

SPRING Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 13, 2021

K. Raza  
S. Agarwal  
Cisco Systems

X. Liu  
Volta Networks

Z. Hu  
Huawei Technologies

I. Hussain  
Infinera Corporation

H. Shah  
Ciena Corporation

D. Voyer  
Bell Canada

S. Matsushima  
K. Horiba  
SoftBank

H. Elmalky

J. Rajamanickam  
A. AbdelSalam  
Cisco Systems

January 14, 2022

YANG Data Model for SRv6 Base and Static  
[draft-ietf-spring-srv6-yang-01](#)

Abstract

This document describes a YANG data model for Segment Routing IPv6 (SRv6) base. The model serves as a base framework for configuring and managing an SRv6 subsystem and expected to be augmented by other SRv6 technology models accordingly. Additionally, this document also specifies the model for the SRv6 Static application.

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

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/>.

Raza, et al.

Expires July 18, 2022

[Page 1]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

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

This Internet-Draft will expire on July 18, 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Specification of Requirements . . . . .	<a href="#">3</a>
<a href="#">3.</a>	YANG Model . . . . .	<a href="#">4</a>
<a href="#">3.1.</a>	Overview . . . . .	<a href="#">4</a>
<a href="#">3.2.</a>	SRv6 Types . . . . .	<a href="#">4</a>
<a href="#">3.3.</a>	SRv6 Base . . . . .	<a href="#">5</a>
<a href="#">3.3.1.</a>	Configuration . . . . .	<a href="#">5</a>
<a href="#">3.3.2.</a>	State . . . . .	<a href="#">6</a>
<a href="#">3.3.3.</a>	Notification . . . . .	<a href="#">8</a>
<a href="#">3.4.</a>	SRv6 Static . . . . .	<a href="#">9</a>

3.4.1.	Configuration . . . . .	9
3.4.2.	State . . . . .	15
3.4.3.	Notification . . . . .	15
4.	Pending Items . . . . .	15
5.	YANG Specification . . . . .	15
5.1.	SRv6 Types . . . . .	15
5.2.	SRv6 Base . . . . .	31
5.3.	SRv6 Static . . . . .	47
6.	Security Considerations . . . . .	70
7.	IANA Considerations . . . . .	71
8.	Acknowledgments . . . . .	71
9.	References . . . . .	72
9.1.	Normative References . . . . .	72
9.2.	Informative References . . . . .	74

Authors' Addresses . . . . .	74
------------------------------	----

## 1. Introduction

The Network Configuration Protocol (NETCONF) [[RFC6241](#)] is one of the network management protocols that defines mechanisms to manage network devices. YANG [[RFC6020](#)] is a modular language that represents data structures in an XML tree format, and is used as a data modeling language for the NETCONF.

Segment Routing (SR), as defined in [[RFC8402](#)], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain. When applied to ipv6 data-plane (i.e. SRv6), SR requires a type of routing header (SRH) in an IPv6 packet that is used to encode an ordered list of IPv6 addresses (SIDs). The active segment is indicated by the Destination Address of the packet, and the next segment is indicated by a pointer in the SRH [[RFC8754](#)]. The various functions and behaviors corresponding to network programming using SRv6 are specified in [[RFC8986](#)].

This document introduces a YANG data model for base SRv6 that would serve as a base framework for configuring and managing an SRv6 subsystem. As needed, other SRv6 technology models (e.g. ISIS, OSPFv3, BGP, EVPN, Service Chaining) may augment this model.

Furthermore, to illustrate basic behaviors as captured in [[RFC8986](#)], this document also specifies a YANG model for the SRv6-Static application.

The model currently defines the following constructs that are used for managing SRv6:

- o Configuration
- o Operational State
- o Notifications

## [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.](#) YANG Model

### [3.1.](#) Overview

This document defines following three new YANG modules:

- o ietf-srv6-types: defines common and basic types related to SRv6
- o ietf-srv6-base: specifies management model for SRv6 base constructs (locator, SIDs, etc.)
- o ietf-srv6-static: specifies management model for SRv6-static application

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) defined in [[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.

In this document, when a simplified graphical representation of YANG model is presented in a tree diagram, the meaning of the symbols in these tree diagrams is defined in [\[RFC8340\]](#).

### [3.2.](#) SRv6 Types

SRv6 common types and definitions are defined in the new module "ietf-srv6-types". The main types defined in this module include:

- o srv6-sid: SRv6 SID
- o srv6-func-value: Typedef for FUNC value in an SRv6 SID
- o srv6-func-value-reserved-type: Enum (list) of "reserved" FUNC opcode
- o srv6-endpoint-type: SRv6 Endpoint behaviors [\[RFC8986\]](#) identity type
- o srv6-headend-type: SRv6 Headend behaviors [\[RFC8986\]](#) identity type
- o srv6-security-type: SRv6 Security rules identity type
- o srv6-counter-type: SRv6 Counter type [\[RFC8986\]](#) identity type

The corresponding YANG specification for this module is captured in [Section 5.1](#).

### [3.3.](#) SRv6 Base

The base SRv6 model is specified in ietf-srv6-base module. This module augments "/rt:routing:/sr:segment-routing" [\[RFC9020\]](#) and specifies the configuration, operational state, and notification events that are required to manage the base SRv6.

The corresponding YANG specification for this module is captured in [Section 5.2](#).

### [3.3.1.](#) Configuration

The module defines some fundamental items required to configure an SRv6 network:

- o SRv6 Enablement: Enable Segment-Routing SRv6 feature
- o Encapsulation Parameters: Provide encapsulation related parameters (such as source-address, hop-limit, and traffic-class) to be used when performing T.Encap\* operation.
- o Locator(s) Specification: SRv6 locator is a fundamental construct for an SRv6 network. This is the construct from which SID (function values) are allocated that on the local box, and advertised to and used by remote nodes for reachability. A locator is identified by a name and has associated prefix and [IGP] algorithm. It can be configured as an anycast locator. It is possible to have more than one locator on a node (e.g. locator per algorithm, anycast and non-anycast locators, etc.).

Following is a simplified graphical tree representation of the data model for SRv6 base configuration

```
module: ietf-srv6-base
  augment /rt:routing/sr:segment-routing:
    +--rw srv6
      +--rw enable?          boolean
      +--rw encapsulation
      |   +--rw source-address?  inet:ipv6-address
```

```

|   +---rw hop-limit
|   |   +---rw value?          uint8
|   |   +---rw propagate?     boolean
|   +---rw traffic-class
|       +---rw value?          uint8
|       +---rw propagate?     boolean
+---rw locators
    +---rw locator* [name]
        +---rw name                string
        +---rw enable?             boolean
        +---rw prefix
            |   +---rw address      inet:ipv6-address
            |   +---rw length      srv6-types:srv6-locator-len
        +---rw algorithm?          uint32
        +---rw anycast?            boolean

```

Figure 1: SRv6 Base - Config Tree

### 3.3.2. State

As per NMDA model, the state related to configuration items specified in above section [Section 3.3.1](#) can be retrieved from the same tree. This section defines other operational state items related to SRv6 base.

The operational state corresponding to the SRv6 base includes:

- o node capabilities: provides information on the node (hardware) capabilities and support regarding various SRv6 aspects and features including endpoint behaviors, headend behaviors, security rules, counter/stats support, and other SRv6 parameters that need to be signaled in an SRv6 network by the protocols.
- o locator: provides information related to a locator. The information includes locator operational state, and state of address conflict with any ipv6 address configured on local interfaces etc.
- o local-sid: provides information related to local-SIDs allocated and/or installed on the node. This includes two types of information:

1. aggregate across all local-SIDs such as aggregate counters
2. per local-SID information such as allocation type (dynamic or explicit), SID owner protocol(s)/client(s), forwarding [paths] information, and stats/counters.

Following is a simplified graphical tree representation of the data model for the SRv6 operational state (for read-only items):

```

module: ietf-srv6-base
augment /rt:routing/sr:segment-routing:
  +--rw srv6
    +--rw locators
      | +--rw locator* [name]
      |   +--rw name string
      |   +--ro operational-status? srv6-types:srv6-status-type
      |   +--ro is-in-address-conflict? boolean
    +--ro node-capabilities
      | +--ro end-behavior* [type]
      | | +--ro type identityref
      | | +--ro supported boolean
      | +--ro headend-behavior* [type]
      | | +--ro type identityref
      | | +--ro supported boolean
      | +--ro msd
      | | +--ro max-sl? uint8
      | | +--ro max-end-pop? uint8
      | | +--ro max-h_encap? uint8
      | | +--ro max-end_d? uint8
      | +--ro security-rule* [type]
      | | +--ro type identityref
      | | +--ro supported boolean
      | +--ro counters* [type]
      | | +--ro type identityref
      | | +--ro supported boolean
    +--ro local-sids
      +--ro counters
      | +--ro cnt-3
      | | +--ro in-pkts? yang:counter64
      | | +--ro in-octets? yang:counter64
      +--ro local-sid* [sid]
        +--ro sid srv6-types:srv6-sid
        +--ro locator? -> /rt:routing/sr:segment-routing/srv6:
        +--ro is-reserved? boolean
        +--ro end-behavior-type? identityref
        +--ro alloc-type? srv6-types:sid-alloc-type
        +--ro owner* [type instance]
  
