

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: 29 July 2022

J. Rajamanickam
K. Raza
Cisco Systems
D. Bernier
Bell Canada
G. Dawra
LinkedIn
C. Li
Huawei
25 January 2022

YANG Data Model for SR Service Programming
draft-jags-spring-sr-service-programming-yang-03

Abstract

This document describes a YANG data model for Segment Routing (SR) Service Programming. The model serves as a base framework for configuring and managing an SR based service programming. Additionally, this document specifies the model for a Service Proxy for SR-unaware services.

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/>.

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 29 July 2022.

Copyright Notice

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

Internet-Draft YANG Data Model for SR Service Programmi January 2022

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 Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Introduction	2
2.	Specification of Requirements	4
3.	YANG Model	4
3.1.	Overview	4
3.2.	Service Function Types	5
3.3.	SR Service Programming Types	5
3.4.	SR Service Programming Base	5
3.4.1.	Configuration	6
3.4.2.	Operational State	8
3.4.3.	Notification	10
3.5.	SR Service Proxy	10
3.5.1.	Static Proxy	11
3.5.2.	Dynamic Proxy	13
3.5.3.	Masquerading Proxy	14
4.	YANG Specification	15
4.1.	Service Types	15
4.2.	SR Service Programming Types	17
4.3.	SR Service Programming Base	22
4.4.	SR Service Proxy	32
5.	Security Considerations	39
6.	IANA Considerations	39
7.	Acknowledgments	41
8.	Normative References	41
	Authors' Addresses	43

[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 is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list

Internet-Draft YANG Data Model for SR Service Programm January 2022

of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, but also steer them through Virtual Network Function (VNF) or physical service appliances available in the network.

In an SR network, each of these services, running either on a physical appliance or in a virtual environment, are associated with a segment identifier (SID). These service SIDs are then leveraged as part of a SID-list to steer packets through the desired services in the service chain. Service SIDs may be combined together in a SID-list to achieve the service programming, but also with other types of segments as defined in [\[RFC8402\]](#). SR thus provides a fully integrated solution for overlay, underlay and service programming. Furthermore, the IPv6 instantiation of SR (SRv6) supports metadata transportation in the Segment Routing header [\[RFC8754\]](#), either natively in the tag field or with extensions such as TLVs.

This document describes how a service can be associated with a SID, including legacy services with no SR capabilities, and how these service SIDs are integrated within an SR policy. The definition of an SR Policy and the traffic steering mechanisms are covered in [\[I-D.ietf-spring-segment-routing-policy\]](#) and hence outside the scope of this document.

This document introduces a YANG data model for the SR based service programming configuration and management. Furthermore, this document also covers the basic SR unaware behaviours as defined in [\[I-D.ietf-spring-sr-service-programming\]](#).

This document does not cover the following:

- * SR-aware service specific management parameters

The model currently defines the following constructs that are used for managing SR based service programming:

- * 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 [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 the following four new YANG modules:

- * `ietf-service-function-types`: Defines common service function types
- * `ietf-sr-service-programming-types`: Defines common type definitions used for SR based service programming YANG model
- * `ietf-sr-service-programming`: Defines management model for SR based service programming framework. This is a base and common framework for both SR-aware and SR-unaware services.
- * `ietf-sr-service-programming-proxy`: Defines management model for SR service proxy for SR unaware services

The modelling 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](#)].

In this document, the SR service programming YANG model is split based on dynamic SID allocation and static SID allocation. In the case of dynamic SID allocation, new SR service programming tree would be used. In the case of static MPLS SID allocation for the SR service programming, the existing SR MPLS YANG model [[RFC9020](#)] would be augmented with the SR MPLS service programming specific parameters. Similarly the static SRv6 base YANG model (TBD) would be augmented with the SRv6 service programming specific parameters.

[3.2.](#) Service Function Types

A service is identified by (type, variant, instance). The type represents the type of service functions (such as Firewall, DPI IPS etc.), The variant value is a unique identifier which could identify the vendor and its product informations, The instance is used to refer to a specific instance of the same (service, variant).

We define a new YANG module `ietf-service-function-types` to specify common definitions and types for service and service function. The types and definitions are generic and hence can be used in any (SR based or non-SR) YANG models.

The main definitions and types defined in `ietf-service-function-types` module include:

- * `service-function-type`: A new identity type to specify service function types, such as firewall, dpi etc. Other identities can be define by other modules in future.

[3.3.](#) SR Service Programming Types

The types required to model SR based service programming are defined in a new module `ietf-sr-service-programming-types`.

The main types defined in this module includes:

- * `service-program-behaviour-type`: Defines SR service program behaviours like `sr-aware`, `static-proxy` etc...
- * `service-program-oper-status-type`: Defines SR service programming operational status. This includes the reason for down status as well
- * `service-proxy-inner-pkt-type`: Defines SR service proxy inner packet types

[3.4.](#) SR Service Programming Base

The base model and framework for SR based service programming using dynamic SID allocation is defined in a new module `ietf-sr-service-programming`.

