YANG Data Model for SRv6
draft-raza-spring-srv6-yang-00.txt

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.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 16, 2018.
Copyright Notice

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

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

Table of Contents

1. Introduction ................................................. 3
2. Specification of Requirements ................................. 3
3. YANG Model ................................................... 4
   3.1. Overview ............................................... 4
   3.2. SRv6 Types ............................................. 4
   3.3. SRv6 Base ............................................. 5
       3.3.1. Configuration .................................... 5
       3.3.2. State ........................................... 6
       3.3.3. Notification .................................... 8
   3.4. SRv6 Static ........................................... 9
       3.4.1. Configuration .................................... 9
       3.4.2. State ........................................... 12
       3.4.3. Notification .................................... 12
4. Work In Progress ............................................ 13
5. Pending Items ................................................. 13
6. YANG Specification ........................................... 13
   6.1. SRv6 Types ............................................. 13
   6.2. SRv6 Base ............................................. 21
   6.3. SRv6 Static ........................................... 34
7. Security Considerations ..................................... 46
8. IANA Considerations ......................................... 47
9. Acknowledgments ............................................. 47
10. References ................................................ 47
    10.1. Normative References ................................. 47
    10.2. Informative References .............................. 48
Authors’ Addresses ............................................ 49
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 [I-D.ietf-spring-segment-routing], 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 [I-D.ietf-6man-segment-routing-header]. The various functions and behaviors corresponding to network programming using SRv6 are specified in [I-D.filsfils-spring-srv6-network-programming].

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

To illustrate basic behaviors as captured in [I-D.filsfils-spring-srv6-network-programming], 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:

- Configuration
- Operational State
- 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 [RFC2119].
3. YANG Model

3.1. Overview

This document defines the following new YANG modules:

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

The modeling in this document complies with the Network Management datastore architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. 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.

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:

- srv6-sid: SRv6 SID
- srv6-func-opcode: Typedef for FUNC opcode in an SRv6 SID
- srv6-func-opcode-reserved: Typedef for "reserved" FUNC opcode
- srv6-func-opcode-unreserved: Typedef for "unreserved" (allocatable) FUNC opcode
- srv6-func-opcode-reserved-type: Enum (list) of "reserved" FUNC opcode
- srv6-end-type: SRv6 End behavior types [I-D.filsfils-spring-srv6-network-programming]
- srv6-transit-type: SRv6 Transit behavior types [I-D.filsfils-spring-srv6-network-programming]

The associated YANG specification for this module is captured in Section 6.1.
3.3. SRv6 Base

The base SRv6 model is captured by ietf-srv6-base module. This module augments "/rt:routing" and specifies the configuration, operational state, and notification events required to manage base SRv6.

The associated YANG specification for this module is captured in Section 6.2.

3.3.1. Configuration

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

- SRv6 Enablement: Enable Segment-Routing SRv6 feature
- Encapsulation Parameters: Provide encapsulation related parameters, such as source-address and ip-ttl-propagation, to be used when performing T.Encap*
- Locator(s) Specification: SRv6 locator is a fundamental construct for an SRv6 network. This is the construct from which SID (function opcodes) 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. It is possible to have more than one locator per node. In case of more than one locator, there is one and only one locator designated as the default locator.

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

- node capabilities: provides information on the node (hardware) capabilities and support regarding various SRv6 aspects and features including end behaviors, transit behaviors, security rules, counter/stats support, and other parameters that need to be signaled in the SRv6 network by protocols.

- 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.

