SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 November 2026

T. Saleh, Ed.
K. Raza
Cisco Systems
Z. Shunwan
Huawei Technologies
S. Matsushima
SoftBank
V. Beeram
Juniper Networks

23 May 2026

YANG Data Model for Segment Routing Policy
draft-ietf-spring-sr-policy-yang-07

Abstract

This document defines a YANG data model for Segment Routing (SR) Policy that can be used for configuring, instantiating, and managing SR policies. The model is generic and applies equally to the MPLS and SRv6 instantiations of SR policies.

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 24 November 2026.

Table of Contents

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] defines mechanisms to manage network devices. YANG [RFC7950] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for NETCONF.

Segment Routing (SR), as defined in [RFC8402], allows a headend node to steer a packet flow along any topological path and/or service chain. The headend node is said to steer a flow into a Segment Routing Policy (SR Policy). An SR policy is a framework that enables instantiation of an ordered list of segments on a node for implementing a policy [RFC9256].

This document introduces a YANG data model for SR policy framework for instantiating, configuring and managing SR policies along with its attributes. It is also expected that other companion models, such as BGP SR Policy [I-D.ietf-idr-segment-routing-te-policy], will be defined and/or augmented accordingly in their respective areas.

This model defines the following constructs for managing an SR policy:

* Configuration

* Operational State

* Notifications

* Executables (Actions)

This document expects and requires the reader to be well familiar with the concepts and constructs of an SR policy [RFC9256] as well as the YANG modeling language and its presentation [RFC7950].

This document uses the YANG terminology defined in Section 3 of [RFC7950].

The meanings of the symbols in the YANG module tree diagrams are defined in [RFC8340].

2. Specification of Requirements

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Building Blocks

Before looking into the YANG model for SR policy, it is important to recall and highlight the major building blocks and constructs that constitute and contribute to an SR policy, as described in [RFC9256].

* policy: specifies constructs to allow a headend node to setup SR path(s) as an ordered list of segments for a given color and endpoint. The endpoint and the color are used to automate the steering of service or transport routes on an SR Policy. For a given headend, the key for an SR policy is (color, endpoint) where endpoint is an IP address that could be also NULL.

* candidate-path: is the unit for signalling of an SR Policy to a headend via protocols (such as PCEP, BGP, CLI etc.). A candidate path is either dynamic or explicit type, where an explicit candidate path is associated with one or more segment-lists and dynamic candidate path expresses optimization objectives and set of constraints. An SR Policy is associated with one or more candidate paths and the preference of the candidate path is used to select the best candidate path for an SR Policy. A candidate path is valid if it is usable (e.g. when its constituents SIDs are reachable). An "active" candidate path is the selected path (for forwarding) that is valid and determined to be the best path of the SR Policy. A candidate path is keyed by (protocol-origin, originator, discriminator).

* segment-list: specifies ordered list of segments to traverse, where a segment can be specified in various forms (refer section 4 of [RFC9256]). The list is sorted by the index of the segment. A segment-list is used and referred by an explicit type of candidate-path. A segment-list is keyed by its name.

* binding-sid: An SR policy is associated with a BSID to provide benefits of scaling, network opacity and service independence.

4. YANG Model

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.

For SR policy YANG specification, this document defines following new YANG modules:

Module Name	Purpose
ietf-sr-policy-types	defines common and basic types related to an SR policy and related constructs
ietf-sr-policy	defines the model for SR policy instantiation, configuration, and management

Table 1

4.1. Prefixes in Data Node Names

The tree diagrams and data node names in this document use prefixes from the following external modules.

Prefix	YANG Module	Reference
inet	ietf-inet-types	[RFC6991]
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
srv6-types	ietf-srv6-types	[I-D.ietf-spring-srv6-yang]
te-types	ietf-te-types	[I-D.ietf-teas-rfc8776-update]
yang	ietf-yang-types	[RFC6991]

Table 2: Prefixes and Corresponding YANG Modules

4.2. Types and Definitions

SR policy common types and definitions are defined in the new module "ietf-sr-policy-types". The main types defined in this module include:

* dataplane-type: A union to specify MPLS or IPv6 as the dataplane type for SR.

* sid-value-type: A Union to specify SID value for SR-MPLS or SRv6 type.

* binding-sid-alloc-mode: Enum to define explicit or dynamic alloc mode types for a BSID.

* protocol-origin-type: The base identity along with its children specify protocol origin (e.g. PCEP) for an SR policy.

* explicit-binding-sid-rule-type: Enum to specify BSID alloc enforcement/rule when doing explicit alloc request.

* binding-sid-oper-state: An Enum representing various operational states for a BSID.

* policy-admin-state: An Enum for admin state of an SR policy.

* policy-oper-state: An Enum for operational state of an SR policy.

* segment-type: The base identity along with its children define various types for a "segment" of a Segment list.

* candidate-path-not-selected-reason: The base identity along with its children to specify reason for not selecting a candidate path as the best/active path.

* policy-down-reason: The base identity along with its children to specify reason for a policy becoming (or remaining) operationally down.

* binding-sid-unavailable-reason: The base identity along with its children to specify reason for a BSID’s unavailability.

The associated YANG specification for this module is captured in Section 7.1.

4.3. SR Policy

The base SR policy model is captured in ietf-sr-policy module. This base module augments "/rt:routing" and specifies the configuration, operational state, executables/rpcs, and notification events required to manage SR policies.

The associated YANG specification for this module is captured in Section 7.2.

4.3.1. Configuration

In terms of configuration hierarchy, SR policy configuration tree has following three main areas:

* optimization-templates: container that defines list of templates per color. The template is used to define a set of optimization objectives and constraints used to instantiate an SR policy on-demand.

* attributes: container that defines common constructs that could be used across policies. Examples of such a construct include segment-lists, affinity-map etc. In future revision of this document, it is expected that this container will have more constructs defined.

* policies: container that defines list of policies with their attributes such as BSID, candidate-paths etc.

Following diagram depicts high level yang organization and hierarchy for an SR policy specification:

```
segment-routing
   traffic-engineering
      +--rw optimization-templates
      |  +--rw optimization-template* [color]
      |     +--rw color              uint32
      |     +--rw candidate-paths
      |        +--rw candidate-path* [discriminator]
      |        |  ....
      |
      + attributes
      |   + affinity-map
      |   |  ....
      |   |
      |   + segment-lists
      |   |    segment-list* [name]
      |   |        segments
      |   |            segment* [index]
      |   |                ...
      |   + explicit-binding-sid-rules
      |       ...
      |
      + policies
           policy* [color endpoint]
              + ...
              |
              + binding-sid
              | ...
              |
              + candidate-paths
                  candidate-path*
                    [protocol-origin originator-asn
                     originator-node-address discriminator]
                     + ...
                     |
                     + type
                         + explicit
                         |   segment-lists
                         |        segment-list* [ref]
                         |             ...
                         + dynamic
                       constraints
                                 ...
```

Figure 1: SR Policy - Hierarchy

The complete YANG tree can be found in Figure 2.

Please take note of the following important points regarding the configurations:

* This model supports both MPLS and SRv6 dataplane for SR -- i.e. items like segments and BSID can be defined as MPLS label or SRv6 SIDs.

* Specification of a segment supports all the types defined in SR policy base specification document

* The above model supports explicit BSID specification on SR policy level as the main mode of specification. The model also allows explicit BSID per candidate-path as an if-feature capability that is optional for implementations

* The above model will be extended in future revisions of this document to enhance constraints specification for dynamic type of candidate-path, as well as add traffic-steering controls.

4.3.2. State

As per NMDA model, the state related to configuration items specified in earlier Section 4.3.1 can be retrieved from the same tree. This section defines the other operational state items related to SR policy.

In addition to configured state, the operational state corresponding to the SR policy includes:

* policy operational state

* policy up/down timestamps

* policy BSID info such as alloc mode, actual value in-use, operational state, and forwarding stats

* Per candidate-path info such as:

  - Whether candidate-path is the best candidate-path

  - In case of non-best, the reason for such non-selection

  - Type of candidate-path - explicit or dynamic

- Per segment-list information - such as validity of the segment-list, as well as forwarding state for a valid segment-list. The forwarding state is represented in terms of per forwarding path info that includes nexthop address, outgoing interface, protection information, and encapsulation (label stack or SRv6 SID stack) etc.

The complete YANG tree can be found in Figure 2.

4.3.3. Notification

This model defines a list of notifications to inform an operator of important events detected regarding an SR policy. These events include events related to:

* policy status: policy operational state changes

* Candidate-path active status and changes

* Explicit Binding SID collision/unavailability events

Section 6 captures the YANG tree for the notification.