In the case of static MPLS SID allocation for the SR service programming, the existing SR MPLS YANG model [[RFC9020](#)] would be augmented with the SR MPLS service programming specific parameters.

In the case of static SRv6 based YANG model (TBD) would be augmented with the SRv6 service programming specific parameters.

This module provides a common base for both the SR-aware and SR-unaware service programming in terms of configuration, operation state and notifications.

The `ietf-sr-service-programming` module hangs off main SR parent by augmenting `"/rt:routing/sr:segment-routing"`.

[3.4.1.](#) Configuration

This module defines some fundamental items required to configure SR based service programming. In particular, it defines service program provisioning as follows:

- * service program behaviour: Defining a service program behaviour
- * service offered: Defining a specific service (type, variant, instance) offered this service programming
- * Assigning a SR service SID: Defining SID data plane, method to allocate the SID etc..
- * service program enablement: Administratively Enable/Disable a service program
- * SR services: Defining a base container which could be augmented to define SR-aware or SR-unaware (via service-proxy) service specific parameters

Following is a simplified graphical tree representation of the data model for SR service programming (Dynamic SID allocation) base configuration only

```

module: ietf-sr-service-programming
  augment /rt:routing/sr:segment-routing:
    +--rw service-programming
      +--rw service-program* [name]
        +--rw name

```

```

-> /rt:routing/
    sr:segment-routing/
    sr-svc-pgm:service-programming/
    service-program/

```

```

service-programming-info/
service-name
+--rw sid-binding
|   +--ro alloc-mode?    sr-svc-pgm-types:sid-alloc-mode-type
|   +--rw mpls
|   |   +--ro sid?      rt-types:mpls-label
|   +--rw srv6
|       +--ro sid?      srv6-types:srv6-sid
|       +--rw locator?  -> /rt:routing/sr:segment-routing/
|                           srv6:srv6/locators/locator/name
+--rw service-programming-info
    +--rw behaviour      identityref
    +--rw dataplane      sr-svc-pgm-types:dataplane-type
    +--rw service-name   string
    +--rw service-type   identityref
    +--rw service-variant string
    +--rw service-instance uint32
    +--rw admin-status?  sr-svc-pgm-types:admin-status-type
    +--rw sr-services

```

Figure 1: SR Service Programming Config Tree - Dynamic SID allocation

Following is a simplified graphical tree representation of the data model for SR service programming (Static SR MPLS SID allocation) base configuration only. In this case SR MPLS base YANG model has been augmented to support SR service programming using static SR MPLS SID allocation. This has been done for the user convince to program all the SR service programming parameters from the based SR MPLS YANG itself


```

augment /rt:routing/sr:segment-routing/sr-mpls:sr-mpls/sr-mpls:bindings:
  +--rw mpls-static-service-programming
    +--rw service-program* [name]
      +--rw name -> /rt:routing/
                    sr:segment-routing/
                    sr-svc-pgm:service-programming/
                    service-program/
                    service-programming-info/
                    service-name
      +--rw sid rt-types:mpls-label
    +--rw service-programming-info
      +--rw behaviour identityref
      +--ro dataplane? sr-svc-pgm-types:dataplane-type
      +--rw service-name string
      +--rw service-type identityref
      +--rw service-variant string
      +--rw service-instance uint32
      +--rw admin-status? sr-svc-pgm-types:admin-status-type
      +--rw sr-services

```

Figure 2: SR Service Programming Config Tree - Static SR MPLS SID allocation

Following is a simplified graphical tree representation of the data model for SR service programming (Static SRv6 SID allocation) base configuration only. TBD (Once the based SRv6 static model is available, this section will be filled)

[3.4.2.](#) Operational 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. This section defines other operational state items related to SR based service programming.

The operational state corresponding to an SR based service program includes:

- * Operational status: Provides detail information on the operational state of the SR service program.
- * statistics: Provides the statistics details such as number of packets/bytes received, processed and dropped corresponding to a SR service program.

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

Dynamic SID allocation case:

```
module: ietf-sr-service-programming
  augment /rt:routing/sr:segment-routing:
    +--rw service-programming
      +--rw service-program* [name]
        +--rw service-programming-info
          +--ro oper-status?          identityref
          +--ro statistics
            +--ro in-packet-count?      yang:counter64
            +--ro in-bytes-count?       yang:counter64
            +--ro out-packet-count?     yang:counter64
            +--ro out-bytes-count?      yang:counter64
            +--ro in-drop-packet-count? yang:counter64
            +--ro out-drop-packet-count? yang:counter64
```