- local-sid: provides information related to local-SIDs allocated and/or installed on the node. This includes two types of information (i) aggregate across all local-SIDs (such as aggregate counters), (ii) 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 representation of the data model for the SRv6 operational state:

```
module: ietf-srv6-base
  augment /rt: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         srv6-types:srv6-end-type
        |     +--ro supported    boolean
        | +--ro transit-behavior* [type]
        |     +--ro type         srv6-types:srv6-transit-type
        |     +--ro supported    boolean
        | +--ro signaled-parameters
        |     +--ro max-sl?            uint8
        |     +--ro max-end-pop-srh?   uint8
        |     +--ro max-t_insert?      uint8
        |     +--ro max-t_encap?       uint8
        |     +--ro max-end_d?         uint8
        | +--ro security-rule* [type]
        |     +--ro type         srv6-types:srv6-security-rule-type
        |     +--ro supported    boolean
        | +--ro counters* [type]
        |     +--ro type         srv6-types:srv6-counter-type
        |     +--ro supported    boolean
      +--ro local-sids
        | +--ro counters
        |     +--ro in-pkts?     yang:counter64
        |     +--ro in-octets?   yang:counter64
        | +--ro local-sid* [sid]
        |     +--ro sid                  srv6-types:srv6-sid
        |     +--ro is-reserved?   boolean
        |     +--ro end-behavior-type? srv6-types:srv6-end-type
        |     +--ro alloc-type?    srv6-types:sid-alloc-type
        |     +--ro owner* [type instance]
        |     | +--ro type         srv6-types:srv6-sid-owner-type
        |     | +--ro instance     string
        |     | +--ro is-winner?   boolean
        |     +--ro forwarding
        |           +--ro is-installed?   boolean
```

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:

- locator operational state changes
- local-SID collision event

Following is a simplified graphical representation of the data model for SRv6 notifications:
module: ietf-srv6-base

notifications:
  +---n srv6-locator-status-event
    |  +--ro operational-status?  srv6-types:srv6-status-type
  +---n srv6-sid-collision-event
    +--ro sid?         srv6-types:srv6-sid
    +--ro existing
    |  +--ro end-behavior-type?  srv6-types:srv6-end-type
    +--ro requested
    |  +--ro end-behavior-type?  srv6-types:srv6-end-type

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 6.3.

3.4.1. Configuration

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

Following are salient features of SRv6-Static config model:

- Allows static (explicit) configuration for local-SIDs under a given locator
- Given that entry is scoped under a locator, the key for each entry is function "opcode"
- A user must also specify end-behavior type (End* function) associated with the entry
- 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.
Each local-SID has zero or more forwarding paths specified.

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.

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.

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/DX6, End.B6 etc.).