5. Pending Items

Following are the items that will be addressed in future revisions of this document:

* Configuration and Specification of:

  - Traffic steering over SR policy

  - Spray policy

* Executables (RPC actions)

6. YANG Tree

Using the building blocks described in Section 3, following is the complete graphical representation of the data model for SR policy:

```
module: ietf-sr-policy

  augment /rt:routing:
    +--rw segment-routing
       +--rw traffic-engineering
          +--rw optimization-templates
          |  +--rw optimization-template* [color]
```

```
|     +--rw color              uint32
|     +--rw candidate-paths
|        +--rw candidate-path* [discriminator]
|           +--rw discriminator              uint32
|           +--rw optimization-objectives
|           |  +--rw minimize-metric
|           |     +--rw metric-type?   identityref
|           |     +--rw margin
|           |        +--rw type?
|           |        |       sr-policy-types:margin-type
|           |        +--rw value?            uint32
|           |        +--ro computed-value?   uint32
|           +--rw constraints
|              +--rw srlg
|              |  +--rw exclude-any*   string
|              +--rw affinity
|              |  +--rw exclude-any*   string
|              |  +--rw include-any*   string
|              |  +--rw include-all*   string
|              +--rw ip-address
|              |  +--rw exclude*   inet:ip-address
|              +--rw upper-bounds
|              |  +--rw cumulative-metric
|              |  |  +--rw metric-type?   identityref
|              |  |  +--rw value?         uint32
|              |  +--rw max-sids?            uint32
|              +--rw segment-rules
|              |  +--rw sid-protection-eligibility?
|              |          sr-policy-types:
|              |            sid-protection-eligibility-type
|              +--rw max-sids?        uint32
|  +--rw attributes
|  |  +--rw segment-lists
|  |  |  +--rw segment-list* [name]
|  |  |     +--rw name        string
|  |  |     +--rw segments
|  |  |        +--rw segment* [index]
|  |  |           +--rw index       uint32
|  |  |           +--rw type        identityref
|  |  |           +--rw sr-mpls
|  |  |           |  +--rw Type-A
|  |  |           |  |  +--rw value?   rt-types:mpls-label
|  |  |           |  +--rw Type-C
|  |  |           |  |  +--rw ipv4-address?
|  |  |           |  |  |       inet:ipv4-address
|  |  |           |  |  +--rw algorithm?      uint8
|  |  |           |  +--rw Type-D
|  |  |           |  |  +--rw ipv6-address?
```

```
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw algorithm?      uint8
|  |           |  +--rw Type-E
|  |           |  |  +--rw ipv4-address?
|  |           |  |  |       inet:ipv4-address
|  |           |  |  +--rw interface-identifier?   uint32
|  |           |  +--rw Type-F
|  |           |  |  +--rw local-ipv4-address?
|  |           |  |  |       inet:ipv4-address
|  |           |  |  +--rw remote-ipv4-address?
|  |           |  |          inet:ipv4-address
|  |           |  +--rw Type-G
|  |           |  |  +--rw local-ipv6-address?
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw local-interface-identifier?
|  |           |  |  |       uint32
|  |           |  |  +--rw remote-ipv6-address?
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw remote-interface-identifier?
|  |           |  |          uint32
|  |           |  +--rw Type-H
|  |           |     +--rw local-ipv6-address?
|  |           |     |       inet:ipv6-address
|  |           |     +--rw remote-ipv6-address?
|  |           |             inet:ipv6-address
|  |           +--rw srv6
|  |           |  +--rw Type-B
|  |           |  |  +--rw value?
|  |           |  |  |       srv6-types:srv6-sid
|  |           |  |  +--rw sid-behavior?    identityref
|  |           |  |  +--rw sid-structure
|  |           |  |     +--rw locator-block-length?
|  |           |  |     |       srv6-types:
|  |           |  |     |         srv6-locator-block-len
|  |           |  |     +--rw locator-node-length?
|  |           |  |     |       srv6-types:
|  |           |  |     |         srv6-locator-node-len
|  |           |  |     +--rw function-length?
|  |           |  |     |       srv6-types:
|  |           |  |     |         srv6-sid-func-len
|  |           |  |     +--rw argument-length?
|  |           |  |             srv6-types:srv6-sid-arg-len
|  |           |  +--rw Type-I
|  |           |  |  +--rw ipv6-address?
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw algorithm?      uint8
|  |           |  +--rw Type-J
|  |           |  |  +--rw local-ipv6-address?
```

```
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw local-interface-identifier?
|  |           |  |  |       uint32
|  |           |  |  +--rw remote-ipv6-address?
|  |           |  |  |       inet:ipv6-address
|  |           |  |  +--rw remote-interface-identifier?
|  |           |  |          uint32
|  |           |  +--rw Type-K
|  |           |     +--rw local-ipv6-address?
|  |           |     |       inet:ipv6-address
|  |           |     +--rw remote-ipv6-address?
|  |           |             inet:ipv6-address
|  |           +--rw validate?   boolean
|  +--rw explicit-binding-sid-rules* [index]
|     +--rw index    uint32
|     +--rw rule
|             sr-policy-types:
|               explicit-binding-sid-rule-type
+--rw policies
   +--rw policy* [headend color endpoint]
      +--rw name?               sr-policy-types:name-type
      +--rw headend             inet:ip-address
      +--rw color               sr-policy-types:color-type
      +--rw endpoint            inet:ip-address
      +--rw description?        string
      +--rw admin-state?
      |       sr-policy-types:policy-admin-state
      +--rw priority?           uint8
      +--rw profile-id?         uint32
      +--rw bandwidth
      |  +--rw requested?   uint64
      |  +--ro reserved?    uint64
      +--ro oper-state?
      |       sr-policy-types:policy-oper-state
      +--ro transition-count?   uint32
      +--ro up-time?            yang:date-and-time
      +--ro down-time?          yang:date-and-time
      +--rw binding-sid
      |  +--rw mpls-sid
      |  |  +--rw value?
      |  |  |       sr-policy-types:mpls-sid-value-type
      |  |  +--ro alloc-mode?
      |  |  |       sr-policy-types:binding-sid-alloc-mode
      |  |  +--ro allocated-sid?
      |  |  |       sr-policy-types:sid-value-type
      |  |  +--ro oper-state?
      |  |          sr-policy-types:binding-sid-oper-state
      |  +--rw srv6-sids
```

```
|     +--rw srv6-sid* [index]
|        +--rw index            uint32
|        +--rw locator-name?
|        |       srv6-types:srv6-locator-name-type
|        +--rw value?           srv6-types:srv6-sid
|        +--rw sid-behavior?    identityref
|        +--rw sid-structure
|        |  +--rw locator-block-length?
|        |  |       srv6-types:srv6-locator-block-len
|        |  +--rw locator-node-length?
|        |  |       srv6-types:srv6-locator-node-len
|        |  +--rw function-length?
|        |  |       srv6-types:srv6-sid-func-len
|        |  +--rw argument-length?
|        |          srv6-types:srv6-sid-arg-len
|        +--ro alloc-mode?
|        |       sr-policy-types:
|        |         binding-sid-alloc-mode
|        +--ro allocated-sid?
|        |       sr-policy-types:sid-value-type
|        +--ro oper-state?
|                sr-policy-types:
|                  binding-sid-oper-state
+--ro counters
|  +--ro pkts?     yang:counter64
|  +--ro octets?   yang:counter64
+--rw candidate-paths
   +--rw candidate-path*
           [
             protocol-origin
             originator-asn
             originator-node-address
             discriminator]
      +--rw protocol-origin
      |       identityref
      +--rw originator-asn                    uint32
      +--rw originator-node-address
      |       sr-policy-types:node-address-type
      +--rw discriminator                     uint32
      +--rw preference                        uint32
      +--rw name?                             string
      +--rw description?                      string
      +--rw drop-upon-invalid?                boolean
      +--rw admin-state?                      boolean
      +--rw binding-sid
      |       {capability-candidate-path-binding-sid}?