Static SR MPLS SID allocation case:

```
module: ietf-sr-service-programming
  augment /rt:routing/sr:segment-routing/sr-mpls:sr-mpls/sr-mpls:bindings:
    +--rw mpls-static-service-programming
      +--rw service-program* [name]
        +--rw service-programming-info
          +--ro oper-status?          identityref
          +--ro statistics
            +--ro in-packet-count?      yang:counter64
            +--ro in-bytes-count?       yang:counter64
            +--ro out-packet-count?     yang:counter64
            +--ro out-bytes-count?      yang:counter64
            +--ro in-drop-packet-count? yang:counter64
            +--ro out-drop-packet-count? yang:counter64
```

Static SRv6 SID allocation case:

TBD

Figure 3: SR Service Programming Operational State Tree

[3.4.3.](#) Notification

This model defines a list of notifications to inform an operator of important events detected during the SR service programming operation. These events are:

- * SR service program operational state changes: This would also give the reason for the state change when it is down

Following is a simplified graphical tree representation of the data model for the SR service programming notification:

```
module: ietf-sr-service-programming
  notifications:
    +---n service-program-oper-status
      +--ro name          -> /rt:routing/sr:segment-routing/
                           sr-svc-pgm:service-programming/
                           service-program/name
      +--ro oper-status   -> /rt:routing/sr:segment-routing/
                           sr-svc-pgm:service-programming/
                           service-program/oper-status
```

Figure 4: SR Service Programming Notification Tree

[3.5.](#) SR Service Proxy

This document also defines a separate and new YANG data model for Service Proxy for SR unaware services. The model defines the configuration and operational state related to different proxy behaviours defined earlier in `ietf-sr-service-programming-types`. The model is defined in a new module `ietf-sr-service-programming proxy`.

To support SR service programming proxy for dynamic SID allocation, this module augments the SR service program tree (`/rt:routing/sr:segment-routing/sr-svc-pgm:service-programming/ sr-svc-pgm:service-program/sr-svc-pgm:sr-services`) as defined earlier in `ietf-sr-service-programming` module.

To support SR service programming proxy for static SR MPLS SID allocation, this module augments the base SR MPLS YANG mode defined in the RFC [\[RFC9020\]](#) (/rt:routing/sr:segment-routing/sr-mpls:sr-mpls/sr-mpls:bindings/ sr-svc-pgm:mpls-static-service-programming/ sr-svc-pgm:service-program/sr-svc-pgm:service-programming-info/ sr-svc-pgm:sr-services:)

To support SR service programming proxy for static SRv6 SID allocation, this module augments the base static SRv6 model - TBD

The following sections describe different types of proxy behaviours and associated YANG modelling constructs.

[3.5.1.](#) Static Proxy

The static proxy is an SR endpoint behaviour for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware services.

The following parameters are required to provision the SR static proxy:

- * inner-packet-type: Inner packet type
- * next-hop: Next hop Ethernet address (only for the inner type is IPv4 or IPv6)
- * out-interface-name: Local interface for sending traffic towards the service Endpoint
- * in-interface-name: Local interface receiving traffic coming back from the service Endpoint
- * packet-cache-info: SR information to be attached on the traffic coming back from the service. This could be list of MPLS Label stack or SRv6 SIDs

Following is a simplified graphical tree representation of the data model for the SR static proxy:

Dynamic SID allocation case:

module: ietf-sr-service-programming-proxy

```

augment /rt:routing/sr:segment-routing/
  sr-svc-pgm:service-programming/
  sr-svc-pgm:service-program/
  sr-svc-pgm:service-programming-info/
  sr-svc-pgm:sr-services:
+--rw service-proxy
+--rw (proxy-type)
+--:(static)
+--rw static-proxy
+--rw inner-packet-type      identityref
+--rw next-hop?             yang:mac-address
+--rw out-interface-name    string
+--rw in-interface-name     string
+--rw packet-cache-info
+--rw (cache-type)
+--:(mpls)

```

```

| +--rw mpls-sids* [index]
|   +--rw index      uint8
|   +--rw mpls-label rt-types:mpls-label
+--:(srv6)
+--rw ipv6-source-address? inet:ipv6-address
+--rw srv6-sids* [index]
+--rw index      uint8
+--rw srv6-sid   srv6-types:srv6-sid

```

Static SR MPLS SID allocation case:

```

module: ietf-sr-service-programming-proxy
augment /rt:routing/sr:segment-routing/
  sr-mpls:sr-mpls/sr-mpls:bindings/
  sr-svc-pgm:mpls-static-service-programming/
  sr-svc-pgm:service-program/
  sr-svc-pgm:service-programming-info/
  sr-svc-pgm:sr-services:
+--rw static-mpls-service-proxy
+--rw (proxy-type)
+--:(static)
+--rw static-proxy
+--rw inner-packet-type      identityref

```

```

+--rw next-hop?                yang:mac-address
+--rw out-interface-name       string
+--rw in-interface-name        string
+--rw packet-cache-info
  +--rw (cache-type)
    +--:(mpls)
      | +--rw mpls-sids* [index]
      |   +--rw index      uint8
      |   +--rw mpls-label  rt-types:mpls-label
    +--:(srv6)
      +--rw ipv6-source-address? inet:ipv6-address
      +--rw srv6-sids* [index]
        +--rw index      uint8
        +--rw srv6-sid    srv6-types:srv6-sid

```

Static SRv6 SID allocation case:
TDB

Figure 5: SR Static Proxy Tree

[3.5.2.](#) Dynamic Proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can be re-attached to the traffic returning from the service Endpoints. The dynamic proxy relies on the local caching.