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

```yang
module: ietf-srv6-static
    augment /rt:routing/srv6:srv6/srv6:locators/srv6:locator:
        +--rw static
            +--rw local-sids
                +--rw sid* [opcode]
                    +--rw opcode              srv6-types:srv6-func-opcode-unreserved
                    +--rw end-behavior-type    srv6-types:srv6-end-type
                    +--rw end
                        |  +--rw usp?   boolean
                        +--rw end-t
                            |  +--rw usp?                 boolean
                            |  +--rw lookup-table-ipv6    srv6-types:table-id
                            +--rw end-x
                                |  +--rw usp?     boolean
                                |  +--rw paths
                                    +--rw path* [path-index]
                                        |  +--rw path-index          uint8
                                        +--rw interface?           if:interface-ref
                                        +--rw next-hop?            inet:ipv6-address
                                        +--rw weight?              uint16
                                        +--rw role?                enumeration
                                        +--rw backup-path-index?   uint8
                                        +--rw encap
                                            |  +--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 weight?               uint16           |
|       |       |       | +++--rw role?                 enumeration     |
|       |       |       | +++--rw backup-path-index?    uint8           |
|       |       |       | +++--rw encap                 |
|       |       |       |       | +++--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 weight?               uint16           |
|       |       |       | +++--rw role?                 enumeration     |
|       |       |       | +++--rw backup-path-index?    uint8           |
|       |       |       | +++--rw encap                 |
|       |       |       |       | +++--rw out-sid* [sid]        |
|       |       |       |       |       | +++--rw sid  srv6-types:srv6-sid |
|       | ++--rw end-dt4                  |
|       | ++--rw lookup-table-ipv4        srv6-types:table-id |
|       | ++--rw end-dt6                  |
|       | ++--rw lookup-table-ipv6        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-b6                   |
|       |       | ++--rw policy-name             string       |
|       |       | ++--rw paths                   |
|       |       |       | ++--rw path* [path-index]     |
|       |       |       |       | +++--rw path-index            uint8     |
|       |       |       |       | +++--rw interface?            if:interface-ref |
|       |       |       |       | +++--rw next-hop?             inet:ipv6-address |
|       |       |       |       | +++--rw weight?               uint16           |
|       |       |       |       | +++--rw role?                 enumeration     |
|       |       |       |       | +++--rw backup-path-index?    uint8           |
|       |       |       |       | +++--rw encap                 |
|       |       |       |       |       | +++--rw out-sid* [sid]        |
|       |       |       |       |       |       | +++--rw sid  srv6-types:srv6-sid |
|       | ++--rw end-b6-encaps            |
|       |       | ++--rw policy-name             string       |
|       |       |       | +++--rw source-address         inet:ipv6-address |

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. Work In Progress

The YANG modeling for the corresponding SRv6 protocols and technologies is in progress as follows:

- SRv6 ISIS: Transport SIDs, TILFA, and Microloop Avoidance [I-D.bashandy-isis-srv6-extensions]
- SRv6 BGP: L3VPN and Global Table [I-D.dawra-idr-srv6-vpn]
- SRv6 TE: SR Policy [I-D.filips-spring-segment-routing-policy]
- SRv6-based EVPN [I-D.dawra-idr-srv6-vpn]
- SRv6-based Service Chaining [I-D.clad-spring-segment-routing-service-chaining]

The corresponding models will be specified later either in this or in a separate document.

5. Pending Items

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

- Align SRv6 base with SR (MPLS) model [I-D.ietf-spring-sr-yang].
- Extend local-SID collision event/notification in SRv6-base model.
- Add RPC support in the SRv6-base model.
- Add EVPN End functions in the SRv6-Static model.
- Add Service Chaining End functions in the SRv6-Static model.
- Add ARGS support in the SRv6-Static model.
- QoS support

6. YANG Specification

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

6.1. SRv6 Types

<CODE BEGINS> file "ietf-srv6-types@2017-11-12.yang" -->
module ietf-srv6-types {
    namespace "urn:ietf:params:xml:ns:yang:ietf-srv6-types";
    prefix srv6-types;

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

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