      |  +--rw mpls-sid
      |  |  +--rw value?
```

```
|  |          sr-policy-types:
|  |            mpls-sid-value-type
|  +--rw srv6-sids
|     +--rw srv6-sid* [index]
|        +--rw index            uint32
|        +--rw locator-name?
|        |       srv6-types:
|        |         srv6-locator-name-type
|        +--rw value?
|        |       srv6-types:srv6-sid
|        +--rw sid-behavior?    identityref
|        +--rw sid-structure
|           +--rw locator-block-length?
|           |       srv6-types:
|           |         srv6-locator-block-len
|           +--rw locator-node-length?
|           |       srv6-types:
|           |         srv6-locator-node-len
|           +--rw function-length?
|           |       srv6-types:
|           |         srv6-sid-func-len
|           +--rw argument-length?
|                   srv6-types:srv6-sid-arg-len
+--rw (type)?
|  +--:(explicit)
|  |  +--rw segment-lists
|  |     +--rw segment-list* [name-ref]
|  |        +--rw name-ref    leafref
|  |        +--rw weight?     uint32
|  |        +--ro is-valid?   boolean
|  |        +--ro counters
|  |           +--ro pkts?     yang:counter64
|  |           +--ro octets?   yang:counter64
|  +--:(dynamic)
|  |  +--rw segment-list
|  |  |  +--rw segment-list-dataplane-type
|  |  |  |       sr-policy-types:dataplane-type
|  |  |  +--ro is-valid?
|  |  |  |       boolean
|  |  |  +--ro counters
|  |  |     +--ro pkts?     yang:counter64
|  |  |     +--ro octets?   yang:counter64
|  |  +--rw optimization-objectives
|  |  |  +--rw minimize-metric
|  |  |     +--rw metric-type?   identityref
|  |  |     +--rw margin
|  |  |        +--rw type?
|  |  |        |       sr-policy-types:
```

```
|  |  |        |         margin-type
|  |  |        +--rw value?            uint32
|  |  |        +--ro computed-value?   uint32
|  |  +--rw constraints
|  |     +--rw srlg
|  |     |  +--rw exclude-any*   string
|  |     |  +--rw include-any*   string
|  |     |  +--rw include-all*   string
|  |     +--rw affinity
|  |     |  +--rw exclude-any*   string
|  |     |  +--rw include-any*   string
|  |     |  +--rw include-all*   string
|  |     +--rw ip-address
|  |     |  +--rw exclude*
|  |     |  |       inet:ip-address
|  |     |  +--rw include-any*
|  |     |  |       inet:ip-address
|  |     |  +--rw include-all*
|  |     |          inet:ip-address
|  |     +--rw upper-bounds
|  |     |  +--rw cumulative-metric
|  |     |  |  +--rw metric-type?   identityref
|  |     |  |  +--rw value?         uint32
|  |     |  +--rw max-sids?            uint32
|  |     +--rw segment-rules
|  |     |  +--rw sid-algorithm?
|  |     |  |       sr-policy-types:
|  |     |  |         sid-algorithm-type
|  |     |  +--rw sid-protection-eligibility?
|  |     |          sr-policy-types:
|  |     |            sid-protection-
|  |     |              eligibility-type
|  |     +--rw disjoint-path
|  |     |  +--rw disjointness-type?
|  |     |  |       te-types:
|  |     |  |         te-path-disjointness
|  |     |  +--rw association-group-id
|  |     |  |       uint16
|  |     |  +--rw association-source
|  |     |  |       inet:ip-address
|  |     |  +--rw association-extended-id?
|  |     |          yang:hex-string
|  |     +--rw max-sids?        uint32
|  +--:(composite)
|     +--rw constituent-policies
|        +--rw constituent-policy* [color]
|           +--rw color
|           |       sr-policy-types:color-type
```

```
                        |           +--rw weight?   uint32
                        +--ro is-best-candidate-path?          boolean
                        +--ro non-selection-reason?
                        |       identityref
                        +--ro is-active?                       boolean
                        +--ro is-valid?                        boolean
                        +--ro computed-sid-list
                        |  +--ro (dataplane-type)?
                        |     +--:(mpls)
                        |     |  +--ro labels* [index]
                        |     |     +--ro index    uint32
                        |     |     +--ro label?   rt-types:mpls-label
                        |     +--:(srv6)
                        |        +--ro sids* [index]
                        |           +--ro index    uint32
                        |           +--ro sid?     srv6-types:srv6-sid
                        +--ro forwarding-paths
                           +--ro forwarding-path* [path-id]
                              +--ro path-id              uint8
                              +--ro next-hop-address?
                              |       inet:ip-address
                              +--ro next-hop-table-id?   uint32
                              +--ro interface?
                              |       if:interface-ref
                              +--ro is-protected?        boolean
                              +--ro backup-path-id?      uint8
                              +--ro is-pure-backup?      boolean
                              +--ro weight?              uint32
                              +--ro sid-list
                              |  +--ro (dataplane-type)?
                              |     +--:(mpls)
                              |     |  +--ro labels* [index]
                              |     |     +--ro index    uint32
                              |     |     +--ro label?
                              |     |             rt-types:mpls-label
                              |     +--:(srv6)
                              |        +--ro sids* [index]
                              |           +--ro index    uint32
                              |           +--ro sid?
                              |                   srv6-types:srv6-sid
                              +--ro counters
                                 +--ro pkts?     yang:counter64
                                 +--ro octets?   yang:counter64

    notifications:
      +---n sr-policy-oper-state-change-event
      |  +--ro policy-name-ref?         leafref
      |  +--ro policy-color-ref?        leafref
```

```
      |  +--ro policy-endpoint-ref?     leafref
      |  +--ro policy-new-oper-state?
      |  |       sr-policy-types:policy-oper-state
      |  +--ro policy-down-reason?      identityref
      +---n sr-policy-candidate-path-change-event
      |  +--ro policy-name-ref?       leafref
      |  +--ro policy-color-ref?      leafref
      |  +--ro policy-endpoint-ref?   leafref
      |  +--ro existing-preference?   uint32
      |  +--ro new-preference?        uint32
      +---n sr-policy-binding-sid-unavailable-event
      |  +--ro policy-name-ref?                              leafref
      |  +--ro policy-color-ref?                             leafref
      |  +--ro policy-endpoint-ref?                          leafref
      |  +--ro (dataplane-type)?
      |  |  +--:(mpls)
      |  |  |  +--ro policy-mpls-binding-sid-value-ref?   leafref
      |  |  |          {capability-candidate-path-binding-sid}?
      |  |  +--:(srv6)
      |  |     +--ro policy-srv6-binding-sid-value-ref?   leafref
      |  |             {capability-candidate-path-binding-sid}?
      |  +--ro reason?                                       identityref
      +---n sr-policy-candidate-path-binding-sid-mismatch-event
         +--ro policy-color-ref?
         |       leafref
         +--ro policy-endpoint-ref?
         |       leafref
         +--ro existing-candidate-path-protocol-origin-ref?
         |       leafref
         +--ro existing-candidate-path-preference-ref?
         |       leafref
         +--ro (dataplane-type)?
         |  +--:(mpls)
         |  |  +--ro existing-candidate-path-mpls-binding-sid-value-ref?
         |  |          leafref {capability-candidate-path-binding-sid}?
         |  +--:(srv6)
         |     +--ro existing-candidate-path-srv6-binding-sid-value-ref?
         |             leafref {capability-candidate-path-binding-sid}?
         +--ro conflicting-candidate-path-protocol-origin?
         |       uint8
         +--ro conflicting-candidate-path-preference?
         |       uint32
         +--ro conflicting-candidate-path-binding-sid-dataplane?
         |       sr-policy-types:dataplane-type
         +--ro conflicting-candidate-path-binding-sid-value?
                 sr-policy-types:sid-value-type
```