The following parameters are required to provision the SR dynamic proxy:

- * out-interface-name: Local interface for sending traffic towards the service Endpoint
- * in-interface-name: Local interface receiving traffic coming back from the service Endpoint

Following is a simplified graphical tree representation of the data

model for the SR static proxy:

Dynamic SID allocation case:

```
module: ietf-sr-service-programming-proxy
  augment /rt:routing/sr:segment-routing/
    sr-svc-pgm:service-programming/
    sr-svc-pgm:service-program/
    sr-svc-pgm:service-programming-info/
    sr-svc-pgm:sr-services:
  +--rw service-proxy
    +--rw (proxy-type)
      +--:(dynamic)
```

```

    +--rw dynamic-proxy
      +--rw out-interface-name  string
      +--rw in-interface-name   string

```

Static SR MPLS SID allocation case:

```

module: ietf-sr-service-programming-proxy
augment /rt:routing/sr:segment-routing/
  sr-mpls:sr-mpls/sr-mpls:bindings/
  sr-svc-pgm:mpls-static-service-programming/
  sr-svc-pgm:service-program/
  sr-svc-pgm:service-programming-info/
  sr-svc-pgm:sr-services:
    +--rw static-mpls-service-proxy
      +--rw (proxy-type)
        +--:(dynamic)
          +--rw dynamic-proxy
            +--rw out-interface-name  string
            +--rw in-interface-name   string

```

Static SRv6 SID allocation case:
TBD

Figure 6: SR Dynamic Proxy Tree

[3.5.3.](#) Masquerading Proxy

The masquerading proxy is an SR endpoint behaviour for processing SRv6 traffic on behalf of an SR-unaware service. This masquerading behaviour is independent from the inner payload type.

The following parameters are required to provision the SR masquerading proxy

- * next-hop: Next hop Ethernet address

- * out-interface-name: Local interface for sending traffic towards the service Endpoint
- * in-interface-name: Local interface receiving traffic coming back from the service Endpoint

Following is a simplified graphical tree representation of the data model for the SR masquerading proxy:

Dynamic SID allocation case:

```

module: ietf-sr-service-programming-proxy
  augment /rt:routing/sr:segment-routing/
    sr-svc-pgm:service-programming/
      sr-svc-pgm:service-program/
        sr-svc-pgm:service-programming-info/
          sr-svc-pgm:sr-services:
            +--rw service-proxy
              +--rw (proxy-type)
                +--:(masquerading)
                  +--rw masquerading-proxy
                    +--rw next-hop?                yang:mac-address
                    +--rw out-interface-name         string
                    +--rw in-interface-name          string

```

Static SRv6 SID allocation case:

TBD

Figure 7: SR masquerading Proxy Tree

[4.](#) YANG Specification

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

[4.1.](#) Service Types

Following are the Service Types definitions.

```
<CODE BEGINS> file "ietf-service-function-types.yang"
-->

module ietf-service-function-types {
  yang-version 1.1;

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

  organization "IETF SPRING Working Group";

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

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

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

    Editor:   Daniel Bernier
              <mailto:daniel.bernier@bell.ca>

    Editor:   Gaurav Dawra
              <mailto:gdawra.ietf@gmail.com>

    Editor:   Cheng Li
              <mailto:c.l@huawei.com>";

  /*
   * Below are the definition for the service types
   * Any new service type could added by extending
   * this identity
   */
  identity service-function-type {
    description
      "Base identity from which specific service function
       types are derived.";
  }

  identity firewall {
    base service-function-type;
    description
      "Firewall Service type";
  }
}
```

```
identity dpi {
```

```
    base service-function-type;
    description
        "Deep Packet Inspection Service type";
}

identity napt44 {
    base service-function-type;
    description
        "Network Address and Port Translation 44
        Service type";
}

identity classifier {
    base service-function-type;
    description
        "classifier Service type";
}

identity load-balancer {
    base service-function-type;
    description
        "load-balancer Service type";
}

identity ips {
    base service-function-type;
    description
        "Intrusion Prevention System Service type (Ex: Snort)";
}
}
<CODE ENDS>
```

Figure 8: ietf-service-function-types.yang

[4.2.](#) SR Service Programming Types

Following are the SR service programming specific types definitions.