```



```

|   +---ro type          identityref
|   +---ro instance      string
|   +---ro is-winner?    boolean
+---ro forwarding
|   +---ro is-installed?  boolean
|   +---ro next-hop-type?  srv6-types:srv6-nexthop-type
|   +---ro paths
|       +---ro path* [path-index]
|           +---ro path-index    uint8
|           +---ro l2
|               |   +---ro interface?    if:interface-ref
|               +---ro l3
|                   |   +---ro interface?          if:interface-ref
|                   |   +---ro next-hop?           inet:ip-address
|                   |   +---ro weight?             uint32
|                   |   +---ro role?               enumeration
|                   |   +---ro backup-path-index?  uint8
|               +---ro (encap-type)?
|                   +---:(srv6)
|                       |   +---ro out-sid* [sid]
|                       |       +---ro sid    srv6-types:srv6-sid
|                   +---:(mpls)
|                       +---ro out-label* [label]
|                           +---ro label    rt-types:mpls-label
+---ro counters
    +---ro cnt-1
        +---ro in-pkts?    yang:counter64
        +---ro in-octets?  yang:counter64

```

Figure 2: SRv6 Base - State Tree

### 3.3.3. Notification

This model defines a list of notifications to inform an operator of important events detected during the SRv6 operation. These events include events related to:

- o locator operational state changes
- o local-SID collision event

Following is a simplified graphical tree representation of the data

module: ietf-srv6-base

notifications:

```

+---n srv6-locator-status-event
|   +---ro operational-status?   srv6-types:srv6-status-type
|   +---ro locator?              -> /rt:routing/sr:segment-routing/srv6:srv6/
+---n srv6-sid-collision-event
    +---ro sid?                  srv6-types:srv6-sid
    +---ro existing
    |   +---ro end-behavior-type? identityref
    +---ro requested
        +---ro end-behavior-type? identityref

```

Figure 3: SRv6 Base - Notification Tree

### [3.4.](#) SRv6 Static

SRv6-Static application allows a user to specify SRv6 local SIDs and program them in the forwarding plane. The SRv6-Static model is captured in the ietf-srv6-static module.

The associated YANG specification for this module is captured in [Section 5.3](#).

#### [3.4.1.](#) Configuration

The SRv6-Static configuration augments the SRv6-base locator tree `"/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator"`

Following are salient features of the SRv6-Static config model:

- o Allows static (explicit) configuration for local-SIDs under a given locator.
- o Given that entry is scoped under a locator, the key for each entry

is "function" value.

- o A user must also specify end-behavior type (End\*) associated with the entry.
- o A user must also specify behavior-specific data with each entry. For example, for any end behavior requiring a table lookup, a lookup-table need be provided. Similarly, for any end behavior with forwarding next-hops need to specify next-hop information. The example of former include End, End.T, End.DT4, End.DT6, and End.DT46, whereas example of later include End.X, End.DX4, End.DX6, End.B6, End.BM etc.

- o Each local-SID entry has zero or more forwarding paths specified.
- o A forwarding path has next-hop type that depends on the end behavior, and could be either ipv6, or ipv4, or mpls, or l2 type. For example, End.X, End.DX4, End.DX6, End.B6, End.BM, and End.DX2 will have ipv6, ipv4, ipv6, ipv6, mpls, and l2 next-hop types respectively
- o For each forwarding next-hop type, the appropriate path attributes are to be specified as well. For L2 type, the only other information required is the L2 interface name. Whereas for L3 (ipv6, ipv4, mpls) types, the information includes L3 interface name, next-hop IP address, weight, and protection information.
- o Depending on the end behavior type, a forwarding path may have either MPLS or SRv6 encapsulation -- i.e., Stack of out-labels or Stack of SRv6 out-SIDs. The example of former is End.BM and example of later include the rest (End.X, End.DX4, End.DX6, End.B6 etc.).

Following is a simplified graphical tree representation of the data model for SRv6 Static configuration

```
module: ietf-srv6-static
  augment /rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator:
    +--rw static
      +--rw local-sids
        +--rw sid* [function]
```

```

+---rw function                srv6-types:srv6-func-value
+---rw end-behavior-type      identityref
+---rw end
+---rw end_psp
+---rw end_usp
+---rw end_psp_usp
+---rw end_usd
+---rw end_psp_usd
+---rw end_usp_usd
+---rw end_psp_usp_usd
+---rw end-t
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_psp
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_usp
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_psp_usp
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_usd

```

```

|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_psp_usd
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_usp_usd
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-t_psp_usp_usd
|   +---rw lookup-table-ipv6   srv6-types:table-id
+---rw end-x
|   +---rw protected?         boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index         uint8
|           +---rw interface?         if:interface-ref
|           +---rw next-hop?         inet:ipv6-address
|           +---rw table?            srv6-types:table-id
|           +---rw weight?           uint32
|           +---rw role?             enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid         srv6-types:srv6-sid
+---rw end-x_psp

```

```

|      +---rw protected?    boolean
|      +---rw paths
|          +---rw path* [path-index]
|              +---rw path-index          uint8
|              +---rw interface?          if:interface-ref
|              +---rw next-hop?          inet:ipv6-address
|              +---rw table?             srv6-types:table-id
|              +---rw weight?            uint32
|              +---rw role?              enumeration
|              +---rw backup-path-index?  uint8
|              +---rw sid-list
|                  +---rw out-sid* [sid]
|                      +---rw sid          srv6-types:srv6-sid
+---rw end-x_osp
|      +---rw protected?    boolean
|      +---rw paths
|          +---rw path* [path-index]
|              +---rw path-index          uint8
|              +---rw interface?          if:interface-ref
|              +---rw next-hop?          inet:ipv6-address
|              +---rw table?             srv6-types:table-id
|              +---rw weight?            uint32
|              +---rw role?              enumeration
|              +---rw backup-path-index?  uint8
|              +---rw sid-list
|                  +---rw out-sid* [sid]

```

```

|          +---rw sid          srv6-types:srv6-sid
+---rw end-x_psp_osp
|      +---rw protected?    boolean
|      +---rw paths
|          +---rw path* [path-index]
|              +---rw path-index          uint8
|              +---rw interface?          if:interface-ref
|              +---rw next-hop?          inet:ipv6-address
|              +---rw table?             srv6-types:table-id
|              +---rw weight?            uint32
|              +---rw role?              enumeration
|              +---rw backup-path-index?  uint8
|              +---rw sid-list
|                  +---rw out-sid* [sid]
|                      +---rw sid          srv6-types:srv6-sid

```

```

+---rw end-x_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index          uint8
|           +---rw interface?          if:interface-ref
|           +---rw next-hop?           inet:ipv6-address
|           +---rw table?              srv6-types:table-id
|           +---rw weight?             uint32
|           +---rw role?               enumeration
|           +---rw backup-path-index?  uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid          srv6-types:srv6-sid
+---rw end-x_psp_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index          uint8
|           +---rw interface?          if:interface-ref
|           +---rw next-hop?           inet:ipv6-address
|           +---rw table?              srv6-types:table-id
|           +---rw weight?             uint32
|           +---rw role?               enumeration
|           +---rw backup-path-index?  uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid          srv6-types:srv6-sid
+---rw end-x_usp_usd
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index          uint8

```

```

|   +---rw interface?          if:interface-ref
|   +---rw next-hop?           inet:ipv6-address
|   +---rw table?              srv6-types:table-id
|   +---rw weight?             uint32
|   +---rw role?               enumeration
|   +---rw backup-path-index?  uint8
|   +---rw sid-list
|       +---rw out-sid* [sid]

```

```

|           +---rw sid      srv6-types:srv6-sid
+---rw end-x_psp_osp_osp
|   +---rw protected?    boolean
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?       inet:ipv6-address
|           +---rw table?          srv6-types:table-id
|           +---rw weight?         uint32
|           +---rw role?           enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid      srv6-types:srv6-sid
+---rw end-b6-encaps
|   +---rw policy-name      string
|   +---rw source-address    inet:ipv6-address
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?       inet:ipv6-address
|           +---rw table?          srv6-types:table-id
|           +---rw weight?         uint32
|           +---rw role?           enumeration
|           +---rw backup-path-index? uint8
|           +---rw sid-list
|               +---rw out-sid* [sid]
|                   +---rw sid      srv6-types:srv6-sid
+---rw end-bm
|   +---rw policy-name      string
|   +---rw paths
|       +---rw path* [path-index]
|           +---rw path-index      uint8
|           +---rw interface?      if:interface-ref
|           +---rw next-hop?       inet:ip-address
|           +---rw weight?         uint32
|           +---rw role?           enumeration
|           +---rw backup-path-index? uint8