Figure 2: SR Policy - YANG Tree

7. YANG Specification

Following are actual YANG definition for the modules defined earlier in the document.

7.1. Types

This module references [RFC9256].

```
<CODE BEGINS> file "ietf-sr-policy-types@2026-05-23.yang"
module ietf-sr-policy-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy-types";

  prefix "sr-policy-types";

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

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

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

  organization "IETF SPRING Working Group";

  contact
      "WG Web:   <http://tools.ietf.org/wg/spring/>
       WG List:  <mailto:spring@ietf.org>
       Editor:   Kamran Raza
       <mailto:skraza@cisco.com>
       Editor:   Tarik Saleh
       <mailto:tasaleh@cisco.com>
       Editor:   Zhuang Shunwan
       <mailto:zhuangshunwan@huawei.com>
       Editor:   Satoru Matsushima
       <mailto:satoru.matsushima@g.softbank.co.jp>
       Editor:   Pavan Vishnu Beeram
       <mailto:vbeeram@juniper.net>
       ";

  description
    "This YANG module defines the essential types for the management
     of SR policy module.
```

```
      Copyright (c) 2024 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 Revised BSD License set
      forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

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

      // RFC Editor: replace XXXX with actual RFC number and remove
      // this note

    revision "2026-05-23" {
      description
        "Remove the metric type definition. The metric type from
         te-types will be reused instead.";
      reference "RFC 9256";
    }

    revision "2025-05-22" {
      // RFC Editor: replace the above date 2025-05-22 with the date of
      // publication and remove this note.

      description
        "Add a type for MPLS SID value";
      reference
        "RFC XXXX: YANG Data Model for Segment-routing Policy Types";
       // RFC Editor: replace XXXX with actual RFC number and remove
       // this note
    }

    revision "2024-03-04" {
      // RFC Editor: replace the above date 2024-03-04 with the date of
      // publication and remove this note.

      description
        "Alignment with SR policy architecture RFC9256";
      reference
        "RFC XXXX: YANG Data Model for Segment-routing Policy Types";
       // RFC Editor: replace XXXX with actual RFC number and remove
       // this note
    }

    revision "2022-03-28" {
```

```
      description
        "Alignment with SR policy architecture";
      reference
        "draft-ietf-spring-sr-policy-yang";
    }

    revision "2019-11-04" {
      description
        "New editor added";
      reference
        "draft-raza-spring-sr-policy-yang";
    }

    revision "2019-07-08" {
      description
        "Dynamic TE candidate-path support";
      reference
        "draft-raza-spring-sr-policy-yang";
    }

    revision "2018-07-01" {
      description
        "Initial version";
      reference
        "draft-raza-spring-sr-policy-yang";
    }

    /* Identities */
    identity candidate-path-not-selected-reason {
      description
        "Base identity for which reasons for not selecting
        candidate path are derived from";
    }
    identity candidate-path-not-selected-not-best {
      base candidate-path-not-selected-reason;
      description
        "Higher preference path exists";
    }
    identity candidate-path-not-selected-no-valid-segment-list {
      base candidate-path-not-selected-reason;
      description
        "Candidate path has no valid segment list(s)";
    }
    identity candidate-path-not-selected-empty-segment-list {
      base candidate-path-not-selected-reason;
      description
        "Candidate path has empty segment list(s)";
    }
```

```
identity candidate-path-not-selected-invalid-binding-sid {
  base candidate-path-not-selected-reason;
  description
    "Candidate path has invalid binding SID";
}

identity policy-down-reason {
 description
    "Base identity for the reasons why SR policy is operationally
    down";
}
identity policy-down-reason-admin-down {
  base policy-down-reason;
  description "Policy is administrately down";
}
identity policy-down-reason-no-source-address {
  base policy-down-reason;
  description "Policy has no source address";
}
identity policy-down-reason-no-endpoint {
  base policy-down-reason;
  description "Policy has no end-point";
}
identity policy-down-reason-no-candidate-path {
  base policy-down-reason;
  description "Policy has no candidate path";
}
identity policy-down-reason-no-valid-candidate-path {
  base policy-down-reason;
  description "Policy has no valid candidate path";
}
identity policy-down-reason-candidate-path-invalid-segment-list {
  base policy-down-reason;
  description "Policy's candidate path has invalid segment list";
}
identity policy-down-reason-policy-unconfigured {
  base policy-down-reason;
  description "Policy is unconfigured";
}
identity policy-down-reason-policy-color-endpoint-updated {
  base policy-down-reason;
  description "Policy's color and end-point are updated";
}
identity policy-down-reason-local-label-setup-failed {
  base policy-down-reason;
  description "Policy's local label setup (allocation/rewrite)
  failed";
}
```

```
    identity policy-down-reason-forwarding-rewrite-failed {
      base policy-down-reason;
      description "Policy's forwarding rewrite installation failed";
    }
    identity policy-down-reason-internal-error {
      base policy-down-reason;
      description "Infra related internal error";
    }

    identity binding-sid-unavailable-reason {
      description
        "Base identity for binding sid unavailable reason types";
    }
    identity binding-sid-allocation-error {
      base binding-sid-unavailable-reason;
      description "SID allocator returned an error";
    }
    identity binding-sid-already-exists {
      base binding-sid-unavailable-reason;
      description "Binding sid already exists/allocated";
    }
    identity binding-sid-internal-error {
      base binding-sid-unavailable-reason;
      description "Internal error with binding sid allocation";
    }
    identity binding-sid-color-endpoint-conflict {
      base binding-sid-unavailable-reason;
      description "Binding sid already allocated by another sr-policy
      with different color/endpoint";
    }
    identity binding-sid-rewrite-error {
      base binding-sid-unavailable-reason;
      description "Binding sid forwarding rewrite error";
    }
    identity binding-sid-outside-srlb-range {
      base binding-sid-unavailable-reason;
      description "Binding sid outside SRLB range";
    }

    identity segment-type {
      description
        "Base identity for the segment type";
    }
    identity segment-type-A {
      base segment-type;
      description "SR-MPLS Label";
    }
    identity segment-type-B {
```

```
      base segment-type;
      description "SRv6 SID";
    }
    identity segment-type-C {
      base segment-type;
      description "IPv4 Prefix with optional SR Algorithm";
    }
    identity segment-type-D {
      base segment-type;
      description "IPv6 Global Prefix with optional SR Algorithm for
      SR-MPLS";
    }
    identity segment-type-E {
      base segment-type;
      description "IPv4 Prefix with Local Interface ID";
    }
    identity segment-type-F {
      base segment-type;
      description "IPv4 Addresses for link endpoints as Local,
      Remote pair";
    }
    identity segment-type-G {
      base segment-type;
      description
        "IPv6 Prefix and Interface ID for link endpoints as Local,
        Remote pair for SR-MPLS";
    }
    identity segment-type-H {
      base segment-type;
      description
        "IPv6 Addresses for link endpoints as Local,
        Remote pair for SR-MPLS";
    }
    identity segment-type-I {
      base segment-type;
      description
        "IPv6 Global Prefix with optional SR Algorithm for SRv6";
    }
    identity segment-type-J {
      base segment-type;
      description
        "IPv6 Prefix and Interface ID for link endpoints as Local,
        Remote pair for SRv6";
    }
    identity segment-type-K {
      base segment-type;
      description
        "IPv6 Addresses for link endpoints as Local, Remote pair for
```

```
        SRv6";
    }

    identity protocol-origin-type {
      description "Base identity for originating protocol type";
    }
    identity protocol-origin-type-pcep {
      base protocol-origin-type;
      description
        "PCEP used as signalling mechanism for the candidate path";
    }
    identity protocol-origin-type-bgp {
      base protocol-origin-type;
      description
        "BGP used as signalling mechanism for the candidate path";
    }
    identity protocol-origin-type-local {
      base protocol-origin-type;
      description
        "CLI, Yang model via Netconf, gRPC, etc used for candidate path
        instantiation";
    }

    /* Typdefs */
    typedef mpls-sid-value-type {
        type rt-types:mpls-label;
      description "MPLS SID value type";
    }

    typedef sid-value-type {
      type union {
        type rt-types:mpls-label;
        type srv6-types:srv6-sid;
      }
      description "The SID value type";
    }

    typedef binding-sid-oper-state {
      type enumeration {
        enum ALLOC-PENDING {
          value 1;
          description "SID allocation pending for Binding SID";
        }
        enum PROGRAMMED {
          value 3;
          description "Binding SID is programmed in forwarding";
        }
        enum CONFLICT {
```

```
          value 4;
          description "Binding SID is in-conflict state with
          regards to SID allocation. This also means that SID
          allocation is pending";
        }
      }
      description
        "Binding SID operational state type";
    }

    typedef policy-admin-state {
      type enumeration {
        enum UP {
          value 1;
          description "SR policy is administratively up";
        }
        enum DOWN {
          value 2;
          description "SR policy is administratively down";
        }
      }
      description "SR policy admin state";
    }

    typedef policy-oper-state {
      type enumeration {
        enum UP {
          value 1;
          description "SR policy is operationally up";
        }
        enum DOWN {
          value 2;
          description "SR policy is operationally down";
        }
      }
      description "SR policy oper state";
    }

    typedef margin-type {
      type enumeration {
        enum absolute { value 1; description "Absolute value"; }
        enum relative { value 2; description "Relative value"; }
      }
      description "Margin type";
    }

    typedef dataplane-type {
      type enumeration {
```

```
        enum mpls {
          value 1;
          description "Segment-routing MPLS";
        }
        enum srv6 {
          value 2;
          description "Segment-routing v6";
        }
      }
      description "Dataplane type of the segments";
    }

    typedef binding-sid-alloc-mode {
      type enumeration {
        enum explicit {
          value 1;
          description "Explicitly specified BSID";
        }
        enum dynamic {
          value 2;
          description "Dynamically allocated BSID";
        }
      }
      description "Binding SID allocation mode";
    }

    typedef explicit-binding-sid-rule-type {
      type enumeration {
        enum enforce-srlb {
          value 1;
          description
              "Explicit Binding SID is enforced with no
               fallback if label does not fall in SRLB or
               if no SRLB is configured";
        }
        enum fallback-dynamic {
          value 2;
          description
              "Explicit Binding SID falls back to dynamic in
              case explicit label is not available.";
        }
      }
      description "Explicit binding sid rule types";
    }

    typedef sid-protection-eligibility-type {
      type enumeration {
        enum protected-preferred {
```

```
            value 1;
            description "Prefer protected SID";
          }
          enum protected-only {
            value 2;
            description "Protected SID only";
          }
          enum unprotected-preferred {
            value 3;
            description "Prefer unprotected SID";
          }
          enum unprotected-only {
            value 4;
            description "Unprotected SID only";
          }
        }
        description "Types for SID protection eligibility";
      }

      typedef node-address-type {
        type inet:ipv6-address;
        description "Originator node address";
      }

      typedef sid-algorithm-type {
          type uint8 {
            range "0..255";
          }
        description "SID algorithm type";
      }

      typedef color-type {
        type uint32 {
          range "1..4294967295";
        }
        description "Policy color type";
      }

      typedef name-type {
        type string {
          length "1..59";
        }
        description "Policy name type";
      }
    }
    <CODE ENDS>
```