WG Chair: Bruno Decraene
<mailto:loa@pi.nu>

WG Chair: Martin Vigoureux
<mailto:rcallon@juniper.net>

Editor: Kamran Raza
<mailto:skraza@cisco.com>

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

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.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:hani.elmalky@ericsson.com>

Editor: Ahmed AbdelSalam
<mailto:ahmed.abdelsalam@gssi.it>

";

description
"This YANG module defines the essential types 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).

reference "RFC XXXX";

revision 2017-11-12 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for SRv6";
}

// TODO: Special opcode

typedef srv6-end-type {

type enumeration {
  /* draft-filsfils-spring-srv6-network-programming */

    enum End {
      value 1;
      description
        "The Endpoint function. This is the most basic function";
    }

    enum End.X {
      description
        "Endpoint with cross-connect to an array of layer-3 adjacencies";
    }

    enum End.T {
      description
        "Endpoint with specific IPv6 table lookup";
    }

    enum End.DX2 {
      description
        "Endpoint with decapsulation and Layer-2 cross-connect to an L2 interface";
    }
enum End.DX2V {
    description
    "Endpoint with decapsulation and specific
    VLAN L2 table lookup";
}
enum End.DT2U {
    description
    "Endpoint with decapsulation and specific
    unicast MAC L2 table lookup";
}
enum End.DT2M {
    description
    "Endpoint with decapsulation and specific L2 table
    flooding";
}
enum End.DX6 {
    description
    "Endpoint with decapsulation and cross-connect
to an array of IPv6 adjacencies";
}
enum End.DX4 {
    description
    "Endpoint with decapsulation and cross-connect
to an array of IPv4 adjacencies";
}
enum End.DT6 {
    description
    "Endpoint with decapsulation and specific
IPv6 table lookup";
}
enum End.DT4 {
    description
    "Endpoint with decapsulation and specific
IPv4 table lookup";
}
enum End.DT46 {
    description
    "Endpoint with decapsulation and specific IP
(IPv4 or IPv6) table lookup";
}
enum End.B6 {
    description
    "Endpoint bound to an SRv6 Policy";
}
enum End.B6.Encaps {
    description
    "This is a variation of the End.B6 behavior
where the SRv6 Policy also includes an IPv6 Source Address A."
}
enum End.BM {
  description
  "Endpoint bound to an SR-MPLS Policy";
}

enum End.S {
  description
  "Endpoint in search of a target in table TE";
}

/* draft-clad-spring-segment-routing-service-chaining */
enum End.AS {
  description
  "Service-Chaining Static proxy for inner type (Ethernet, IPv4 or IPv6)";
}
enum End.AD {
  description
  "Service-Chaining Dynamic proxy for inner type (Ethernet, IPv4 or IPv6)";
}
enum End.ASM {
  description
  "Service-Chaining Shared memory SR proxy for inner type (Ethernet, IPv4 or IPv6)";
}
enum End.AM {
  description
  "Service-Chaining Masquerading SR proxy";
}

description "SRv6 End behavior types";
}

typedef srv6-transit-type {
  type enumeration {
    /* draft-filsfils-spring-srv6-network-programming */
    enum T { value 1; description "Transit"; }
    enum T.Encaps { description "T.Encaps"; }
    enum T.Insert { description "T.Insert"; }
    enum T.Encaps.L2 { description "T.Encaps.L2"; }
  }

description "SRv6 Transit behavior types";
}

typedef srv6-security-rule-type {
type enumeration {
    /* draft-filsfils-spring-srv6-network-programming */
    enum SEC1 { value 1; description "Security rule SEC1"; }
    enum SEC2 { description "Security rule SEC2"; }
    enum SEC3 { description "Security rule SEC3"; }
    enum SEC4 { description "Security rule SEC4"; }
}

description "SRv6 Security rule types";
}

typedef srv6-counter-type {
    type enumeration {
        /* draft-filsfils-spring-srv6-network-programming */
        enum CNT0 { value 0; description "CNT0"; }
        enum CNT1 { description "CNT1"; }
        enum CNT2 { description "CNT2"; }
    }

description "SRv6 counter types";
}

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

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

typedef srv6-func-opcode-reserved {
    type uint32 {
        range "1 .. 63";
    }

description "This is a typedef for SID FUNC’s reserved opcode type";
}

typedef srv6-func-opcode-unreserved {
    type uint32 {
        range "64 .. max";
    }
}
typedef srv6-func-opcode-reserved-type {
  type enumeration {
    enum invalid {  value 0; description "Invalid opcode"; }
    enum default-end-psp {  value 1;
      description "Opcode for Default End/PSP funcion"; }
    enum default-end-usp {  value 2;
      description "Opcode for Default End/USP funcion"; }
    // TODO enum wildcard { value -0; description "Opcode for Wildcard/mass operations"; }
  }
  description "SRv6 SID FUNC Reserved Opcodes";
}

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 "33 .. 128";
  }
  default 128;
  description "This type defines a SID prefixlen with range constraints";