```

```

|           +---rw sid-list

```

```

|             +---rw out-sid* [sid]
|             +---rw sid      srv6-types:srv6-sid
+---rw end-dx6
|   +---rw paths
|     +---rw path* [path-index]
|       +---rw path-index      uint8
|       +---rw interface?      if:interface-ref
|       +---rw next-hop?       inet:ipv6-address
|       +---rw table?          srv6-types:table-id
|       +---rw weight?         uint32
|       +---rw role?           enumeration
|       +---rw backup-path-index? uint8
|       +---rw sid-list
|         +---rw out-sid* [sid]
|         +---rw sid      srv6-types:srv6-sid
+---rw end-dx4
|   +---rw paths
|     +---rw path* [path-index]
|       +---rw path-index      uint8
|       +---rw interface?      if:interface-ref
|       +---rw next-hop?       inet:ipv4-address
|       +---rw table?          srv6-types:table-id
|       +---rw weight?         uint32
|       +---rw role?           enumeration
|       +---rw backup-path-index? uint8
|       +---rw sid-list
|         +---rw out-sid* [sid]
|         +---rw sid      srv6-types:srv6-sid
+---rw end-dt6
|   +---rw lookup-table-ipv6    srv6-types:table-id
+---rw end-dt4
|   +---rw lookup-table-ipv4    srv6-types:table-id
+---rw end-dt46
|   +---rw lookup-table-ipv4    srv6-types:table-id
|   +---rw lookup-table-ipv6    srv6-types:table-id
+---rw end-dx2
|   +---rw path
|     +---rw l2-interface      if:interface-ref
+---rw end-dx2v
|   +---rw lookup-table-vlan    srv6-types:table-id
+---rw end-dt2u
|   +---rw lookup-table-mac     srv6-types:table-id
+---rw end-dt2m
|   +---rw flooding-table      srv6-types:table-id
|   +---rw paths
|     +---rw path* [path-index]
|       +---rw path-index      uint8

```



```
+-rw l2-interface?  if:interface-ref
```

Figure 4: SRv6 Static - Config Tree

### [3.4.2.](#) State

As per NMDA model, the state related to configuration items specified in above section [Section 3.4.1](#) can be retrieved from the same tree. The state regarding the local-SIDs created by SRv6-static model can be obtained using the state model of SRv6-base. Hence, there is no additional state identified at this time for SRv6-static.

### [3.4.3.](#) Notification

None.

## [4.](#) Pending Items

Following are the items that will be addressed in next revisions:

- o Extend local-SID collision event/notification in SRv6-base model.
- o Add RPC support in the SRv6-base model.
- o Add ARGS support in the SRv6-Static model.
- o QoS support

## [5.](#) YANG Specification

Following are actual YANG definition for SRv6 modules defined earlier in the document.

### [5.1.](#) SRv6 Types

This YANG module imports types defined in [[RFC6991](#)].

Moreover, the module models behaviors defined in [[RFC8986](#)], [[I-D.ietf-spring-sr-service-programming](#)], and [[I-D.ietf-dmm-srv6-mobile-uplane](#)].

```
<CODE BEGINS> file "ietf-srv6-types@2022-01-14.yang"
```

```
// RFC Editor: replace the above date with the date of
```

// publication and remove this note.

```
module ietf-srv6-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-types";
  prefix srv6-types;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data 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:   Jaganbabu Rajamanickam
              <mailto:jrajaman@cisco.com>

    Editor:   Xufeng Liu
              <mailto:xufeng.liu.ietf@gmail.com>

    Editor:   Zhibo Hu
              <mailto:huzhibo@huawei.com>

    Editor:   Iftekhar Hussain
              <mailto:IHussain@infinera.com>

    Editor:   Himanshu Shah
              <mailto:hshah@ciena.com>

    Editor:   Daniel Voyer
              <mailto:daniel.voyer@bell.ca>

    Editor:   Hani Elmalky
```

<mailto:helmalky@google.com>

Editor: Satoru Matsushima  
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba  
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam

Raza, et al.

Expires July 18, 2022

[Page 16]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

<mailto:ahabdel@cisco.com>

";

description

"This YANG module defines the essential types for the management of Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

reference "RFC XXXX";

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

revision 2022-01-14 {

description

"Alignment with SRv6 net-pgm rev16";

reference

"RFC XXXX: YANG Data Model for SRv6";

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

}

```

revision 2019-10-30 {
  description
    "Renaming of some types";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

revision 2019-07-08 {
  description
    "Alignment with latest SRv6 network programming";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note

```

```

}

revision 2018-10-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

identity srv6-endpoint-type {
  description
    "Base identity from which specific SRv6 Endpoint types are
    derived.";
}

/* Endpoints defined under draft-ietf-spring-
 * srv6-network-programming */

identity End {
  base srv6-endpoint-type;
  description
    "End function (variant: no PSP, no USP).";
  reference

```

```

        "RFC 8986";
    }

    identity End_PSP {
        base srv6-endpoint-type;
        description
            "End function (variant: PSP only).";
        reference
            "RFC 8986";
    }

    identity End_USP {
        base srv6-endpoint-type;
        description
            "End function (variant: USP only).";
        reference
            "RFC 8986";
    }

    identity End_PSP_USP {
        base srv6-endpoint-type;
        description
            "End function (variant: PSP and USP).";
    }

```

```

        reference
            "RFC 8986";
    }

    identity End.X {
        base srv6-endpoint-type;
        description
            "Endpoint with cross-connect to an array
            of layer-3 adjacencies (variant: no PSP, no USP).";
        reference
            "RFC 8986";
    }

    identity End.X_PSP {
        base srv6-endpoint-type;
        description
            "Endpoint with cross-connect to an array
            of layer-3 adjacencies (variant: PSP only).";
    }

```

```

        reference
            "RFC 8986";
    }

    identity End.X_USP {
        base srv6-endpoint-type;
        description
            "Endpoint with cross-connect to an array
            of layer-3 adjacencies (variant: USP only).";
        reference
            "RFC 8986";
    }

    identity End.X_PSP_USP {
        base srv6-endpoint-type;
        description
            "Endpoint with cross-connect to an array
            of layer-3 adjacencies (variant: PSP and USP).";
        reference
            "RFC 8986";
    }

    identity End.T {
        base srv6-endpoint-type;
        description
            "Endpoint with specific IPv6 table lookup
            (variant: no PSP, no USP).";
        reference
            "RFC 8986";
    }

```

```

    identity End.T_PSP {
        base srv6-endpoint-type;
        description
            "Endpoint with specific IPv6 table lookup
            (variant: PSP only).";
        reference
            "RFC 8986";
    }

    identity End.T_USP {
        base srv6-endpoint-type;

```

```

    description
        "Endpoint with specific IPv6 table lookup
        (variant: USP only).";
    reference
        "RFC 8986";
}

identity End.T_PSP_USP {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP and USP).";
    reference
        "RFC 8986";
}

identity End.B6.Encaps {
    base srv6-endpoint-type;
    description
        "Endpoint bound to an SRv6 Policy
        where the SRv6 Policy also includes an
        IPv6 Source Address A.";
    reference
        "RFC 8986";
}

identity End.BM {
    base srv6-endpoint-type;
    description
        "Endpoint bound to an SR-MPLS Policy";
    reference
        "RFC 8986";
}

identity End.DX6 {
    base srv6-endpoint-type;
    description

```

```

        "Endpoint with decapsulation and cross-connect
        to an array of IPv6 adjacencies";
    reference
        "RFC 8986";

```

```

}

identity End.DX4 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and cross-connect
        to an array of IPv4 adjacencies";
    reference
        "RFC 8986";
}

identity End.DT6 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific
        IPv6 table lookup";
    reference
        "RFC 8986";
}

identity End.DT4 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific
        IPv4 table lookup";
    reference
        "RFC 8986";
}

identity End.DT46 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and specific IP
        (IPv4 or IPv6) table lookup";
    reference
        "RFC 8986";
}

identity End.DX2 {
    base srv6-endpoint-type;
    description
        "Endpoint with decapsulation and Layer-2
        cross-connect to an L2 interface";
    reference

```



```
        "RFC 8986";
    }

    identity End.DX2V {
        base srv6-endpoint-type;
        description
            "Endpoint with decapsulation and specific
             VLAN L2 table lookup";
        reference
            "RFC 8986";
    }

    identity End.DT2U {
        base srv6-endpoint-type;
        description
            "Endpoint with decapsulation and specific
             unicast MAC L2 table lookup";
        reference
            "RFC 8986";
    }

    identity End.DT2M {
        base srv6-endpoint-type;
        description
            "Endpoint with decapsulation and specific L2 table
             flooding";
        reference
            "RFC 8986";
    }

    identity End.B6.Encaps.Red {
        base srv6-endpoint-type;
        description
            "This is a reduced encap variation of the End.B6.Encap
             behavior.";
        reference
            "RFC 8986";
    }

    identity End_USD {
        base srv6-endpoint-type;
        description
            "End function (variant: USD).";
        reference
            "RFC 8986";
    }
}
```

identity End.PSP\_USD {

```
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USD).";
    reference
        "RFC 8986";
}
```

```
identity End.USD_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: USD and USD).";
    reference
        "RFC 8986";
}
```

```
identity End.PSP_USD_USD {
    base srv6-endpoint-type;
    description
        "End function (variant: PSP and USD and USD).";
    reference
        "RFC 8986";
}
```

```
identity End.X_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: USD).";
    reference
        "RFC 8986";
}
```

```
identity End.X_PSP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with cross-connect to an array
        of layer-3 adjacencies (variant: PSP and USD).";
    reference
        "RFC 8986";
}
```

```

identity End.X_USP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with cross-connect to an array
      of layer-3 adjacencies (variant: USP and USD).";
  reference
    "RFC 8986";
}