Figure 3: ietf-sr-policy-types.yang

7.2. SR Policy

This module references [RFC6780] and [RFC9256].

```
<CODE BEGINS> file "ietf-sr-policy@2026-05-23.yang"
module ietf-sr-policy {

  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-policy";

  prefix "sr-policy";

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

  import ietf-interfaces {
    prefix if;
  }

  import ietf-routing {
    prefix "rt";
  }

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

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

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

  import ietf-sr-policy-types {
    prefix "sr-policy-types";
  }

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

  organization "IETF SPRING Working Group";
  contact
```

```
    "WG Web:   <http://tools.ietf.org/wg/spring/>
     WG List:  <mailto:spring@ietf.org>
     Editor:   Kamran Raza
     <mailto:skraza@cisco.com>
     Editor:   Tarik Saleh
     <mailto:tasaleh@cisco.com>
     Editor:   Zhuang Shunwan
     <mailto:zhuangshunwan@huawei.com>
     Editor:   Satoru Matsushima
     <mailto:satoru.matsushima@g.softbank.co.jp>
     Editor:   Vishnu Pavan Beeram
     <mailto:vbeeram@juniper.net>
    ";

  description
    "This module contains a collection of YANG definitions
    for SR policy module.

     Copyright (c) 2024 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 Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

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

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  revision "2026-05-23" {
    description
      "Use the path metric type from te-types.";
    reference "RFC 9256";
  }

  revision "2025-10-20" {
    description
      "Add the headend-address to the policy key to ensure global
       uniqueness across multiple headends, consistent with RFC 9256.";
```

```
    reference "RFC 9256";
  }

  revision "2025-05-22" {
    description
      "Update the SR policy YANG model with more alignment with SR
       policy architecture";
    reference
      "RFC XXXX: YANG Data Model for Segment-routing Policy";
     // RFC Editor: replace XXXX with actual RFC number and remove
     // this note
  }

  revision "2024-11-22" {
    description
      "Fix SR policy YANG model";
    reference
      "RFC XXXX: YANG Data Model for Segment-routing Policy";
     // RFC Editor: replace XXXX with actual RFC number and remove
     // this note
  }

  revision "2024-03-04" {
    description
      "Alignment with SR policy architecture RFC9256";
    reference
      "RFC XXXX: YANG Data Model for Segment-routing Policy";
     // RFC Editor: replace XXXX with actual RFC number and remove
     // this note
  }

  revision "2022-03-28" {
    description
      "Alignment with SR policy architecture";
    reference
      "draft-ietf-spring-sr-policy-yang";
  }

  revision "2019-11-04" {
    description
      "Changes in keys for policy and its candidate paths";
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  revision "2019-07-08" {
    description
      "Dynamic TE candidate-path support";
```

```
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  revision "2018-07-01" {
    description
      "Initial version";
    reference
      "draft-raza-spring-sr-policy-yang";
  }

  //TBD: To be moved to srv6-base model when the model is ready
  grouping srv6-sid-info {
    description "Define an SRv6 SID";

    leaf value {
      type srv6-types:srv6-sid;
      description "SRv6 SID value";
    }

    leaf sid-behavior {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "SRv6 BSID behavior";
    }
    container sid-structure {
      description "SRv6 BSID structure";
      uses srv6-types:srv6-sid-structure-grouping;
    }
  }

  grouping srv6-bsid-type-b-config {
    description "SRv6 BSID Type-B config grouping";

    leaf sid-behavior {
      type identityref {
        base srv6-types:srv6-endpoint-type;
      }
      description "SRv6 BSID behavior";
    }

    container sid-structure {
      description "SRv6 BSID structure";
      uses srv6-types:srv6-sid-structure-grouping;
    }
  }
```

```
  grouping segment_config {
    description "Segment properties grouping";
    leaf index {
      type uint32;
      description "Segment index";
    }
    leaf type {
      type identityref {
        base sr-policy-types:segment-type;
      }
      mandatory true;
      description "Segment type";
    }

    container sr-mpls {
      description "SR-MPLS specific types";

      container Type-A {
        description
          "Segment declared by MPLS label";
        leaf value {
          type rt-types:mpls-label;
          description "MPLS label value";
        }
      }

      container Type-C {
        description
          "Segment declared by IPv4 Prefix with optional SR Algorithm
          for SR-MPLS";
        leaf ipv4-address {
          type inet:ipv4-address;
          description "Segment IPv4 address";
        }
        leaf algorithm {
          type uint8;
          description "Prefix SID algorithm identifier";
        }
      }

      container Type-D {
        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 Type-E {
        description
          "Segment declared by IPv4 Prefix with Local Interface ID -
          SR-MPLS";
        leaf ipv4-address {
          type inet:ipv4-address;
          description "Node IPv4 address";
        }
        leaf interface-identifier {
          type uint32;
          description "local interface identifier";
        }
      }

      container Type-F {
        description
          "Segment declared by IPv4 Addresses for link endpoints
          as Local, Remote pair - SR-MPLS";
        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 Type-G {
        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 Type-H {
        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";
        }
      }
    } // sr-mpls

    container srv6 {
      description "SRv6 specific types";

      container Type-B {
        description
          "Segment declared by SRv6 SID value";
        leaf value {
          type srv6-types:srv6-sid;
          description "SRv6 SID value";
        }

        uses srv6-bsid-type-b-config;
      }

      container Type-I {
        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 Type-J {
        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 Type-K {
        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";
        }
      }
    } // srv6

    leaf validate {
      type boolean;
      default 'false';
      description
        "Indicates whether the segment should be validated.
        The default applies to all segments other than the first
        segment. For the first segment, validation is always done.";
    }
  }
```

```
  grouping segment-properties {
    description
      "SR segment properties grouping";
    uses segment_config;
  }

  grouping optimization-template-constraints {
    description "Constraints of optimization template";

      container srlg { // the rule applies per link
        description "SRLG constraints on the computed dynamic path";

        leaf-list exclude-any {
          type string;
          description
            "The link is excluded if it has any of these SRLG.";
        }
      }

      container affinity {  // the rule applies per link
        description
          "Affinity constraints on the computed dynamic path";

        leaf-list exclude-any {
          type string;
          description
            "The link is excluded if it has any of these affinities."
            ;
        }

        leaf-list include-any {
          type string;
          description
            "The link is accepted if it has any of these affinities";
        }
        leaf-list include-all {
          type string;
          description
            "The link is accepted if it has all these affinities";
        }
      }

      container ip-address {
        // the rule applies to e2e path and not per link
        description
          "IP address constraints on the computed dynamic path";

        leaf-list exclude {
```

```
          type inet:ip-address;
          description
            "Must exclude the links that has any of these IP
            addresses.";
        }
      }

      container upper-bounds {
        description
          "Upper-bound constraints on the computed dynamic path";

        container cumulative-metric {
          description
            "Path is invalid if its metric exceeds this value";

          leaf metric-type {
            type identityref {
              base te-types:path-metric-type;
            }
            description "Metric type";
          }

          leaf value { type uint32; description "Cumulative value"; }
        }

        leaf max-sids {
          type uint32;
          description
            "Path is invalid if it has more SIDs than this value";
        }
      }

      container segment-rules {
        description
          "Constraints on the segments to be used in the path";

        leaf sid-protection-eligibility {
          // currently applies to adj-sid segments only
          type sr-policy-types:sid-protection-eligibility-type;
          description "SID protection eligibility";
        }
      }

      leaf max-sids {
        type uint32;
        description "Maximum number of SIDs in a segment-list";
      }
    }
  }
```

```
  grouping optimization-templates {
    description
      "Path optimization templates properties";

    container optimization-templates {
      description "Path optimization templates";

      list optimization-template {
        key "color";
        description "Path optimization template";

        leaf color {
          type uint32;
          description "Template identifier as a policy color";
        }

        container candidate-paths {
          description "A list of candidate paths representing the
                       optimization objectives and constraints";

          list candidate-path {
            key "discriminator";
            description "Candidate paths representing the
                         optimization objectives and constraints";
            leaf discriminator {
              type uint32;
              description "Candidate path distinguisher";
            }
            uses cpath-optimization-objs;
            container constraints {
              description "Constraints of a path optimization template";

              uses optimization-template-constraints;
            }
          }
        }
      }
    }
  }

  grouping attributes {
    description
      "Grouping containing attributes applicable to all SR policies";

    container attributes {
      description
        "Attributes applicable to SR policies";
```

```
      uses segment-lists;
      uses explicit-binding-sid-rules;
    }
  }

  grouping forwarding-counters {
    description
      "Grouping for counters";
    container counters {
      config false;
      description
        "Counters containing stats related to forwarding";

      leaf pkts {
        type yang:counter64;
        description "Number of packets forwarded";
      }
      leaf octets {
        type yang:counter64;
        units "byte";
        description "Number of bytes forwarded";
      }
    }
  }
  grouping segment-lists {
    description
      "Segment lists grouping";
    container segment-lists {
      description "Segment-lists properties";

      list segment-list {
        key "name";
        description "Segment-list properties";
        leaf name {
          type string;
          description "Segment-list name";
        }
        container segments {
          description
            "Segments for given segment list";

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

