```

```

        description "SR Policy list";
        uses sr-policy-color-endpoint;
    }
}

grouping sr-policy-steering-state {
    description "Per route Automatic Steering parameters";
    container automatic-steering {
        description "Per route Automatic Steering parameters";
        leaf color {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols/rt:control-plane-protoco
                    "bgp:bgp/bgp:global/ietf-bgp-srte:segment-routing/" +
                    "ietf-bgp-srte:policy-state/ietf-bgp-srte:sr-policy/" +
                    "ietf-bgp-srte:color";
            }
            description "Color of the SR Policy being used for
                Automatic Steering";
        }
        leaf end-point {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols/rt:control-plane-protoco
                    "bgp:bgp/bgp:global/ietf-bgp-srte:segment-routing/" +
                    "ietf-bgp-srte:policy-state/ietf-bgp-srte:sr-policy/" +
                    "ietf-bgp-srte:end-point";
            }
            description "End-point of the SR Policy being used
                for Automatic Steering";
        }
    }
}

```

```

}
leaf co-flag {
    type enumeration {
        enum 00 {
            description "Color-Only flag 00";
        }
        enum 01 {
            description "Color-Only flag 01";
        }
        enum 10 {
            description "Color-Only flag 10";
        }
    }
}

```

```

    }
    default "00";
    description "Color-Only (CO) flags applicable for
        Automatic Steering of this route";
}
leaf binding-sid {
    type leafref {
        path "/rt:routing/rt:control-plane-protocols/rt:control-plane-protoco
            "bgp:bgp/bgp:global/ietf-bgp-srte:segment-routing/" +
            "ietf-bgp-srte:policy-state/ietf-bgp-srte:sr-policy/" +
            "ietf-bgp-srte:binding-sid";
    }
    description "Binding SID of the SR Policy";
}
}
}
}

```

```

grouping route-key-leafs {
    description "Grouping for key leafs identifying a route";
    leaf prefix {
        type union {
            type inet:ip-prefix;
            type string;
        }
        description "BGP Prefix. This is a temp definition to
            cover ip-prefix and other NLRI formats.
            Import the type once defined in base
            BGP RIB model";
    }
    leaf neighbor {
        type inet:ip-address;
        description "BGP Neighbor";
    }
    leaf add-path-id {
        type uint32;
        description "Add-path ID";
    }
}

```

```

    }
}

grouping common-bgp-route-grouping {
    description "BGP route list" ;
}

```

```

    container routes {
        config false;
        description "BGP Route in local RIB";
        list route {
            key "prefix neighbor add-path-id";
            description "BGP route list";
            uses route-key-leafs;
        }
    }
}

grouping common-bgp-vpn-route-grouping {
    description "BGP route list" ;
    container routes {
        config false;
        description "BGP VPN Route in local RIB";
        list route {
            key "rd prefix neighbor add-path-id";
            description "Route List";

            leaf rd {
                type rt-types:route-distinguisher;
                description "Route Distinguisher";
            }
            uses route-key-leafs;
        }
    }
}

//
// BGP Specific Paramters
//
// Augment AF with route list
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgpglobal:bgp:afi-safis/bgpg:afi-safi/bgpg:ipv4-unicast" {
    description
        "Augment BGP SAFI route";
    uses common-bgp-route-grouping;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgpglobal:bgp:afi-safis/bgpg:afi-safi/bgpg:ipv6-unicast" {
    description
        "Augment BGP SAFI route";
}

```

```

    uses common-bgp-route-grouping;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv4-labeled-unicast"
    description
        "Augment BGP SAFI route";
    uses common-bgp-route-grouping;

}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv6-labeled-unicast"
    description
        "Augment BGP SAFI route";
    uses common-bgp-route-grouping;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:l3vpn-ipv4-unicast"
    description
        "Augment BGP SAFI route";
    uses common-bgp-vpn-route-grouping;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:l3vpn-ipv6-unicast"
    description
        "Augment BGP SAFI route";
    uses common-bgp-vpn-route-grouping;
}

// SR Policy Related
// On Demand authorized colors table
// SR Policy state data
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global" {
    description
        "Segment Routing parameters in BGP global model";
    container segment-routing {
        description "Segment Routing parameters";
        container on-demand-policies {
            description
                "Segment Routing On Demand Nexthop
                (ODN) SR Policies";
            uses sr-odn-auth-colors;
            uses sr-odn-policies;
        }
        uses sr-policy-state;
    }
}

```

Internet-Draft

Yang Model for BGP Segment Routing TE

January 2022