```

```

}

identity End.X_PSP_USP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with cross-connect to an array
      of layer-3 adjacencies (variant: PSP and USP and USD).";
  reference
    "RFC 8986";
}

identity End.T_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with decapsulation and Layer-2
      cross-connect to an L2 interface";
  reference
    "RFC 8986";
}

identity End.T_PSP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup
      (variant: PSP and USD).";
  reference
    "RFC 8986";
}

identity End.T_USP_USD {
  base srv6-endpoint-type;
  description
    "Endpoint with specific IPv6 table lookup

```

```

        (variant: USP and USD).";
    reference
        "RFC 8986";
}

identity End.T_PSP_USP_USD {
    base srv6-endpoint-type;
    description
        "Endpoint with specific IPv6 table lookup
        (variant: PSP and USP and USD).";
    reference
        "RFC 8986";
}

/* Endpoints defined under draft-ietf-spring-sr-service-programming */

```

```

identity End.AN {
    base srv6-endpoint-type;
    description
        "Service-Chaining SR Aware function (native)";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AS {
    base srv6-endpoint-type;
    description
        "Service-Chaining Static proxy";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AD {
    base srv6-endpoint-type;
    description
        "Service-Chaining Dynamic proxy";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

```

```

identity End.AM {
    base srv6-endpoint-type;
    description
        "Service-Chaining Masquerading SR proxy";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AM_NAT {
    base srv6-endpoint-type;
    description
        "Service-Chaining Masquerading SR proxy with NAT";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AM_CACHE {
    base srv6-endpoint-type;
    description

```

```

        "Service-Chaining Masquerading SR proxy with Caching";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.AM_NAT_CACHE {
    base srv6-endpoint-type;
    description
        "Service-Chaining Masquerading SR proxy with NAT and Caching";
    reference
        "draft-ietf-spring-sr-service-programming-05";
    // RFC Editor: replace with actual RFC number and remove this note
}

/* Endpoints defined under draft-ietf-dmm-srv6-mobile-uplane */

identity End.MAP {
    base srv6-endpoint-type;

```

```

    description
        "DMM End.MAP";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-17";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.Limit {
    base srv6-endpoint-type;
    description
        "DMM End.Limit";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-17";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.D {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP6.D";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-17";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.Di {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP6.Di";

```

```

    reference
        "draft-ietf-dmm-srv6-mobile-uplane-17";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity End.M.GTP6.E {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP6.E";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note

```

```

}

identity End.M.GTP4.E {
    base srv6-endpoint-type;
    description
        "DMM End.M.GTP4.E";
    reference
        "draft-ietf-dmm-srv6-mobile-uplane-05";
    // RFC Editor: replace with actual RFC number and remove this note
}
/* DMM end */

identity srv6-headend-type {
    description
        "Base identity from which SRv6 headend rule types are derived.";
}

identity H.Encaps {
    base srv6-headend-type;
    description
        "Headend rule H.Encaps with encapsulated of an SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-16";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity H.Encaps.Red {
    base srv6-headend-type;
    description
        "Headend rule H.Encaps.Red with reduced encap of an
        SRv6 policy";
    reference
        "draft-ietf-spring-srv6-network-programming-16";
    // RFC Editor: replace with actual RFC number and remove this note
}

```

```

identity H.Encaps.L2 {
    base srv6-headend-type;
    description
        "Headend rule H.Encaps.l2 on the received L2 frame";
    reference

```

```

        "draft-ietf-spring-srv6-network-programming-16";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity H.Encaps.L2.Red {
    base srv6-headend-type;
    description
        "Headend rule H.Encaps.L2.Red on the received L2 frame";
    reference
        "draft-ietf-spring-srv6-network-programming-16";
    // RFC Editor: replace with actual RFC number and remove this note
}

identity srv6-security-type {
    description
        "Base identity from which SRv6 Security rule types are
        derived.";
}

identity SEC-1 {
    base srv6-security-type;
    description
        "Support an ACL on the external interface that
        drops any traffic with SA or DA in the internal SID space.
        (aka Security rule SEC-1)";
}

identity SEC-2 {
    base srv6-security-type;
    description
        "Support an ACL that drops any traffic on local SID as DA
        from a source SA that does not belong to internal address
        or SID space. (aka Security rule SEC-2)";
}

identity srv6-counter-type {
    description
        "Base identity from which SRv6 counter types are derived.";
}

identity LocalSID {
    base srv6-counter-type;

```



```

    description
        "Count packets and bytes traffic that matched that SID and
        was processed successfully (aka Counter CNT-1)";
    reference
        "RFC 8986";
    // RFC Editor: replace with actual RFC number and remove this note
}

typedef srv6-sid {
    type inet:ipv6-prefix;
    description
        "This type defines a SID value in SRv6";
}

typedef srv6-func-value {
    type uint32;
    description
        "This is a typedef for SID's FUNC value";
}

typedef srv6-func-value-reserved-type {
    type enumeration {
        enum invalid { value 0; description "Invalid function value"; }
    }

    description "SRv6 SID's FUNC Reserved values";
}

typedef srv6-locator-len {
    type uint8 {
        range "32 .. 96";
    }
    description
        "This type defines an SRv6 locator len with range constraints";
}

typedef srv6-sid-pfxlen {
    type uint8 {
        range "32 .. 128";
    }
    default 128;
    description
        "This type defines a SID prefixlen with range constraints";
}

typedef sid-alloc-type {
    type enumeration {

```

Internet-Draft

YANG Data Model for SRv6

January 2022

```
    enum Dynamic {
      description
        "SID allocated dynamically.";
    }
    enum Explicit {
      description
        "SID allocated with explicit (static) value";
    }
  }
  description
    "Types of sid allocation used.";
}

identity srv6-sid-owner-type {
  description
    "Base identity from which SID owner types are derived.";
}

identity isis {
  base srv6-sid-owner-type;
  description "ISIS";
}

identity ospfv3 {
  base srv6-sid-owner-type;
  description "OSPFv3";
}

identity bgp {
  base srv6-sid-owner-type;
  description "BGP";
}

identity evpn {
  base srv6-sid-owner-type;
  description "EVPN";
}

identity sr-policy {
  base srv6-sid-owner-type;
  description "SR Policy";
}
```

```

identity service-function {
    base srv6-sid-owner-type;
    description "SF";
}

```

```

typedef table-id {
    type uint32;
    description
        "Routing/switching/bridging/VLAN Table Id";
}

typedef srv6-status-type {
    type enumeration {
        enum up { value 1; description "State is Up"; }
        enum down { description "State is Down"; }
    }
    description
        "Status type";
}

typedef srv6-nexthop-type {
    type enumeration {
        enum ipv4 { value 1; description "IPv4 next-hop"; }
        enum ipv6 { description "IPv6 next-hop"; }
        enum mpls { description "MPLS next-hop"; }
        enum l2 { description "L2 next-hop"; }
    }
    description
        "Forwarding Next-hop type";
}

} // module

<CODE ENDS>

```

Figure 5: ietf-srv6-types.yang

## [5.2.](#) SRv6 Base

This YANG module imports types defined in [[RFC6991](#)], [[RFC8294](#)], [[RFC8343](#)], and [[RFC8349](#)].

```
<CODE BEGINS> file "ietf-srv6-base@2022-01-14.yang"

// RFC Editor: replace the above date with the date of
// publication and remove this note.

module ietf-srv6-base {
  yang-version 1.1;
```

Raza, et al.

Expires July 18, 2022

[Page 31]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

```
namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-base";
prefix srv6;

import ietf-interfaces {
  prefix "if";
  reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-inet-types {
  prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-yang-types {
  prefix "yang";
  reference "RFC 6991: Common YANG Data Types";
}

import ietf-routing-types {
  prefix "rt-types";
  reference "RFC 8294: Common YANG Data Types for the Routing Area";
}

import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
```

```

}

import ietf-segment-routing {
    prefix sr;
    reference "draft-ietf-spring-sr-yang";
}

import ietf-srv6-types {
    prefix srv6-types;
    reference "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

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: Jaganbabu Rajamanickam  
<mailto:jrajanaman@cisco.com>

Editor: Xufeng Liu  
<mailto:Xufeng\_Liu@jabril.com>

Editor: Zhibo Hu  
<mailto:huzhibo@huawei.com>

Editor: Iftekhar Hussain  
<mailto:IHussain@infinera.com>

Editor: Himanshu Shah  
<mailto:hshah@ciena.com>

Editor: Daniel Voyer  
<mailto:daniel.voyer@bell.ca>

Editor: Hani Elmalky  
<mailto:helmalky@google.com>

Editor: Satoru Matsushima  
<mailto:satoru.matsushima@gmail.com>

Editor: Katsuhiro Horiba  
<mailto:katsuhiro.horiba@g.softbank.co.jp>

Editor: Ahmed AbdelSalam  
<mailto:ahabdel@cisco.com>

";

description

"This YANG module defines the essential elements for the management of Segment-Routing with IPv6 dataplane (SRv6).

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

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

Raza, et al.

Expires July 18, 2022

[Page 33]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

reference "RFC XXXX";

revision 2022-01-14 {

description

"Alignment with SRv6 network programming rev16";

reference

"RFC XXXX: YANG Data Model for SRv6";

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

// this note

}

revision 2019-10-30 {

description

"Alignment with SRv6 network programming";