<CODE BEGINS> file "ietf-sr-service-programming-types.yang"

-->

```
module ietf-sr-service-programming-types {  
  yang-version 1.1;  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-types";  
  prefix "sr-service-types";
```

```
  organization "IETF SPRING Working Group";  
  
  contact  
    "WG Web:  <http://tools.ietf.org/wg/spring/>  
    WG List:  <mailto:spring@ietf.org>  
  
    Editor:    Jaganbabu Rajamanickam  
              <mailto:jrajaman@cisco.com>  
  
    Editor:    Kamran Raza  
              <mailto:skraza@cisco.com>  
  
    Editor:    Daniel Bernier  
              <mailto:daniel.bernier@bell.ca>  
  
    Editor:    Gaurav Dawra  
              <mailto:gdawra.ietf@gmail.com>  
  
    Editor:    Cheng Li  
              <mailto:c.l@huawei.com>";  
  
  /*  
  * SR Service programming behaviour  
  */  
  identity service-program-behaviour-type {  
    description  
      "Base identity for SR service programming behaviour";  
  }  
  
  identity sr-aware {  
    base service-program-behaviour-type;  
    description  
      "SR aware native applications.";
```

```

}

identity static-proxy {
    base service-program-behaviour-type;
    description
        "Static Proxy";
}

identity dynamic-proxy {
    base service-program-behaviour-type;
    description
        "Dynamic Proxy";
}

identity Masquerading-proxy {

```

```

    base service-program-behaviour-type;
    description
        "Masquerading Proxy";
}

identity Masquerading-NAT-proxy {
    base service-program-behaviour-type;
    description
        "Masquerading Proxy with NAT flavor";
}

identity Masquerading-caching-proxy {
    base service-program-behaviour-type;
    description
        "Masquerading Proxy with caching flavor";
}

identity Masquerading-NAT-caching-proxy {
    base service-program-behaviour-type;
    description
        "Masquerading Proxy with caching flavor";
}

/*
 * Below are the definition for the service proxy inner packet types

```

```

    * Any new service proxy inner packet type could added by extending
    * this identity
    */
identity service-proxy-inner-pkt-type {
    description
        "Base identity from which SR service proxy types are derived.";
}

identity Ethernet {
    base service-proxy-inner-pkt-type;
    description
        "Expected inner packet type as Ethernet - derived from
        service-proxy-inner-pkt-type";
}

identity IPv4 {
    base service-proxy-inner-pkt-type;
    description
        "Expected inner packet type as IPv4 - derived from
        service-proxy-inner-pkt-type";
}

```

```

identity IPv6 {
    base service-proxy-inner-pkt-type;
    description
        "Expected inner packet type as IPv6 - derived from
        service-proxy-inner-pkt-type";
}

/*
 * SR Service SID operational status
 */
identity service-program-oper-status-type {
    description
        "Base identity from which SR service program operational
        status types are derived.";
}

identity up {
    base service-program-oper-status-type;
}

```

```

        description
            "Service program status is operational";
    }

    identity down-unknown {
        base service-program-oper-status-type;
        description
            "Service program status is down because of unknown reason";
    }

    identity sid-allocation-pending {
        base service-program-oper-status-type;
        description
            "Service program status is down because of SID allocation is pending";
    }

    identity sid-allocation-conflict {
        base service-program-oper-status-type;
        description
            "Service program status is down because of SID conflict";
    }

    identity sid-out-of-bound {
        base service-program-oper-status-type;
        description
            "Service program status is down because of SID is out of bound";
    }

    identity interface-down {

```

```

        base service-program-oper-status-type;
        description
            "Service program status is down because of out/in interface is down";
    }

    identity admin-forced-down {
        base service-program-oper-status-type;
        description
            "Service program status is administratively forced down";
    }

    /*

```

```

* Typedefs
*/
typedef admin-status-type {
  type enumeration {
    enum up {
      description "Admin Up";
    }
    enum down {
      description "Admin Down";
    }
  }
}

typedef dataplane-type {
  type enumeration {
    enum mpls {
      description "MPLS dataplane";
    }
    enum srv6 {
      description "SRv6 dataplane";
    }
  }
}

typedef sid-alloc-mode-type {
  type enumeration {
    enum static {
      description "Static SID allocation";
    }
    enum dynamic {
      description "Dynamic SID allocation";
    }
  }
}
}
<CODE ENDS>

```

Figure 9: ietf-sr-service-programming-types.yang

[4.3.](#) SR Service Programming Base

Following are the SR service programming base model definition.


```

<CODE BEGINS> file "ietf-sr-service-programming.yang"
-->

module ietf-sr-service-programming {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-service-programming";
  prefix "sr-svc-pgm";

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

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

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

  import ietf-service-function-types {
    prefix "service-types";
  }

  import ietf-segment-routing {
    prefix sr;
  }

  import ietf-segment-routing-mpls {
    prefix srmpls;
  }

  import ietf-sr-service-programming-types {
    prefix "sr-svc-pgm-types";
  }

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