```
    }
  }

  grouping binding-sid_config {
    description
      "Binding SID configuration properties grouping";
    container mpls-sid {
      description "MPLS binding SIDs";
      leaf value {
        type sr-policy-types:mpls-sid-value-type;
        description "MPLS binding SID value";
      }
    }

    container srv6-sids {
      description "SRv6 binding SIDs";

      list srv6-sid {
        key "index";
        description "Configure an SRv6 binding-sid";
        leaf index {
          type uint32;
          description "Binding SID index";
        }
        leaf locator-name {
          type srv6-types:srv6-locator-name-type;
          description "Locator name used to allocate SRv6 binding SID";
        }
        uses srv6-sid-info;
      }
    }
  }

  grouping binding-sid_state {
    description
      "Binding SID state properties grouping";
    leaf alloc-mode {
      type sr-policy-types:binding-sid-alloc-mode;
      config false;
      description "Binding SID allocation mode";
    }
    leaf allocated-sid {
      type sr-policy-types:sid-value-type;
      config false;
      description "Allocated SID value for the Binding SID";
    }
    leaf oper-state {
      type sr-policy-types:binding-sid-oper-state;
```

```
      config false;
      description
        "Binding SID operational state";
    }
  }

  grouping binding-sid-properties {
    description
      "Binding SID properties grouping";
    container binding-sid {
      description "Binding Segment ID";
      uses binding-sid_config;
    }
  }

  grouping path-forwarding_state {
    description "Policy Forwarding path information";
    leaf path-id {
      type uint8;
      description "Primary path id";
    }
    leaf next-hop-address {
      type inet:ip-address;
      description "Nexthop address";
    }
    leaf next-hop-table-id {
      type uint32;
      description "Table ID for nexthop address";
    }
    leaf interface {
      type if:interface-ref;
      description "Outgoing interface handle";
    }
    leaf is-protected {
      type boolean;
      description "Is this path protected ?";
    }
    leaf backup-path-id {
      type uint8;
      description "Backup path id";
    }
    leaf is-pure-backup {
      type boolean;
      description "Is this path a pure backup ?";
    }
    leaf weight {
      type uint32;
      description "Path's weight for W-ECMP balancing";
```

```
    }
    container sid-list {
      description "Outgoing sid stack";

      uses sid-list-grouping;
    }
  }

  grouping cpath-cmn-properties {
    description
      "Common properties of the candidate path";

    leaf is-active {
      type boolean;
      config false;
      description
        "True if the candidate path is valid and used to forward
        traffic, False otherwise";
    }
    leaf is-valid {
      type boolean;
      config false;
      description
        "True if at least one segment-list is valid,
        False otherwise";
    }

    container computed-sid-list {
      config false;
      description "Computed SID list which could be different from
                   outgoing SID list";

      uses sid-list-grouping;
    }

    container forwarding-paths {
      config false;
      description
        "Forwarding state of paths";
      list forwarding-path {
        key "path-id";
        description "Forwarding path";
        uses path-forwarding_state;
        uses forwarding-counters;
      }
    }
  }
```

```
  grouping explicit-path-properties {
    description
      "Explicit path properties of the candidate path";
    container segment-lists {
      description
        "Path segment list(s) properties";
      list segment-list {
        key "name-ref";
        description "SR policy candidate path segment lists";

        leaf name-ref {
          type leafref {
            path "/rt:routing/"
              + "sr-policy:segment-routing/"
              + "sr-policy:traffic-engineering/"
              + "sr-policy:attributes/"
              + "sr-policy:segment-lists/"
              + "sr-policy:segment-list/"
              + "sr-policy:name";
          }
          description "Reference to segment-list name";
        }
        leaf weight {
          type uint32;
          default 1;
          description "Segment-list weighted loadshare";
        }
        leaf is-valid {
          type boolean;
          config false;
          description
            "True if the segment-list is valid, False otherwise";
        }
        uses forwarding-counters;
      }
    }
  }

  grouping affinity-mapping {
    description "Affinity-map grouping";

    container affinity-map {
      description
        "Mapping of affinity names to bit position";
      list affinity {
        key "name";
        unique "bit-position";
        leaf name {
```

```
          type string;
          description
            "Name of the affinity";
        }
        leaf bit-position {
          type uint16;
          description
            "The affinity entry in this list is mapped to the this
            bit-position in the affinity bitmap";
        }

        description "Affinity";
      }
    }
  }

  grouping srlg {
    description "SRLG grouping";

    container srlgs {
      description "SRLG";
      list srlg {
        key "name";

        leaf name {
          type string;
          description
            "Name of the SRLG";
        }

        leaf description {
          type string;
          description
            "Description of the SRLG";
        }
        leaf value {
          type uint32;
          mandatory true;
          description
            "The associated value (for advertisements etc.) ";
        }

        description "SRLG";
      }
    }
  }

  grouping cpath-optimization-objs {
```

```
    description "Candidate-path optimization objective grouping";

    container optimization-objectives {
      description
        "Optimization objectives for the dynamic path computation";
      container minimize-metric {
        description "The metric for which the path is optimized";
        leaf metric-type {
            type identityref {
              base te-types:path-metric-type;
            }
          description "Metric type";
        }

        container margin {
          description
            "The margin by which two paths with similar metric-type
            would be considered equal";
          leaf type {
            type sr-policy-types:margin-type;
            description "Margin type";
          }
          leaf value {
            type uint32;
            description "Margin value";
          }
          leaf computed-value {
            type uint32;
            config false;
            description "Computed metric value";
          }
        }
      }
    }
  }

  grouping cpath-constraints {
    description "Candidate-path constraints grouping";

    container constraints {
      description "Constraints for the dynamic path computation";

      container srlg { // the rule applies per link
        description "SRLG constraints on the computed dynamic path";

        leaf-list exclude-any {
          type string;
          description
```

```
            "The link is excluded if it has any of these SRLG.";
        }
        leaf-list include-any {
          type string;
          description
            "The link is accepted if it has any of these SRLG";
        }

        leaf-list include-all {
          type string;
          description
            "The link is accepted if it has all these SRLG";
        }
      }

      container affinity {  // the rule applies per link
        description
          "Affinity constraints on the computed dynamic path";

        leaf-list exclude-any {
          type string;
          description
            "The link is excluded if it has any of these affinities."
            ;
        }

        leaf-list include-any {
          type string;
          description
            "The link is accepted if it has any of these affinities";
        }
        leaf-list include-all {
          type string;
          description
            "The link is accepted if it has all these affinities";
        }
      }

      container ip-address {
        // the rule applies to e2e path and not per link
        description
          "IP address constraints on the computed dynamic path";

        leaf-list exclude {
          type inet:ip-address;
          description
            "Must exclude the links that has any of these IP
            addresses.";
```

```
        }
        leaf-list include-any {
          type inet:ip-address;
          description
            "Must include atleast 1 link (or possibly more) with any
            of these IP addresses.";
        }
        leaf-list include-all {
          type inet:ip-address;
          description
            "Must include all the links that have any of these IP
            addresses.";
        }
      }

      container upper-bounds {
        description
          "Upper-bound constraints on the computed dynamic path";

        container cumulative-metric {
          description
            "Path is invalid if its metric exceeds this value";

          leaf metric-type {
            type identityref {
              base te-types:path-metric-type;
            }
            description "Metric type";
          }

          leaf value { type uint32; description "Cumulative value"; }
        }

        leaf max-sids {
          type uint32;
          description
            "Path is invalid if it has more SIDs than this value";
        }
      }

      container segment-rules {
        description
          "Constraints on the segments to be used in the path";
        leaf sid-algorithm {
          type sr-policy-types:sid-algorithm-type;
          description
            "The prefix-sid algorithm to be used in
```

```
            path calculation";
        }

        leaf sid-protection-eligibility {
          // currently applies to adj-sid segments only
          type sr-policy-types:sid-protection-eligibility-type;
          description "SID protection eligibility";
        }
      }

      container disjoint-path {
        description "Path disjointness constraints";

        leaf disjointness-type {
          type te-types:te-path-disjointness;
          description
            "Type of disjointness computation used to find the path";
        }

        leaf association-group-id {
          type uint16;
          mandatory true;
          description "Association group Id";
        }

        leaf association-source {
          type inet:ip-address;
          mandatory true;
          description "Association source address";
        }

        leaf association-extended-id {
          type yang:hex-string;
          description
            "Association extended identifier.";
          reference
            "RFC6780";
        }
      }

      leaf max-sids {
        type uint32;
        description "Maximum number of SIDs in a segment-list";
      }
    }
  }

  grouping dynamic-path-properties {
```

```
    description
      "Dynamic path properties of the candidate path";
    container segment-list {
      description
        "Segment-list properties";
      leaf segment-list-dataplane-type {
        type sr-policy-types:dataplane-type;
        mandatory true;
        description
          "The dataplane type for the segment-list";
      }
      leaf is-valid {
        type boolean;
        config false;
        description
          "True if the segment-list is valid, False otherwise";
      }
      uses forwarding-counters;
    }

    uses cpath-optimization-objs;
    uses cpath-constraints;
  }
  grouping composite-path-properties {
    description
      "Composite path properties of the candidate path";

    container constituent-policies {
      description "Constituent SR policies";

      list constituent-policy {
        key "color";
        description "List of constituent SR policies";

        leaf color {
          type sr-policy-types:color-type;
          description
            "Color associated with the constituent policy
            (the endpoint is same as the parent policy)";
        }
        leaf weight {
          type uint32;
          default 1;
          description "Constituent policy weighted loadshare";
        }
      }
    }
  }
```