```

    reference
      "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  }

  revision 2019-07-08 {
    description
      "Alignment with SRv6 network programming";
    reference
      "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  }

  revision 2018-10-22 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  }

/*
 * Common
 */

  grouping path-attrs-cmn {
    description
      "Path properties -common for v4/v6";

    leaf weight {
      type uint32;

```

```

    description
      "This value is used to compute a loadshare to perform un-equal
      load balancing when multiple outgoing path(s) are specified. A
      share is computed as a ratio of this number to the total under
      all configured path(s).";
  }

```

```

leaf role {
  type enumeration {
    enum PRIMARY { description "Path as primary traffic carrying"; }
    enum BACKUP { description "Path acts as a backup"; }
    enum PRIMARY_AND_BACKUP {
      description "Path acts as primary and backup simultaneously"; }
    }
  description "The path role";
}

leaf backup-path-index {
  type uint8;
  description "Index of the protecting (backup) path";
}
}

grouping path-out-sids {
  description "Grouping for path's SID stack";

  list out-sid {
    key "sid";
    description "Out SID";

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

grouping path-out-labels {
  description "Grouping for path's label stack";

  list out-label {
    key "label";
    description "Out label";

    leaf label {
      type rt-types:mpls-label;
      description "Label value";
    }
  }
}

```



```

    }

}

/*
 * Config and State
 */

grouping srv6-encap {
    description "Grouping for encap param config.";

    container encapsulation {
        description "Configure encapsulation related parameters";
        leaf source-address {
            type inet:ipv6-address;
            description "Specify a source address (for T.Encap).
                        The address must locally exists and be routable";
        }
        container hop-limit {
            description "Configure IPv6 header's Hop-limit options";
            leaf value {
                type uint8;
                default 64;
                description "Set encapsulating outer IPv6 header's Hoplimit
                            field to specified value when doing
                            encapsulation";
            }
        }
        leaf propagate {
            type boolean;
            default false;
            description "IP TTL/Hop-limit propagation from encapsulated
                        packet to encapsulating outer IPv6 header's
                        Hoplimit field. When configured on decapsulation
                        side, this refers to propagating Hop-limit from
                        outer IPv6 header to inner header after decap";
        }
    }
}

container traffic-class {
    description "Configure IPv6 header's Traffic-class options";
    leaf value {
        type uint8;
        default 0;
        description "Set encapsulating outer IPv6 header's
                    Traffic-class field to specified value when
                    doing encapsulation";
    }
}

```

```
    leaf propagate {
      type boolean;
      default false;
      description "Propagate (or map) Traffic-class/CoS/PCP from
                  the incoming packet or L2 Ethernet frame being
                  encapsulated to the encapsulating IPv6 header's
                  Traffic-class field.";
    }
  }
}

grouping srv6-locator-state {
  description "SRv6 grouping Locator state";

  leaf operational-status {
    type srv6-types:srv6-status-type;
    config false;
    description "Indicates whether locator state is UP";
  }

  leaf is-in-address-conflict {
    type boolean;
    config false;
    description "Indicates whether locator address conflicts with
                some other IPv6 address on the box";
  }
}

grouping srv6-locators {
  description "SRv6 locator grouping";

  container locators {
    description "SRv6 locators";

    list locator {
      key "name";
      description "Configure a SRv6 locator";

      leaf name {
        type string;
        description "Locator name";
      }
    }
  }
}
```

```

}

leaf enable {
    type boolean;
    default false;

```

```

    description "Enable a SRv6 locator";
}

container prefix {
    description "Specify locator prefix value";
    leaf address {
        type inet:ipv6-address;
        mandatory true;
        description "IPv6 address";
    }
    leaf length {
        type srv6-types:srv6-locator-len;
        mandatory true;
        description "Locator (prefix) length";
    }
}

leaf algorithm {
    type uint32 {
        range "128..255";
    }

    description "Algorithm Id (for Flex-Algo)";
}

leaf anycast {
    type boolean;
    default false;
    description "Set to true if locator is an Anycast locator";
}

uses srv6-locator-state;
}
}
}

```

```

grouping srv6-stats-in {
  description "Grouping for inbound stats";

  leaf in-pkts {
    type yang:counter64;
    description
      "A cumulative counter of the total number of packets
      received";
  }

  leaf in-octets {
    type yang:counter64;
  }
}

```

```

    description
      "A cumulative counter of the total bytes received.";
  }
}

grouping srv6-stats-out {
  description "Grouping for inbound stats";

  leaf out-pkts {
    type yang:counter64;
    description
      "A cumulative counter of the total number of packets
      transmitted";
  }

  leaf out-octets {
    type yang:counter64;
    description
      "A cumulative counter of the total bytes transmitted.";
  }
}

grouping path-out-sids-choice {
  description "Grouping for Out-SID choices";
  choice encap-type {
    description "Out-SID encap-based choice";
    case srv6 {
      uses path-out-sids;
    }
  }
}

```

```

        case mpls {
            uses path-out-labels;
        }
    }
}

grouping local-sid-fwd-state {
    description "SRv6 local-SID forwarding state grouping";

    container forwarding {
        description "SRv6 local-SID forwarding state";

        leaf is-installed {
            type boolean;
            description "Indicates whether SID is installed in forwarding";
        }

        leaf next-hop-type {
            type srv6-types:srv6-nexthop-type;
        }
    }
}

```

```

        description "Forwarding next-hop types";
    }

    container paths {
        when "../is-installed = 'true'" {
            description "This container is valid only when the
                local-SID is installed in forwarding";
        }

        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }
        }

        container l2 {
            when "../../next-hop-type = 'l2'" {
                description "This container is valid only for L2 type
                    of NHs";
            }
        }
    }
}

```

```

    }

    leaf interface {
        type if:interface-ref;
        description "The outgoing Layer2 interface";
    }

    description "L2 information";
}

container l3 {
    when "../../../next-hop-type != 'l2'" {
        description "This container is valid only for L3 type
            of NHs";
    }

    leaf interface {
        type if:interface-ref;
        description "The outgoing Layer3 interface";
    }

    leaf next-hop {
        type inet:ip-address;
        description "The IP address of the next-hop";
    }
}

```

```

        uses path-attrs-cmn;

        description "L3 information";
    }
    uses path-out-sids-choice;
}

description "Forwarding paths";
}
}

grouping srv6-state-sid {
    description "SRv6 SID state grouping";
}

```

```

container local-sids {
  config false;
  description "Local-SID state";

  container counters {
    description "SRv6 counters";
    container cnt-3 {
      description "Counts SRv6 traffic received/dropped on local
        prefix not instantiated as local-SID";
      uses srv6-stats-in;
    }
  }
}

list local-sid {
  key "sid";
  description "Per-localSID Counters";

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

  uses srv6-locator;

  leaf is-reserved {
    type boolean;
    description "Set to true if SID comes from reserved pool";
  }

  leaf end-behavior-type {
    type identityref {
      base srv6-types:srv6-endpoint-type;
    }
  }
}

```

```

    description "Type of SRv6 end behavior.";
  }

  leaf alloc-type {
    type srv6-types:sid-alloc-type;
    description
      "Type of sid allocation.";
  }
}

```

```

list owner {
  key "type instance";
  description "SID Owner clients";
  leaf type {
    type identityref {
      base srv6-types:srv6-sid-owner-type;
    }
    description "SID owner/client type";
  }
  leaf instance {
    type string;
    description "Client instance";
  }
  leaf is-winner {
    type boolean;
    description "Is this client/owner the winning in terms of forwarding";
  }
}

uses local-sid-fwd-state;

container counters {
  description "SRv6 per local-SID counters";

  container cnt-1 {
    description "Counts SRv6 traffic received on local-SID prefix and processed successfully";
    uses srv6-stats-in;
  }
}
}
}

grouping srv6-support-ends {
  description "SRv6 End behavior support grouping";

  list end-behavior {

```

```
key "type";
```



```

description "End behavior support";

leaf type {
  type identityref {
    base srv6-types:srv6-endpoint-type;
  }
  description "End behavior (End*) type";
}
leaf supported {
  type boolean;
  mandatory true;
  description "True if supported";
}
}

grouping srv6-support-headends {
  description "SRv6 Headend behavior support grouping";

  list headend-behavior {
    key "type";
    description "Headend behavior support";
    leaf type {
      type identityref {
        base srv6-types:srv6-headend-type;
      }
      description "Headend behavior (H*) type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-msd-signaled {
  description "SRv6 MSD signaled parameter support grouping";

  container msd {
    description "SRv6 signaled MSD parameter support";

    leaf max-sl {
      type uint8;
      description "Maximum value of the SL field in the SRH of
        a received packet before applying the Endpoint behavior
        associated with a SID";
    }
  }
}

```