```

```

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

organization "IETF SPRING Working Group";

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

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

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

    Editor:   Daniel Bernier
              <mailto:daniel.bernier@bell.ca>

    Editor:   Gaurav Dawra
              <mailto:gdawra.ietf@gmail.com>

    Editor:   Cheng Li
              <mailto:c.l@huawei.com>";

grouping service-statistics {

    container statistics {

        config false;
        description "Service statistics";

        leaf in-packet-count {
            type yang:counter64;
            description
                "Total number of packets processed by this service";
        }

        leaf in-bytes-count {
            type yang:counter64;
            description
                "Total number of bytes processed by this service";
        }

        leaf out-packet-count {
            type yang:counter64;
            description
                "Total number of packets end out after processing by this service"

```

Internet-Draft YANG Data Model for SR Service Programmi January 2022

```
    }

    leaf out-bytes-count {
      type yang:counter64;
      description
        "Total number of bytes end out after processing by this service";
    }

    leaf in-drop-packet-count {
      type yang:counter64;
      description
        "Total number of packets dropped while processing by this service"
    }

    leaf out-drop-packet-count {
      type yang:counter64;
      description
        "Total number of packets dropped while this service try to
        forward to its destination";
    }
  }
}

grouping service-mpls-sid-binding {
  container mpls {
    description
      "MPLS Service SID binding Container";

    when "../../service-programming-info/dataplane = 'mpls'";

    leaf sid {
      config false;
      type rt-types:mpls-label;
      description
        "MPLS SID value.";
    }
  }
}

grouping service-srv6-sid-binding {
  container srv6 {
    description
```

```

    "SRv6 Service SID binding Container";

when "../../../service-programming-info/dataplane = 'srv6'";

leaf sid {
    config false;

```

```

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

leaf locator {
    type leafref {
        path "/rt:routing/sr:segment-routing"
            + "/srv6:srv6/srv6:locators/srv6:locator/srv6:name";
    }
    description
        "Reference to a SRv6 locator. This is valid only when
        the SID allocation mode is dynamic";
}
}

grouping service-sid-binding {
    container sid-binding {
        description
            "Service SID binding Container";

        leaf alloc-mode {
            config false;
            default dynamic;
            type sr-svc-pgm-types:sid-alloc-mode-type;
            description
                "Service SID allocation mode";
        }

        uses service-mpls-sid-binding;
        uses service-srv6-sid-binding;
    }
}

```

```

grouping service-programming-infos {
  container service-programming-info {

    leaf behaviour {
      mandatory true;
      type identityref {
        base sr-svc-pgm-types:service-program-behaviour-type;
      }
      description
        "SR program behaviour";
    }

    leaf dataplane {

```

```

      mandatory true;
      type sr-svc-pgm-types:dataplane-type;
      description
        "Service SID dataplane.";
    }

    leaf service-name {
      mandatory true;
      type string;
      description
        "Service program name to identify a specific program.";
    }

    leaf service-type {
      mandatory true;
      type identityref {
        base service-types:service-function-type;
      }
      description
        "Service-Type defined by IANA Service Type Table (STT). Like
        Firewall, DPI etc...";
    }

    leaf service-variant {
      mandatory true;
      type string;
      description
        "This identifies the variant of the service. This value should

```

```

        be unique in the given network. Example Format:
        <vendor>-<vendor-sub-variant>-<product-version>.";
    }

    leaf service-instance {
        mandatory true;
        type uint32;
        description
            "Service instance which differentiates the same service -- e.g.
            same vendors Firewall service could have several instances
            available. This could be used to differentiate the VPN
            customers or for load sharing purposes.";
    }

    leaf admin-status {
        type sr-svc-pgm-types:admin-status-type;
        default down;
        description
            "Admin Status";
    }

```

```

    leaf oper-status {
        config false;
        type identityref {
            base sr-svc-pgm-types:service-program-oper-status-type;
        }
        description
            "Service SID operational mode.";
    }

    uses service-statistics;

    container sr-services {

        description
            "Any SR-aware or AR-unaware services could augment this contain
            reference "Segment Routing Service Programming Architecture.";
    }
}

```

```

grouping service-programmings {
  container service-programming {
    description
      "service programming container.
      Any new services programming added could augment
      this container to support that specific services.
      Currently in this model, only service proxy
      is defined. (i.e) For example if
      a Firewall services needs to be added then
      they could augment this container and
      extend this model";

    list service-program {
      key "name";
      description
        "Service program is keyed by the service program name";

      leaf name {
        mandatory true;
        type leafref {
          path "/rt:routing/sr:segment-routing/"
            + "sr-svc-pgm:service-programming/"
            + "sr-svc-pgm:service-program/"
            + "sr-svc-pgm:service-programming-info/"
            + "sr-svc-pgm:service-name";
        }
      }
    }
  }
}