```
  grouping mpls-label-stack {
    description
      "Grouping for MPLS label stack";

    list labels {
      key "index";
      description
        "Stack containing MPLS labels";

      leaf index {
        type uint32;
        description "A unique ID of an MPLS label in labels list";
      }
      leaf label {
        type rt-types:mpls-label;
        description
          "MPLS label value";
      }
    }
  }

  grouping srv6-sid-stack {
    description
      "Grouping for SRv6 label stack";

    list sids {
      key "index";
      description
        "Stack containing SRv6 SIDs";

      leaf index {
        type uint32;
        description "A unique ID of an SRv6 sid in sid list";
      }
      leaf sid {
        type srv6-types:srv6-sid;
        description
          "SRv6 sid value";
      }
    }
  }

  grouping sid-list-grouping {
    description "Grouping for SID list";

    choice dataplane-type {
      description
        "Outgoing sids dataplane choice";
```

```
      case mpls {
        uses mpls-label-stack;
      }
      case srv6 {
        uses srv6-sid-stack;
      }
    }
  }

  grouping candidate-path_state {
    description
      "Candidate path state properties grouping";
    leaf is-best-candidate-path {
      type boolean;
      default 'false';
      config false;
      description
        "True if the candidate path is the best candidate path,
        False otherwise";
    }
    leaf non-selection-reason {
      type identityref {
        base sr-policy-types:candidate-path-not-selected-reason;
      }
      config false;
      description
        "Candidate path not selected reason";
    }
  }

  grouping policy-properties_config {
    description
      "SR policy configuration grouping";
    leaf name {
      type sr-policy-types:name-type;
      description "SR policy name";
    }
    leaf headend {
      type inet:ip-address;
      description
        "The IP address identifying the SR Policy headend node.
         This addition ensures global uniqueness of the policy key
         across multiple headends";
      reference
        "RFC 9256: Section 2.1 - Identification of an SR Policy.";
    }
    leaf color {
      type sr-policy-types:color-type;
```

```
      description "Color associated with the policy";
    }
    leaf endpoint {
      type inet:ip-address;
      description "Policy end point IP address";
    }
    leaf description {
      type string;
      description "Description of the policy";
    }
    leaf admin-state {
      type sr-policy-types:policy-admin-state;
      default 'UP';
      description
        "SR policy administrative state, true for
        enabled, false for disabled";
    }
    leaf priority {
      type uint8;
      default 128;
      description
        "Priority considered when policy is recomputed due to
        topology changes";
    }
    leaf profile-id {
      type uint32;
      description "A profile represents a set of configuration knobs
                   specifying a policy or candidate-path attributes.
                   The profile-ID feature allows usage of
                   vendor/implementation specific functionality per
                   policy without requiring  explicit support by
                   controller.";
    }
    container bandwidth {
      description "Bandwidth constraints";

      leaf requested {
        type uint64;
        description
          "Requested bandwidth expressed in bytes per second";
      }
      leaf reserved {
        type uint64;
        config false;
        description
          "Reserved bandwidth expressed in bytes per second";
      }
    }
```

```
  }

  grouping policy-properties_state {
    description
      "SR policy property grouping";
    leaf oper-state {
      type sr-policy-types:policy-oper-state;
      config false;
      description
        "SR policy operational state";
    }
    leaf transition-count {
      type uint32;
      config false;
      description "Indicates number of up/down transitions";
    }
    leaf up-time {
      type yang:date-and-time;
      config false;
      description "Policy up time in seconds";
    }
    leaf down-time {
      type yang:date-and-time;
      config false;
      description "Policy down time in seconds";
    }
  }
  grouping policy-properties {
    description
      "SR policy properties";
    uses policy-properties_config;
    uses policy-properties_state;
    uses binding-sid-properties;
    uses forwarding-counters;
  }

  grouping candidate-path-type {
    description "Candidate path type grouping";
    choice type {
      description
        "Type of candidate paths";
      case explicit {
        description
          "Candidate path with explicitly defined set/s of
          segment-lists";
        uses explicit-path-properties;
      }
      case dynamic {
```

```
        description
          "Candidate path with dynamic computed segment-lists";
        uses dynamic-path-properties;
      }
      case composite {
        description
          "Candidate path that groups SR policies each with explicit
          and/or dynamic candidate path with potentially different
          optimization objectives and constraints";
        uses composite-path-properties;
      }

    }
  }

  grouping candidate-paths {
    description "SR policy candidate path grouping";
    container candidate-paths {
      description "SR policy candidate path(s) ";

      list candidate-path {
        key "protocol-origin originator-asn originator-node-address
          discriminator";
        unique "preference";

        description "SR policy Candidate path(s) list entry";

        leaf protocol-origin {
          type identityref {
            base sr-policy-types:protocol-origin-type;
          }
          description
            "Instantiation mechanism used to create the candidate
            path";
        }
        leaf originator-asn {
          type uint32;
          description
            "The autonomous system number (ASN) of the provisioning
            node. If 2-byte ASNs are in use, the low-order 16 bits
            MUST be used, and the high-order bits MUST be set to 0";
        }
        leaf originator-node-address {
          type sr-policy-types:node-address-type;
          description
            "represented as a 128-bit value. IPv4 addresses MUST be
            encoded in the lowest 32 bits, and the high-order bits
            MUST be set to 0";
```

```
        }
        leaf discriminator {
          type uint32;
          description "Candidate path distinguisher";
        }
        leaf preference {
          type uint32 {
            range "1..65535";
          }
          mandatory true;
          description "Candidate path preference";
        }
        leaf name {
          type string;
          description "Candidate path name";
        }
        leaf description {
          type string;
          description "Candidate path description";
        }
        leaf drop-upon-invalid {
          type boolean;
          description "Enable traffic dropping when LSP is invalid";
        }
        leaf admin-state {
          type boolean;
          description "Shutdown the candidate-path";
        }
        container binding-sid {
          if-feature capability-candidate-path-binding-sid;
          description
            "Binding segment ID";
          uses binding-sid_config;
        }

        uses candidate-path-type;
        uses candidate-path_state;
        uses cpath-cmn-properties;
      }
    }
  }

  grouping policies {
    description "SR policy grouping";
    container policies {
      description "SR Policy container";

      list policy {
```

```
        key "headend color endpoint";
        unique "name";
        description
          "List of SR Policies. Each SR Policy is uniquely identified
           by the tuple <Headend, Color, Endpoint> globally,
           as defined in RFC 9256. The headend identifies the ingress
           node that instantiates or owns this policy.";

        uses policy-properties;
        uses candidate-paths;
      }
    }
  }

  grouping explicit-binding-sid-rules {
    description
      "Grouping for explicit binding sid rules";

    list explicit-binding-sid-rules {
      key "index";
      description
        "Explicit binding sid rules applicable for all policies";
      leaf index {
        type uint32;
        description "Explicit binding SID rules list index";
      }
      leaf rule {
        type sr-policy-types:explicit-binding-sid-rule-type;
        mandatory true;
        description "Explicit binding sid rule";
      }
    }
  }

  augment "/rt:routing" {
    description
      "This augments routing-instance configuration with
      segment-routing sr-policy.";
    container segment-routing {
      description "Main segment routing container";
      container traffic-engineering {
        description "Traffic-engineering container";

        uses optimization-templates;
        uses attributes;
        uses policies;
      }
    }
```

```
  }

  augment "/rt:routing/"
    + "sr-policy:segment-routing/"
    + "sr-policy:traffic-engineering/"
    + "sr-policy:policies/"
    + "sr-policy:policy/"
    + "sr-policy:binding-sid/"
    + "sr-policy:mpls-sid" {
    description
      "Augments binding-sid state with MPLS binding-sid";
    uses binding-sid_state;
  }

  augment "/rt:routing/"
    + "sr-policy:segment-routing/"
    + "sr-policy:traffic-engineering/"
    + "sr-policy:policies/"
    + "sr-policy:policy/"
    + "sr-policy:binding-sid/"
    + "sr-policy:srv6-sids/"
    + "sr-policy:srv6-sid" {
    description
      "Augments binding-sid state with SRv6 binding-sid";
    uses binding-sid_state;
  }

  /* Notifications */

  notification sr-policy-oper-state-change-event {
    description
      "Notification event when the operational state of the SR policy
      changes";

    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
```

```
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf policy-new-oper-state  {
      type sr-policy-types:policy-oper-state;
      description "New operational state of the SR policy";
    }

    leaf policy-down-reason {
      type identityref {
        base sr-policy-types:policy-down-reason;
      }
      description
        "Down reason if the SR policy's new operational state is
        down";
    }
  }

  notification sr-policy-candidate-path-change-event {
    description
      "Notification event when candidate path changes for SR policy";