```
// Steering state in overlay BGP routes
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:ipv4-unicast/ietf-bgp-srte:routes/ietf-bgp-srte:rou
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:ipv6-unicast/ietf-bgp-srte:routes/ietf-bgp-srte:rou
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:ipv4-labeled-unicast/ietf-bgp-srte:routes/ietf-bgp-
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:ipv6-labeled-unicast/ietf-bgp-srte:routes/ietf-bgp-
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:l3vpn-ipv4-unicast/ietf-bgp-srte:routes/ietf-bgp-sr
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
}
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +
    "bgp:bgp/bgp:global/bgp:afi-safis/" +
    "bgp:afi-safi/bgp:l3vpn-ipv6-unicast/ietf-bgp-srte:routes/ietf-bgp-sr
description
    "Augment BGP SAFI route with steering info";
uses sr-policy-steering-state;
```

```
}
```

```
// BGP Signaled SR Policy explicit candidate paths state  
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +  
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi" {  
    description "Augment IPv4 SR Policy SAFI list entry";
```

```
    container ipv4-srpolicy {  
        when "../afi-safi-name = 'bgp-types:IPV4_SRPOLICY'" {  
            description  
                "Include this container for IPv4 SR Policy specific  
                configuration";  
        }  
        description "IPv4 SR Policy specific parameters";  
        uses sr-exp-policy-cp-state;  
    }  
}  
  
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/" +  
    "bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi" {  
    description "Augment IPv6 SR Policy SAFI list entry";  
    container ipv6-srpolicy {  
        when "../afi-safi-name = 'bgp-types:IPV6_SRPOLICY'" {  
            description  
                "Include this container for IPv6 SR Policy specific  
                configuration";  
        }  
        description "IPv6 SR Policy specific parameters";  
        uses sr-exp-policy-cp-state;  
    }  
}  
}  
<CODE ENDS>
```

[5. Contributors](#)

Dhanendra Jain
Cisco Systems
US

Email: dhanendra.ietf@gmail.com

Zhichun Jiang
Cisco Systems
US

Email: zcjiang@tencent.com

Zafar Ali
Cisco Systems
US

Email: zali@cisco.com

Deevi, et al.

Expires July 26, 2022

[Page 25]

Internet-Draft

Yang Model for BGP Segment Routing TE

January 2022

Sharmila Palani
Cisco Systems
US

Email: spalani@cisco.com

[6.](#) IANA Considerations

[7.](#) Security Considerations

The transport protocol used for sending the BGP Segment Routing data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

This draft does not change any underlying security issues inherent in [\[I-D.ietf-idr-bgp-model\]](#).

[8.](#) Acknowledgements

TBD.

[9.](#) References

[9.1.](#) Normative References

[I-D.ietf-idr-bgp-model]

Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", [draft-ietf-idr-](#)

[bgp-model-10](#) (work in progress), November 2020.

[I-D.ietf-idr-segment-routing-te-policy]

Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Rosen, E., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", [draft-ietf-idr-segment-routing-te-policy-11](#) (work in progress), November 2020.

[I-D.ietf-spring-segment-routing-policy]

Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", [draft-ietf-spring-segment-routing-policy-09](#) (work in progress), November 2020.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

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

[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>>.

[9.2](#). Informative References

- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D.,
Matsushima, S., and Z. Li, "SRv6 Network Programming",
[draft-ietf-spring-srv6-network-programming-28](#) (work in
progress), December 2020.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
[RFC 7951](#), DOI 10.17487/RFC7951, August 2016,
<<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017,
<<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of
Documents Containing YANG Data Models", [BCP 216](#), [RFC 8407](#),
DOI 10.17487/RFC8407, October 2018,
<<https://www.rfc-editor.org/info/rfc8407>>.

Authors' Addresses

Krishna Deevi
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: kdeevi@cisco.com

Kamran Raza
Cisco
2000 Innovation Drive

Kanata, ON K2K-3E8
CA

Email: skraza@cisco.com

Kausik Majumdar
Commscope

Email: kausik.majumdar@comscope.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Zhichun Jiang
Tencent

Email: zcjiang@tencent.com