```

```

    uses service-sid-binding;
    uses service-programming-infos;
  }
}

/*
 * MPLS/SRv6 SR service programming using dynamic SID allocation
 */
augment "/rt:routing/sr:segment-routing" {
  description
    "Augmenting the segment-routing to add SR service programming";

  uses service-programmings;
}

```

```

}

/*
 * MPLS SR service programming using static MPLS binding SID
 */
augment "/rt:routing/sr:segment-routing/srmls:sr-mpls/srmls:bindings" {
  description
    "Augmenting the segment-routing MPLS static binding to add static
    MPLS SR service programming";

  container mpls-static-service-programming {
    description
      "Augmenting the MPLS segment-routing bindings with the SR service
      programming";
    list service-program {
      key "name";
      description
        "Service program is keyed by the service program name";

      leaf name {
        mandatory true;
        type leafref {
          path "/rt:routing/sr:segment-routing/"
            + "sr-svc-pgm:service-programming/"
            + "sr-svc-pgm:service-program/"
            + "sr-svc-pgm:service-programming-info/"
            + "sr-svc-pgm:service-name";
        }
      }

      leaf sid {
        mandatory true;
        type rt-types:mpls-label;
        description

```

```

    "MPLS SID value.";
  }

  uses service-programming-infos {
    /*
     * In the case of MPLs static binding configuration
     * the dataplane is set to mpls and not allowed to

```



```

        * configure
        */
        refine service-programming-info/dataplane {
            mandatory false;
            default mpls;
            config false;
        }
    }
}

}

}

/*
 * SRv6 SR service programming using static SRv6 binding SID
 */
augment "/rt:routing/sr:segment-routing/srv6:srv6/srv6:locators/srv6:locat
description
    "Augmenting the segment-routing SRv6 static to add static binding to
    SRv6 SR service programming";

container end-AS {
    description
        "End.AS - Static Proxy SID behaviour";
    list service-program {
        key "name";
        description
            "Service program is keyed by the service program name";

        leaf name {
            mandatory true;
            type leafref {
                path "/rt:routing/sr:segment-routing/"
                    + "sr-svc-pgm:service-programming/"
                    + "sr-svc-pgm:service-program/"
                    + "sr-svc-pgm:service-programming-info/"
                    + "sr-svc-pgm:service-name";
            }
        }
    }

    uses service-programming-infos {

```

```

/*
 * In the case of SRv6 static binding configuration
 * the dataplane is set to mpls and not allowed to
 * configure
 */
refine service-programming-info/dataplane {
    config false;
    mandatory false;
    default srv6;
}
refine service-programming-info/behaviour {
    config false;
    //when "service-programming-info/dataplane = 'srv6'";
    mandatory false;
    default sr-svc-pgm-types:static-proxy;
}
}
}
}

```

```

container end-AD {
    description
        "End.AD - Dynamic Proxy SID behaviour";
    list service-program {
        key "name";
        description
            "Service program is keyed by the service program name";

        leaf name {
            mandatory true;
            type leafref {
                path "/rt:routing/sr:segment-routing/"
                    + "sr-svc-pgm:service-programming/"
                    + "sr-svc-pgm:service-program/"
                    + "sr-svc-pgm:service-programming-info/"
                    + "sr-svc-pgm:service-name";
            }
        }
    }

    uses service-programming-infos {

        refine service-programming-info/dataplane {
            config false;
            mandatory false;
            default srv6;
        }
        refine service-programming-info/behaviour {

```

Internet-Draft YANG Data Model for SR Service Programmi January 2022

```
        //when "service-programming-info/dataplane = 'srv6'";
        config false;
        mandatory false;
        default sr-svc-pgm-types:dynamic-proxy;
    }

}

}

}

container end-AM {
    description
        "End.AD - Masquerading Proxy SID behaviour";
    list service-program {
        key "name";
        description
            "Service program is keyed by the service program name";

        leaf name {
            mandatory true;
            type leafref {
                path "/rt:routing/sr:segment-routing/"
                    + "sr-svc-pgm:service-programming/"
                    + "sr-svc-pgm:service-program/"
                    + "sr-svc-pgm:service-programming-info/"
                    + "sr-svc-pgm:service-name";
            }
        }
    }

    uses service-programming-infos {

        refine service-programming-info/dataplane {
            config false;
            mandatory false;
            default srv6;
        }

        refine service-programming-info/behaviour {
            //when "service-programming-info/dataplane = 'srv6'";
            mandatory false;
            default sr-svc-pgm-types:Masquerading-proxy;
        }
    }
}
```

```

    }
  }
}