```
}
leaf max-end-pop {
  type uint8;
  description "Maximum number of SIDs in the top SRH in an
              SRH stack to which the router can apply
              PSP or USP flavors";
}
leaf max-h_encap {
  type uint8;
  description "Maximum number of SIDs that can be pushed as
              part of the H.Encaps* behavior";
}
leaf max-end_d {
  type uint8;
  description "Maximum number of SIDs in an SRH when applying
              End.D* behaviors (e.g. End.X6 and End.DT6)";
}
}
}

grouping srv6-support-security-rules {
  description "SRv6 Security rules grouping";

  list security-rule {
    key "type";
    description "Security rule support";

    leaf type {
      type identityref {
        base srv6-types:srv6-security-type;
      }
      description "Security rule type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

grouping srv6-support-counters {
  description "SRv6 Counters grouping";

  list counters {
    key "type";
```

description "SRv6 counter support";

```
leaf type {
  type identityref {
    base srv6-types:srv6-counter-type;
  }
  description "Counter type";
}
leaf supported {
  type boolean;
  mandatory true;
  description "True if supported";
}
}
}

grouping srv6-state-capabilities {
  description "SRv6 node capabilities grouping";
  container node-capabilities {
    config false;
    description "Node's SRv6 capabilities";

    uses srv6-support-ends;
    uses srv6-support-headends;
    uses srv6-msd-signaled;
    uses srv6-support-security-rules;
    uses srv6-support-counters;
  }
}

augment "/rt:routing/sr:segment-routing" {
  description
    "This augments Segment Routing (SR) with SRv6.";

  container srv6 {
    description "Segment Routing with IPv6 dataplane";

    /* config */
    leaf enable {
      type boolean;
      default false;
    }
  }
}
```

```

        description "Enable SRv6";
    }

    uses srv6-encap;
    uses srv6-locators;
    uses srv6-state-capabilities;
    uses srv6-state-sid;
}
}

```

```

/* Notifications */

```

```

grouping srv6-locator {
    description
        "An absolute reference to an SRv6 locator";
    leaf locator {
        type leafref {
            path "/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locat
        }
        description
            "Reference to a SRv6 locator.";
    }
}

notification srv6-locator-status-event {
    description
        "Notification event for a change of SRv6 locator operational
        status.";
    leaf operational-status {
        type srv6-types:srv6-status-type;
        description "Operational status";
    }
    uses srv6-locator;
}

notification srv6-sid-collision-event {
    description
        "Notification event for an SRv6 SID collision - i.e., attempt
        to bind an already bound SID to a new context";
    leaf sid {
        type srv6-types:srv6-sid;
        description "SRv6 SID";
    }
}

```

```

}
container existing {
  description "Current assignment / bind";
  leaf end-behavior-type {
    type identityref {
      base srv6-types:srv6-endpoint-type;
    }
    description "End type";
  }
  // TODO: More
}
container requested {
  description "Requested assignment / bind";

  leaf end-behavior-type {
    type identityref {

```

```

    base srv6-types:srv6-endpoint-type;
  }
  description "End type";
}
}
}
} // module

```

<CODE ENDS>

Figure 6: ietf-srv6-base.yang

### [5.3.](#) SRv6 Static

This YANG module imports types defined in [\[RFC6991\]](#), [\[RFC8343\]](#), and [\[RFC8349\]](#).

<CODE BEGINS> file "ietf-srv6-static@2022-01-14.yang"

```

// RFC Editor: replace the above date with the date of
// publication and remove this note.

```

```

module ietf-srv6-static {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-static";
  prefix srv6-static;

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management (NMDA
      version)";
  }

```

```

import ietf-segment-routing {
  prefix sr;
  reference "draft-ietf-spring-sr-yang";
}

import ietf-srv6-types {
  prefix srv6-types;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

import ietf-srv6-base {
  prefix srv6;
  reference "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}

```

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: Jaganbabu Rajamanickam  
<<mailto:jrajaman@cisco.com>>

Editor: Xufeng Liu  
<<mailto:xufeng.liu.ietf@gmail.com>>

Editor: Zhibo Hu  
<<mailto:huzhibo@huawei.com>>

Editor: Iftekhar Hussain  
<<mailto:IHussain@infinera.com>>

Editor: Himanshu Shah  
<<mailto:hshah@ciena.com>>

Editor: Daniel Voyer  
<<mailto:daniel.voyer@bell.ca>>

Editor: Hani Elmalky  
<<mailto:helmalky@google.com>>

Raza, et al.

Expires July 18, 2022

[Page 48]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

Editor: Satoru Matsushima  
<<mailto:satoru.matsushima@gmail.com>>

Editor: Katsuhiro Horiba  
<<mailto:katsuhiro.horiba@g.softbank.co.jp>>

Editor: Ahmed AbdelSalam  
<<mailto:ahabdel@cisco.com>>

";

description

"This YANG module defines the essential elements for the management of Static application for Segment-Routing with IPv6 dataplane (SRv6).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

```
reference "RFC XXXX";
// RFC Editor: replace XXXX with actual RFC number and remove
// this note
```

```
revision 2022-01-14 {
  description
    "Alignment with SRv6 network programming rev16";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}
```

```
revision 2019-10-30 {
  description
    "Extended model for EVPN behaviors";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
  // RFC Editor: replace XXXX with actual RFC number and remove
  // this note
}
```

```
revision 2019-07-08 {
```

```
description
  "Alignment with SRv6 network programming";
reference
  "RFC XXXX: YANG Data Model for SRv6";
```



```

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

revision 2018-10-22 {
    description
        "Initial revision.";
    reference
        "RFC XXXX: YANG Data Model for SRv6";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Config and State
 */

grouping path-attrs-v6 {
    description
        "IPv6 Path properties";

    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ipv6-address;
        description "The IP address of the next-hop";
    }

    leaf table {
        type srv6-types:table-id;
        description "The routing table associated with the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping path-attrs-v4 {
    description
        "IPv4 Path properties";

```

```
    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ipv4-address;
        description "The IP address of the next-hop";
    }

    leaf table {
        type srv6-types:table-id;
        description "The routing table associated with the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping path-attrs-mpls {
    description
        "MPLS Path properties";

    leaf interface {
        type if:interface-ref;
        description "The outgoing interface";
    }

    leaf next-hop {
        type inet:ip-address;
        description "The IP address of the next-hop";
    }

    uses srv6:path-attrs-cmn;
}

grouping multi-paths-v6 {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }
        }
    }
}
```

```
}
```

```
    uses path-attrs-v6;
    container sid-list {
        description "SID-list associated with the path";
        uses srv6:path-out-sids;
    }
}

}

grouping multi-paths-v4 {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            uses path-attrs-v4;
            container sid-list {
                description "SID-list associated with the path";
                uses srv6:path-out-sids;
            }
        }
    }
}

grouping multi-paths-mpls {
    description "Multipath grouping";

    container paths {
        description "List of outgoing paths";
        list path {
            key path-index;
            description "The list of paths associated with the SID";
```

```

leaf path-index {
    type uint8;
    description "Index of the path";
}

```

```

uses path-attrs-mpls;
container sid-list {

```

```

        description "SID-list associated with the path";
        uses srv6:path-out-sids;
    }
}
}
}

```

```

grouping multi-paths-v6-BUM {
    description
        "Multipath grouping for EVPN bridging BUM use case";

    container paths {
        description
            "List of outgoing paths for flooding";
        list path {
            key path-index;
            description "The list of paths associated with the SID";

            leaf path-index {
                type uint8;
                description "Index of the path";
            }

            leaf l2-interface {
                type if:interface-ref;
                description "The outgoing L2 interface for flooding";
            }
        }
    }
}

```

```

grouping srv6-sid-config {
    description

```

```

    "Configuration parameters relating to SRv6 sid.";

leaf function {
    type srv6-types:srv6-func-value;
    description
        "SRv6 function value.";
}
leaf end-behavior-type {
    type identityref {
        base srv6-types:srv6-endpoint-type;
    }
    mandatory true;
    description
        "Type of SRv6 end behavior.";
}

```

```

container end {
    when "../end-behavior-type = 'End'" {
        description
            "This container is valid only when the user chooses End
            behavior (variant: no PSP, no USP).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: no PSP, no USP)";
}

container end_psp {
    when "../end-behavior-type = 'End_PSP'" {
        description
            "This container is valid only when the user chooses End
            behavior (variant: PSP only).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID

```

```

        (variant: PSP only)";
    }

    container end_usp {
        when "../end-behavior-type = 'End_USP'" {
            description
                "This container is valid only when the user chooses End
                behavior (variant: USP only).";
        }
        description
            "The Endpoint function is the most basic function.
            FIB lookup on updated DA and forward accordingly
            to the matched entry.
            This is the SRv6 instantiation of a Prefix SID
            (variant: USP only)";
    }

    container end_psp_usp {
        when "../end-behavior-type = 'End_PSP_USP'" {
            description

```

```

        "This container is valid only when the user chooses End
        behavior (variant: PSP/USP).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: PSP/USP)";
    }

    container end_usd {
        when "../end-behavior-type = 'End_USD'" {
            description
                "This container is valid only when the user chooses End
                behavior (variant: USD only).";
        }
        description

```