}

typedef sid-alloc-type {
  type enumeration {
    enum Dynamic {
      description "SID allocated dynamically.";
    }
    enum Explicit {
      description "SID allocated with explicit (static) value";
    }
  }
  description "Types of sid allocation used.";
}
typedef srv6-sid-owner-type {
  type enumeration {
    enum isis { value 1; description "ISIS"; }
    enum ospfv3 { description "OSPFv3"; }
    enum bgp { description "BGP"; }
    enum evpn { description "EVPN"; }
    enum sr-policy { description "SR Policy"; }
    enum service-chain { description "Service-Chaining"; }
    // TODO: Others
  }
  description
  "SID Client types.";
}

// TODO: Rtg module?
typedef table-id {
  type uint32;
  description
  "Routing 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
6.2. SRv6 Base

<CODE BEGINS> file "ietf-srv6-base@2017-11-12.yang" -->

module ietf-srv6-base {

  prefix srv6;

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix inet;
  }

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

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

  import ietf-routing {
    prefix "rt"
  }

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

  organization
    "IETF SPRING Working Group";

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

    WG Chair: Bruno Decraene
     <mailto:loa@pi.nu>

    WG Chair: Martin Vigoureux
     <mailto:rcallon@juniper.net>

    Editor: Kamran Raza
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).

reference "RFC XXXX";

revision 2017-11-12 {
    description
    "Initial revision.";
    reference
    "RFC XXXX: YANG Data Model for SRv6";
}

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

    leaf weight {
        type uint16;
        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";
  }
}

Configurerencapsulation {
  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";
  }
  leaf ip-ttl-propagation {
    type boolean;
    default false;
    description "IP TTL propagation from encapsulated packet to encapsulating outer IPv6 header. When configured on decapsulation side, this refers to propagating IP TTL from outer IPv6 header to inner header after decap";
  }
}

Groupingsrv6-locator-state {
  description "SRv6 grouping Locastateor ";

  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";
            }

            leaf is-default {
                type boolean;
                mandatory true;
                description "Indicates if the locator is a default 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";
                }
            }

            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 cnt0 {
        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-ref;
leaf is-reserved {
  type boolean;
  description "Set to true if SID comes from reserved pool";
}

leaf end-behavior-type {
  type srv6-types:srv6-end-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 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 cnt1 {
    description "Counts SRv6 traffic received on local-SID prefix and processed successfully";
    uses srv6-stats-in;
  }
}

}
description "SRv6 End behavior support grouping";

list end-behavior {
  key "type";
  description "End behavior support";

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

grouping srv6-support-transits {
  description "SRv6 Transit behavior support grouping";

  list transit-behavior {
    key "type";
    description "Transit behavior support";

    leaf type {
      type srv6-types:srv6-transit-type;
      description "Transit behavior (T*) type";
    }
    leaf supported {
      type boolean;
      mandatory true;
      description "True if supported";
    }
  }
}

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

  container signaled-parameters {
    description "SRv6 signaled parameter support";

    leaf max-sl {
      type uint8;
      //mandatory true;
      description "Maximum value of the SL field in the SRH of a received packet before applying the function...";
    }
  }
}
associated with a SID;
}
leaf max-end-pop-srh {
    type uint8;
    //mandatory true;
    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-t_insert {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs that can be inserted as part of the T.insert behavior";
}
leaf max-t_encap {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs that can be inserted as part of the T.Encap behavior";
}
leaf max-end_d {
    type uint8;
    //mandatory true;
    description "Maximum number of SIDs in an SRH when applying End.DX6 and End.DT6 functions";
}
}
}
grouping srv6-support-security-rules {
    description "SRv6 Security rules grouping";
    list security-rule {
        key "type";
        description "Security rule support";
        leaf type {
            type srv6-types:srv6-security-rule-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 srv6-types:srv6-counter-type;