    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
```

```
          + "sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-preference {
      type uint32;
      description "Existing candidate path preference";
    }

    leaf new-preference {
      type uint32;
      description "New candidate path preference";
    }
  }

  notification sr-policy-binding-sid-unavailable-event {
    description
      "Notification event when the binding sid of sr-policy is
      unavailable";

    leaf policy-name-ref {
      type leafref {
        path "/rt:routing/"
```

```
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:name";
      }
      description "Reference to sr-policy name";
    }

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:color";
      }
      description "Reference to sr-policy color";
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    choice dataplane-type {
      description
        "Binding SID dataplane choice";
      case mpls {
        leaf policy-mpls-binding-sid-value-ref {
          if-feature capability-candidate-path-binding-sid;
          type leafref {
            path "/rt:routing/"
              + "sr-policy:segment-routing/"
              + "sr-policy:traffic-engineering/"
              + "sr-policy:policies/"
              + "sr-policy:policy/"
              + "sr-policy:binding-sid/"
              + "sr-policy:mpls-sid/"
              + "sr-policy:value";
```

```
          }
          description "Reference to sr-policy MPLS binding SID value";
        }
      }
      case srv6 {
        leaf policy-srv6-binding-sid-value-ref {
          if-feature capability-candidate-path-binding-sid;
          type leafref {
            path "/rt:routing/"
              + "sr-policy:segment-routing/"
              + "sr-policy:traffic-engineering/"
              + "sr-policy:policies/"
              + "sr-policy:policy/"
              + "sr-policy:binding-sid/"
              + "sr-policy:srv6-sids/"
              + "sr-policy:srv6-sid/"
              + "sr-policy:value";
          }
          description "Reference to sr-policy SRv6 binding SID value";
        }
      }
    }

    leaf reason {
      type identityref {
        base sr-policy-types:binding-sid-unavailable-reason;
      }
      description
        "Reason why the binding sid is unavailable";
    }
  }

  notification sr-policy-candidate-path-binding-sid-mismatch-event {
    description
      "Notification event when binding sid of requested candidate
      path is different from the binding sid of the existing
      candidate path";

    leaf policy-color-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:color";
      }
      description "Reference to sr-policy color";
```

```
    }

    leaf policy-endpoint-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:endpoint";
      }
      description "Reference to sr-policy endpoint";
    }

    leaf existing-candidate-path-protocol-origin-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:candidate-paths/"
          + "sr-policy:candidate-path/"
          + "sr-policy:protocol-origin";
      }
      description
        "Reference to existing candidate path protocol origin";
    }

    leaf existing-candidate-path-preference-ref {
      type leafref {
        path "/rt:routing/"
          + "sr-policy:segment-routing/"
          + "sr-policy:traffic-engineering/"
          + "sr-policy:policies/"
          + "sr-policy:policy/"
          + "sr-policy:candidate-paths/"
          + "sr-policy:candidate-path/"
          + "sr-policy:preference";
      }
      description "Reference to existing candidate path preference";
    }

    choice dataplane-type {
      description
        "Binding SID dataplane choice";
      case mpls {
        leaf existing-candidate-path-mpls-binding-sid-value-ref {
```

```
          if-feature capability-candidate-path-binding-sid;
          type leafref {
            path "/rt:routing/"
              + "sr-policy:segment-routing/"
              + "sr-policy:traffic-engineering/"
              + "sr-policy:policies/"
              + "sr-policy:policy/"
              + "sr-policy:candidate-paths/"
              + "sr-policy:candidate-path/"
              + "sr-policy:binding-sid/"
              + "sr-policy:mpls-sid/"
              + "sr-policy:value";
          }
          description
            "Reference to existing candidate path MPLS binding SID "
          + "value";
        }
      }
      case srv6 {
        leaf existing-candidate-path-srv6-binding-sid-value-ref {
          if-feature capability-candidate-path-binding-sid;
          type leafref {
            path "/rt:routing/"
              + "sr-policy:segment-routing/"
              + "sr-policy:traffic-engineering/"
              + "sr-policy:policies/"
              + "sr-policy:policy/"
              + "sr-policy:candidate-paths/"
              + "sr-policy:candidate-path/"
              + "sr-policy:binding-sid/"
              + "sr-policy:srv6-sids/"
              + "sr-policy:srv6-sid/"
              + "sr-policy:value";
          }
          description
            "Reference to existing candidate path SRv6 binding SID "
          + "value";
        }
      }
    }

    leaf conflicting-candidate-path-protocol-origin {
      type uint8;
      description "Conflicting candidate path protocol origin";
    }

    leaf conflicting-candidate-path-preference {
      type uint32;
```

```
      description "Conflicting candidate path preference";
    }

    leaf conflicting-candidate-path-binding-sid-dataplane {
      type sr-policy-types:dataplane-type;
      description
        "Conflicting candidate path binding sid dataplane type";
    }

    leaf conflicting-candidate-path-binding-sid-value {
      type sr-policy-types:sid-value-type;
      description "Conflicting candidate path binding sid value";
    }
  }

  /* Features */

  feature capability-candidate-path-binding-sid {
    description
      "This feature enables the capability of specifying binding-sid
      for a candidate path.";
  }
}
<CODE ENDS>
```

Figure 4: ietf-sr-policy.yang

8. Security Considerations

The YANG modules defined in this document are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of the protocol operations and content.

The "ietf-sr-policy-types" module defines reusable identities, typedefs, and groupings. By itself, it does not define configuration or state data nodes, RPCs, actions, or notifications. As such, it does not by itself create additional security concerns beyond those that apply to modules that reuse these definitions.

The "ietf-sr-policy" module defines writable configuration data under /rt:routing/sr-policy:segment-routing/traffic-engineering. These writable data nodes are likely to be reasonably sensitive or

vulnerable in some network environments. Write or delete access to these nodes without proper protection can change the path that traffic follows, redirect traffic, black-hole traffic, or disrupt service. The following writable subtrees have particular sensitivities:

* /rt:routing/sr-policy:segment-routing/traffic-engineering/optimization-templates

* /rt:routing/sr-policy:segment-routing/traffic-engineering/attributes/segment-lists

* /rt:routing/sr-policy:segment-routing/traffic-engineering/attributes/explicit-binding-sid-rules

* /rt:routing/sr-policy:segment-routing/traffic-engineering/policies

Some of the readable data nodes and notifications in the "ietf-sr-policy" module may also be considered sensitive or vulnerable in some network environments because they expose policy identifiers, endpoints, candidate-path selection state, segment list contents, binding SID values, and failure reasons. In particular, read access to the operational state under /rt:routing/sr-policy:segment-routing/traffic-engineering/policies and to the notifications sr-policy-oper-state-change-event, sr-policy-candidate-path-change-event, sr-policy-binding-sid-unavailable-event, and sr-policy-candidate-path-binding-sid-mismatch-event should be controlled.

The YANG modules in this document do not define any RPC or action operations.

9. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

URI	Registrant	XML
urn:ietf:params:xml:ns:yang:ietf-sr-policy-types	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-sr-policy	The IESG	N/A

Table 3

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC7950]:

Name	Namespace	Prefix	Reference
ietf-sr-policy-types	urn:ietf:params:xml:ns:yang:ietf-sr-policy-types	sr-policy-types	This document
ietf-sr-policy	urn:ietf:params:xml:ns:yang:ietf-sr-policy	sr-policy	This document

Table 4

10. Acknowledgments

The authors of this document/YANG model would like to acknowledge the contributions/reviews by Johnson Thomas, Clarence Filsfils, Siva Sivabalan, Tarek Saad, Kris Michielsen, Dhanendra Jain, Ketan Talaulikar, Bhupendra Yadav, and Bruno Decraene.

11. References

11.1. 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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP ASSOCIATION Object Extensions", RFC 6780, DOI 10.17487/RFC6780, October 2012, <https://www.rfc-editor.org/info/rfc6780>.

[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>.

[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>.

[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>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

11.2. Informative References

[I-D.ietf-idr-segment-routing-te-policy] Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-segment-routing-te-policy-26, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-26>.

[I-D.ietf-spring-srv6-yang] Raza, S. K., Rajamanickam, J., Matsushima, S., Yu, P., and X. Liu, "YANG Data Model for SRv6 Base and Static", Work in Progress, Internet-Draft, draft-ietf-spring-srv6-yang-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-srv6-yang-05>.

[I-D.ietf-teas-rfc8776-update] Busi, I., Guo, A., Liu, X., Saad, T., and I. Bryskin, "Common YANG Data Types for Traffic Engineering", Work in Progress, Internet-Draft, draft-ietf-teas-rfc8776-update-23, 20 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-rfc8776-update-23>.

[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>.

Appendix A. Contributors

Robert Sawaya
Cisco Systems
Email: rsawaya@cisco.com

Daniel Voyer
Individual
Email: danvoyerwork@gmail.com

Muhammad Durrani
Oracle Corporation
Muahmmad.durrani@oracle.com

Authors' Addresses

Tarik Saleh (editor)
Cisco Systems
Email: tasaleh@cisco.com

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Zhuang Shunwan
Huawei Technologies
Email: zhuangshunwan@huawei.com

Satoru Matsushima
SoftBank
Email: satoru.matsushima@g.softbank.co.jp

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net