```

    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: USD)";
}

container end_psp_usd {
    when "../end-behavior-type = 'End_PSP_USD'" {
        description
            "This container is valid only when the user chooses End
            behavior (variant: PSP/USD).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: PSP/USD)";
}

container end_usp_usd {
    when "../end-behavior-type = 'End_USP_USD'" {
        description
            "This container is valid only when the user chooses End
            behavior (variant: USP/USD).";
    }
    description

```

```

    "The Endpoint function is the most basic function.
    FIB lookup on updated DA and forward accordingly
    to the matched entry.
    This is the SRv6 instantiation of a Prefix SID
    (variant: USP/USD)";
}

container end_psp_usp_usd {
    when "../end-behavior-type = 'End_PSP_USP_IUSD'" {
        description
            "This container is valid only when the user chooses End

```

```

        behavior (variant: PSP/USP/USD).";
    }
    description
        "The Endpoint function is the most basic function.
        FIB lookup on updated DA and forward accordingly
        to the matched entry.
        This is the SRv6 instantiation of a Prefix SID
        (variant: PSP/USP/USD)";
}

container end-t {
    when "../end-behavior-type = 'End.T'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: no PSP, no USP).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant: no PSP,
        no USP).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";
    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-t_psp {

```

```

    when "../end-behavior-type = 'End.T_PSP'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: PSP only).";
    }

```



```

description
  "Endpoint with specific IPv6 table lookup (variant: PSP only).
  Lookup the next segment in IPv6 table T
  associated with the SID and forward via
  the matched table entry.

  The End.T is used for multi-table operation
  in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

container end-t_usp {
  when "../end-behavior-type = 'End.T_USP'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USP only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USP only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

```

```

container end-t_psp_usp {
  when "../end-behavior-type = 'End.T_PSP_USP'" {
    description
      "This container is valid only when the user chooses
        End.T behavior (variant: USP/PSP).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USP/PSP).
      Lookup the next segment in IPv6 table T
      associated with the SID and forward via
      the matched table entry.
      The End.T is used for multi-table operation
      in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

container end-t_usd {
  when "../end-behavior-type = 'End.T_USD'" {
    description
      "This container is valid only when the user chooses
        End.T behavior (variant: USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: USD only).
      Lookup the next segment in IPv6 table T
      associated with the SID and forward via
      the matched table entry.
      The End.T is used for multi-table operation
      in the core.";

  // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

```

```
container end-t_psp_usd {
  when "../end-behavior-type = 'End.T_PSP_USD'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: PSP/USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant: PSP/USD
    only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

    // TODO presence "Mandatory child only if container is present";

  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
      "Table Id for lookup on updated DA (next segment)";
  }
}

container end-t_usp_usd {
  when "../end-behavior-type = 'End.T_USP_USD'" {
    description
      "This container is valid only when the user chooses
      End.T behavior (variant: USP/USD only).";
  }
  description
    "Endpoint with specific IPv6 table lookup (variant:
    USP/USD only).
    Lookup the next segment in IPv6 table T
    associated with the SID and forward via
    the matched table entry.
    The End.T is used for multi-table operation
    in the core.";

    // TODO presence "Mandatory child only if container is present";
```

```

leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description
        "Table Id for lookup on updated DA (next segment)";
}

```

```

}

container end-t_psp_usp_usd {
    when "../end-behavior-type = 'End.T_PSP_USP_USD'" {
        description
            "This container is valid only when the user chooses
            End.T behavior (variant: USP only).";
    }
    description
        "Endpoint with specific IPv6 table lookup (variant:
        PSP/USP/USD only).
        Lookup the next segment in IPv6 table T
        associated with the SID and forward via
        the matched table entry.
        The End.T is used for multi-table operation
        in the core.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description
            "Table Id for lookup on updated DA (next segment)";
    }
}

container end-x {
    when "../end-behavior-type = 'End.X'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: no USP/PSP)";
    }
    description
        "Endpoint with cross-connect to an array of

```

```

        layer-3 adjacencies (variant: no USP/PSP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

```

```

container end-x_psp {
    when "../end-behavior-type = 'End.X_PSP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP only).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_osp {
    when "../end-behavior-type = 'End.X_USP'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USP only)";
    }
}

```

```

description
  "Endpoint with cross-connect to an array of
    layer-3 adjacencies (variant: USP only).
    Forward to layer-3 adjacency bound to the SID S.
    The End.X function is required to express any
    traffic-engineering policy.";

leaf protected {
  type boolean;
  default false;
  description "Is Adj-SID protected?";
}

uses multi-paths-v6;
}

container end-x_psp_usp {
  when "../end-behavior-type = 'End.X_PSP_USP'" {
    description
      "This container is valid only when the user chooses

```

```

      End.X behavior (variant: PSP/USP)";
    }
  description
    "Endpoint with cross-connect to an array of
      layer-3 adjacencies (variant: PSP/USP).
      Forward to layer-3 adjacency bound to the SID S.
      The End.X function is required to express any
      traffic-engineering policy.";

  leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
  }

  uses multi-paths-v6;
}

container end-x_usd {
  when "../end-behavior-type = 'End.X_USD'" {

```

```

        description
            "This container is valid only when the user chooses
            End.X behavior (variant: USD only)";
    }
    description
        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

container end-x_psp_usd {
    when "../end-behavior-type = 'End.X_PSP_USD'" {
        description
            "This container is valid only when the user chooses
            End.X behavior (variant: PSP/USD only)";
    }
    description

```

```

        "Endpoint with cross-connect to an array of
        layer-3 adjacencies (variant: PSP/USP).
        Forward to layer-3 adjacency bound to the SID S.
        The End.X function is required to express any
        traffic-engineering policy.";

    leaf protected {
        type boolean;
        default false;
        description "Is Adj-SID protected?";
    }

    uses multi-paths-v6;
}

```

```

container end-x_osp_osp {
  when "../end-behavior-type = 'End.X_USP_USD'" {
    description
      "This container is valid only when the user chooses
        End.X behavior (variant: USP/USD only)";
  }
  description
    "Endpoint with cross-connect to an array of
      layer-3 adjacencies (variant: PSP/USP).
      Forward to layer-3 adjacency bound to the SID S.
      The End.X function is required to express any
      traffic-engineering policy.";

  leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
  }

  uses multi-paths-v6;
}

```

```

container end-x_osp_osp {
  when "../end-behavior-type = 'End.X_PSP_USP_USD'" {
    description
      "This container is valid only when the user chooses
        End.X behavior (variant: PSP/USP/USD only)";
  }
  description
    "Endpoint with cross-connect to an array of
      layer-3 adjacencies (variant: PSP/USP).
      Forward to layer-3 adjacency bound to the SID S.
      The End.X function is required to express any

```

```

      traffic-engineering policy.";

  leaf protected {
    type boolean;
    default false;
    description "Is Adj-SID protected?";
  }

```



```

    uses multi-paths-v6;
}

container end-b6-encaps {
    when "../end-behavior-type = 'End.B6.Encaps' or
        ../end-behavior-type = 'End.B6.Encaps.Red'" {
        description
            "This container is valid only when the user chooses
            End.B6.Encaps or End.B6.Encaps.Red behavior.";
    }
    description
        "Endpoint bound to an SRv6 Policy.
        Insert SRH based on the policy and forward the
        packet toward the first hop configured in the policy.
        This is the SRv6 instantiation of a Binding SID.
        This behavior also adds an outer IPv6 header";

    // TODO presence "Mandatory child only if container is present";

    leaf policy-name {
        type string;
        mandatory true;
        description "SRv6 policy name.";
    }
    leaf source-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "IPv6 source address for Encap.";
    }
}

uses multi-paths-v6;
}

container end-bm {
    when "../end-behavior-type = 'End.BM'" {
        description
            "This container is valid only when the user chooses
            End.BM behavior.";
    }
}

```

```

description
  "Endpoint bound to an SR-MPLS Policy.
   push an MPLS label stack <L1, L2, L3> on the
   received packet and forward the according to
   Lable L1.
   This is an SRv6 instantiation of an SR-MPLS Binding SID.";

// TODO presence  "Mandatory child only if container is present";

leaf policy-name {
  type string;
  mandatory true;
  description "SRv6 policy name";
}
uses multi-paths-mpls;
}

container end-dx6 {
  when "../end-behavior-type = 'End.DX6'" {
    description
      "This container is valid only when the user chooses
       End.DX6 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
     an array of IPv6 adjacencies. Pop the (outer)
     IPv6 header and its extension headers and forward
     to layer-3 adjacency bound to the SID S.
     The End.DX6 used in the L3VPN use-case.";

  uses multi-paths-v6;
  // TODO: Backup path of type "Lookup in table"
}

container end-dx4 {
  when "../end-behavior-type = 'End.DX4'" {
    description
      "This container is valid only when the user chooses
       End.DX4 behavior.";
  }
  description
    "Endpoint with decapsulation and cross-connect to
     an array of IPv4 adjacencies.
     Pop the (outer) IPv6 header and its extension
     header and forward to layer-3 adjacency bound
     to the SID S.
     This would be equivalent to the per-CE VPN
     label in MPLS.";
}

```

Internet-Draft

YANG Data Model for SRv6

January 2022