            description "Counter type";
        }

        leaf supported {
            type boolean;
            mandatory true;
            description "True if supported";
        }
    }
}

augment "/rt:routing" {
    description "This augments routing-instance configuration with segment-routing SRv6.";

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

        /* config */
        leaf enable {
            type boolean;
            default false;
            description "Enable SRv6";
        }

        uses srv6-encap;
    }
}

Raza, et al.  Expires May 16, 2018
uses srv6-locators;
uses srv6-state-capabilities;
uses srv6-state-sid;
}
}

/* Notifications */

grouping srv6-locator-ref {
  description "An absolute reference to an SRv6 locator";
  leaf locator-ref {
    type leafref {
    }
    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-ref;
}

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 srv6-types:srv6-end-type;
      description "End type";
    }
    // TODO: More
  }
  container requested {
    description "Requested assignment / bind";
  }
}
Figure 6: ietf-srv6-base.yang

6.3. SRv6 Static

<CODE BEGINS> file "ietf-srv6-static@2017-11-12.yang" -->

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

    import ietf-interfaces {
        prefix "if";
    }

    import ietf-inet-types {
        prefix inet;
    }

    import ietf-routing {
        prefix "rt";
    }

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

    import ietf-srv6-base {
        prefix srv6;
    }

    organization

    leaf end-behavior-type {
        type srv6-types:srv6-end-type;
        description "End type";
    }

    // TODO: More

} // module

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

WG Chair: Bruno Decraene
<mailto:loa@pi.nu>

WG Chair: Martin Vigoureux
<mailto:rcallon@juniper.net>

Editor: Kamran Raza
<mailto:skraza@cisco.com>

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

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

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

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

Editor: Daniel Voyer
<mailto:daniel.voyerbell.ca>

Editor: Hani Elmalky
<mailto:hani.elmalky@ericsson.com>

Editor: Ahmed AbdelSalam
<mailto:ahmed.abdelsalam@gssi.it>
"

description
"This YANG module defines the essential elements for the
management of Static application for 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
/* 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";
  }
  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";
  }
}
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 encap {
        description "Encapsulation on 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 encap {
    description "Encapsulation on 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 encap {
        description "Encapsulation on path";
        uses srv6:path-out-labels;
      }
    }
  }
}

grouping srv6-sid-config {
  description "Configuration parameters relating to SRv6 sid.";

  leaf opcode {
    type srv6-types:srv6-func-opcode-unreserved;
    description
  }
"SRv6 function opcode."
}
leaf end-behavior-type {
  type srv6-types:srv6-end-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.";
  }
  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."
    leaf usp {
      type boolean;
      default false;
      description
        "Ultimate SRH poping. Default is PSP";
    }
}

container end-t {
  when ".../end-behavior-type = 'End.T'" {
    description
      "This container is valid only when the user chooses End.T behavior.";
  }
  description
    "Endpoint with specific IPv6 table lookup. 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."
    leaf usp {
      type boolean;
      default false;
      description
        "Ultimate SRH poping. Default is PSP";
    }
}
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.";
  }
  description
    "Endpoint with cross-connect to an array of
     layer-3 adjacencies.
     Forward to layer-3 adjacency bound to the SID S.
     The End.X function is required to express any
     traffic-engineering policy.";
  leaf usp {
    type boolean;
    default false;
    description
      "Ultimate SRH poping. Default is PSP";
  }
  uses multi-paths-v6;
}

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.";
  uses multi-paths-v4;
}
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;
}
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.";
  leaf lookup-table-ipv4 {
    type srv6-types:table-id;
    mandatory true;
    description "IPv4 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.

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

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";
}

container end-b6 {
  when ".../end-behavior-type = 'End.B6'" {
    description
    "This container is valid only when the user chooses
     End.B6 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."
}

leaf policy-name {