```
    uses multi-paths-v4;
    // TODO: Backup path of type "Lookup in table"
}
container end-dt6 {
    when "../end-behavior-type = 'End.DT6'" {
        description
            "This container is valid only when the user chooses
            End.DT6 behavior.";
    }
    description
        "Endpoint with decapsulation and specific IPv6 table
        lookup.
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv6 DA in IPv6
        table T and forward via the matched table entry.
        End.DT6 function is used in L3VPN use-case.";

    // TODO presence "Mandatory child only if container is present";

    leaf lookup-table-ipv6 {
        type srv6-types:table-id;
        mandatory true;
        description "IPv6 table";
    }
}
container end-dt4 {
    when "../end-behavior-type = 'End.DT4'" {
        description
            "This container is valid only when the user chooses
            End.DT4 behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        IPv4 table lookup.
        Pop the (outer) IPv6 header and its extension
        headers.
        Lookup the exposed inner IPv4 DA in IPv4
        table T and forward via the matched table entry.
        This would be equivalent to the per-VRF VPN label
        in MPLS.";

    // TODO presence "Mandatory child only if container is present";
```

```

leaf lookup-table-ipv4 {
  type srv6-types:table-id;
  mandatory true;
  description "IPv4 table";

```

```

}
}
container end-dt46 {
  when "../end-behavior-type = 'End.DT46'" {
    description
      "This container is valid only when the user chooses
        End.DT46 behavior.";
  }
  description
    "Endpoint with decapsulation and specific
      IP table lookup.
      Depending on the protocol type (IPv4 or IPv6)
      of the inner ip packet and the specific VRF name
      forward the packet.
      This would be equivalent to the per-VRF VPN
      label in MPLS.";

  // TODO presence  "Mandatory child only if container is present";

  leaf lookup-table-ipv4 {
    type srv6-types:table-id;
    mandatory true;
    description "IPv4 table";
  }
  leaf lookup-table-ipv6 {
    type srv6-types:table-id;
    mandatory true;
    description "IPv6 table";
  }
}

/* EVPN END behavior types */
container end-dx2 {
  when "../end-behavior-type = 'End.DX2'" {
    description
      "This container is valid only when the user chooses

```

```

        End.DX2 behavior.";
    }
    description
        "This is an Endpoint with decapsulation and Layer-2
        cross-connect to OIF.
        Pop the (outer) IPv6 header and its extension headers.
        Forward the resulting frame via OIF associated to the SID.
        The End.DX2 function is the L2VPN/EVPN VPWS use-case.";

    container path {
        description "Outgoing path";
        leaf l2-interface {

```

```

        type if:interface-ref;
        mandatory true;
        description "Outgoing L2 interface";
    }
}

container end-dx2v {
    when "../end-behavior-type = 'End.DX2V'" {
        description
            "This container is valid only when the user chooses
            End.DX2V behavior.";
    }
    description
        "Endpoint with decapsulation and specific VLAN
        L2 table lookup.
        Pop the (outer) IPv6 header and its extension headers.
        Lookup the exposed inner VLANs in L2 table T.
        Forward via the matched table entry.
        The End.DX2V is used for EVPN Flexible cross-connect
        use-cases";

    leaf lookup-table-vlan {
        type srv6-types:table-id;
        mandatory true;
        description
            "VLAN lookup table. There could be multiple
            vlan demux tables on the node, where a DX2V SID
            points to one vlan table";
    }
}

```

```

    }
}

container end-dt2u {
    when "../end-behavior-type = 'End.DT2U'" {
        description
            "This container is valid only when the user chooses
            End.DT2U behavior.";
    }
    description
        "Endpoint with decapsulation and specific
        unicast L2 MAC table lookup.
        Pop the (outer) IPv6 header and its extension headers.
        Learn the exposed inner MAC SA in L2 MAC table T.
        Lookup the exposed inner MAC DA in L2 MAC table T.
        Forward via the matched T entry else to all L2OIF in T.
        The End.DT2U is used for EVPN Bridging unicast use cases";

    leaf lookup-table-mac {

```

```

        type srv6-types:table-id;
        mandatory true;
        description "MAC L2 lookup table";
    }
}

container end-dt2m {
    when "../end-behavior-type = 'End.DT2M'" {
        description
            "This container is valid only when the user chooses
            End.DT2M behavior.";
    }
    description
        "Endpoint with decapsulation and specific flooding table.
        Pop the (outer) IPv6 header and its extension headers.
        Learn the exposed inner MAC SA in L2 MAC table T.
        Forward on all L2OIF (in the flooding table) excluding the one
        identified by Arg.FE2.
        The End.DT2M is used for EVPN Bridging BUM use case with
        ESI (Split Horizon) filtering capability.";

    leaf flooding-table {

```

```

    type srv6-types:table-id;
    mandatory true;
    description "L2 Flooding table (list of OIFs)";
}

uses multi-paths-v6-BUM;

/* TODO - Support for argument Arg.FE2. It is an argument specific
   to EVPN ESI filtering and EVPN-ETREE used to exclude specific
   OIF (or set of OIFs) from flooding table. */
}

/* End of EVPN END behavior types */
}

grouping srv6-static-cfg {
    description
        "Grouping configuration and operation for SRv6 sid.";

    list sid {
        key "function";
        description "List of locally instantiated SIDs";

        uses srv6-sid-config;
    }
}

```

```

augment "/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locator"
    description
        "This augments locator leaf within SRv6.";

    container static {
        description "Static SRv6";

        /* Local SIDs */
        container local-sids {
            description
                "SRv6-static locally instantiated SIDs";

            uses srv6-static-cfg;
            /* no state for now; SID state accessible through base model */
        }
    }

```

```

    }
  }
} // module

```

<CODE ENDS>

Figure 7: ietf-srv6-static.yang

## 6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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 all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

It goes without saying that this specification also inherits the security considerations captured in the SRv6 specification document [RFC8986].



## 7. 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-srv6-types	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-srv6-base	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-srv6-static	The IESG	N/A

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [[RFC6020](#)]:

Name	Namespace	Prefix	Reference
ietf-srv6-types	urn:ietf:params:xml:ns:yang:ietf-srv6-types	srv6-types	This document
ietf-srv6-base	urn:ietf:params:xml:ns:yang:ietf-srv6-base	srv6	This document
ietf-srv6-static	urn:ietf:params:xml:ns:yang:ietf-srv6-static	srv6-static	This document

-- RFC Editor: Replace "This document" with the document RFC number at time of publication, and remove this note.

## 8. Acknowledgments

The authors would like to acknowledge Darren Dukes, Les Ginsberg, Ahmed Bashandy, Rajesh Venkateswaran, and Mike Mallin for their review of some of the contents in this draft.

## 9. References

## 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.
- [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", [BCP 216](#), [RFC 8407](#), DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", [RFC 8446](#), DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", [RFC 8986](#), DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.
- [RFC9020] Litkowski, S., Qu, Y., Lindem, A., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", [RFC 9020](#), DOI 10.17487/RFC9020, May 2021, <<https://www.rfc-editor.org/info/rfc9020>>.

## [9.2.](#) Informative References

[I-D.ietf-dmm-srv6-mobile-uplane]

Matsushima, S., Filsfils, C., Kohno, M., Garvia, P. C., Voyer, D., and C. E. Perkins, "Segment Routing IPv6 for Mobile User Plane", [draft-ietf-dmm-srv6-mobile-uplane-17](#) (work in progress), October 2021.

[I-D.ietf-spring-sr-service-programming]

Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", [draft-ietf-spring-sr-service-programming-05](#) (work in progress), September 2021.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", [RFC 8754](#), DOI 10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

## Authors' Addresses

Kamran Raza  
Cisco Systems  
Email: [skraza@cisco.com](mailto:skraza@cisco.com)

Sonal Agarwal  
Cisco Systems  
Email: [agarwaso@cisco.com](mailto:agarwaso@cisco.com)

Xufeng Liu  
Volta Networks  
Email: [xufeng.liu.ietf@gmail.com](mailto:xufeng.liu.ietf@gmail.com)

Zhibo Hu  
Huawei Technologies

Email: huzhibo@huawei.com

Raza, et al.

Expires July 18, 2022

[Page 74]

---

Internet-Draft

YANG Data Model for SRv6

January 2022

Iftekhar Hussain  
Infinera Corporation  
Email: IHussain@infinera.com

Himanshu Shah  
Ciena Corporation  
Email: hshah@ciena.com

Daniel Voyer  
Bell Canada  
Email: daniel.voyer@bell.ca

Hani Elmalky  
Individual  
Email: helmalky@google.com

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

Katsuhiro Horiba  
SoftBank  
Email: katsuhiro.horiba@g.softbank.co.jp

Jaganbabu Rajamanickam  
Cisco Systems  
Email: jrajaman@cisco.com

Ahmed AbdelSalam

Cisco Systems  
Email: ahabdels@cisco.com

Raza, et al.

Expires July 18, 2022

[Page 75]