  type string;
  mandatory true;
  description "SRv6 policy name.";
uses multi-paths-v6;
}

container end-b6-encaps {
  when "./end-behavior-type = 'End.B6.Encaps'" {
    description "This container is valid only when the user chooses End.B6.Encaps behavior.";
  }
  description "This is a variation of the End.B6 behavior where the SRv6 Policy also includes an IPv6 Source Address. Insert SRH based on the policy and update the source IP and forward the packet toward the first hop configured in the policy. Instead of simply inserting an SRH with the policy (End.B6), this behavior also adds an outer IPv6 header.";

  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 Label L1. This is an SRv6 instantiation of an SR-MPLS Binding SID.";

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

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 use-case";

    container paths {
        description "List of outgoing paths";

        leaf interface {
            type if:interface-ref;
            mandatory true;
            description "Layer-2 cross-connect to Out interface.";
        }
    }

    /* TODO */

    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 end-dx2v {
        type empty;
        description"
"End_DX2V behavior";
}
}
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 MAC L2 table lookup.
Pop the (outer) IPv6 header and its extension headers.
Learn the exposed inner MAC SA in L2 table T.
Lookup the exposed inner MAC DA in L2 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 end-dt2u {
  type empty;
  description
  "End_DT2U behavior";
}
}
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 L2 table flooding.
Pop the (outer) IPv6 header and its extension headers.
Learn the exposed inner MAC SA in L2 table T.
Forward on all L2OIF excluding the one specified in Arg.FE2.
The End.DT2M is used for EVPN Bridging BUM use case with
ESI filtering capability.";
leaf end-dt2m {
  type empty;
  description
  "End_DT2M behavior";
}
*/
}

grouping srv6-static-cfg {
  description
  "Grouping configuration and operation for SRv6 sid.";
}
list sid {
  key "opcode";
  description "Local SID list";

  uses srv6-sid-config;
}

augment "/rt:routing/srv6:srv6/locators/srv6:locator" {
  description "This augments locator leaf withing SRv6."

  container static {
    description "Static SRv6";

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

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

} // module

<CODE ENDS>

Figure 7: ietf-srv6-static.yang

7. Security Considerations

The configuration, state, and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC6536].
8. IANA Considerations

None

9. Acknowledgments

The authors would like to acknowledge Darren Dukes, Les Ginsberge, and Ahmed Bashandy.

10. References

10.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d.,
daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R.,
Matsushima, S., Lebrun, D., Decraene, B., Peirens, B.,
Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P.,
Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W.,
Bashandy, A., Raza, K., Dukes, D., Clad, F., and P.
Camarillo, "SRv6 Network Programming", draft-filsfils-
spring-srv6-network-programming-02 (work in progress),
October 2017.

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
and R. Wilton, "Network Management Datastore
Architecture", draft-ietf-netmod-revised-datastores-06
(work in progress), October 2017.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG
Data Model Documents", draft-ietf-netmod-rfc6087bis-14
(work in progress), September 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing
Architecture", draft-ietf-spring-segment-routing-13 (work
in progress), October 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
10.2. Informative References

[I-D.bashandy-isis-srv6-extensions]

[I-D.clad-spring-segment-routing-service-chaining]

[I-D.dawra-idr-srv6-vpn]

[I-D.filsfils-spring-segment-routing-policy]
[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B.,
daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d.,
Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi,
T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk,
"IPv6 Segment Routing Header (SRH)", draft-ietf-6man-

[I-D.ietf-spring-sr-yang]
Litkowski, S., Qu, Y., Sarkar, P., and J. Tantsura, "YANG
Data Model for Segment Routing", draft-ietf-spring-sr-
yang-07 (work in progress), July 2017.

Authors’ Addresses

Kamran Raza
Cisco Systems, Inc.
Email: skraza@cisco.com

Jaganbabu Rajamanickam
Cisco Systems, Inc.
Email: jrajan@cisco.com

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com

Iftekhar Hussain
Infinera Corporation
Email: IHussain@infinera.com

Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com
Daniel Voyer
Bell Canada
Email: daniel.voyerbell.ca

Hani Elmalky
Ericsson
Email: hani.elmalky@ericsson.com

Ahmed AbdelSalam
Gran Sasso Science Institute, Italy
Email: ahmed.abdelsalam@gssi.it