```

```

notification service-program-oper-status {
  description
    "This notification is sent when there is a change in the service
    program oper status.";
  leaf name {
    mandatory true;
    type leafref {
      path "/rt:routing/sr:segment-routing/"
        + "sr-svc-pgm:service-programming/"
        + "sr-svc-pgm:service-program/"
        + "sr-svc-pgm:name";
    }
    description
      "Service program name to identify a specific programming.";
  }

  leaf oper-status {
    mandatory true;
    type leafref {
      path "/rt:routing/sr:segment-routing/"
        + "sr-svc-pgm:service-programming/"
        + "sr-svc-pgm:service-program/"
        + "sr-svc-pgm:service-programming-info/"
        + "sr-svc-pgm:oper-status";
    }
    description
      "Service program operational status.";
  }
}
}
<CODE ENDS>

```

Figure 10: ietf-sr-service-programming.yang

[4.4.](#) SR Service Proxy

Following are the SR service programming service proxy model definition.

```
<CODE BEGINS> file "ietf-sr-service-programming-proxy.yang"
-->
module ietf-sr-service-programming-proxy {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-proxy";
  prefix "sr-svc-proxy";
```

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

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

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

import ietf-segment-routing {
  prefix sr;
}

import ietf-sr-service-programming {
  prefix "sr-svc-pgm";
}

import ietf-sr-service-programming-types {
  prefix "sr-svc-pgm-types";
}

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

```

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

import ietf-segment-routing-mpls {
    prefix sr-mpls;
}

organization "IETF SPRING Working Group";

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

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

    Editor:    Kamran Raza

```

```

    <mailto:skraza@cisco.com>

    Editor:    Daniel Bernier
               <mailto:daniel.bernier@bell.ca>

    Editor:    Gaurav Dawra
               <mailto:gdawra.ietf@gmail.com>

    Editor:    Cheng Li
               <mailto:c.l@huawei.com>";

grouping service-proxy-parameters {

    leaf out-interface-name {
        mandatory true;
        type string;
        description
            "Interface name on which the packet sent to the service endpoint";
    }

    leaf in-interface-name {
        mandatory true;

```

```

        type string;
        description
            "Interface name on which the packet received from the service endpoi
    }
}

grouping mpls-packet-cache-info {
    description
        "MPLS Label stack";

    list mpls-sids {
        key "index";

        leaf index {
            type uint8 {
                range "1..16";
            }
            description
                "cache index - MPLS Label stack index";
        }

        leaf mpls-label {
            mandatory true;
            type rt-types:mpls-label;
            description
                "MPLS Label value.";
        }
    }
}

```

```

    }
}
}

grouping srv6-packet-cache-info {
    description
        "SRv6 SID stack";

    leaf ipv6-source-address {
        type inet:ipv6-address;
        description
            "IPv6 source address that needs in the case if SRv6.";
    }
    list srv6-sids {
        key "index";
    }
}

```

```

    leaf index {
      type uint8 {
        range "1..16";
      }
      description
        "cache index - SRv6 SID index";
    }

    leaf srv6-sid {
      mandatory true;
      type srv6-types:srv6-sid;
      description
        "SRv6 SID.";
    }
  }
}

grouping service-proxy-packet-cache-info {
  description
    "SRv6 Proxy header cache";

  container packet-cache-info {

    choice cache-type {
      mandatory true;
      case mpls {

        when "/rt:routing/sr:segment-routing
          /sr-svc-pgm:service-programming
          /sr-svc-pgm:service-program
          /sr-svc-pgm:service-programming-info
          /sr-svc-pgm:dataplane = 'mpls'";

```

```

    uses mpls-packet-cache-info;
  }
  case srv6 {

    when "/rt:routing/sr:segment-routing/sr-svc-pgm:service-programmin
      /sr-svc-pgm:service-program
      /sr-svc-pgm:service-programming-info
      /sr-svc-pgm:dataplane = 'srv6'";

```



```

        uses srv6-packet-cache-info;
    }
}
}

grouping static-service-proxy {
    container static-proxy {
        when "/rt:routing/sr:segment-routing/sr-svc-pgm:service-programming
            /sr-svc-pgm:service-program
            /sr-svc-pgm:service-programming-info
            /sr-svc-pgm:behaviour = 'static-proxy'";
        description
            "Parameters related to static service proxy";

        leaf inner-packet-type {
            mandatory true;
            type identityref {
                base sr-svc-pgm-types:service-proxy-inner-pkt-type;
            }
            description
                "Defines the expected inner packet type";
        }

        leaf next-hop {
            when "(../inner-packet-type = 'IPv4' or ../inner-packet-type = 'IPv6'
                type yang:mac-address;
            description
                "Nexthop Ethernet address for inner packet type IPv4/IPv6";
        }
        uses service-proxy-parameters;
        uses service-proxy-packet-cache-info;
    }
}

grouping dynamic-service-proxy {
    container dynamic-proxy {
        when "/rt:routing/sr:segment-routing/sr-svc-pgm:service-programming
            /sr-svc-pgm:service-program

```

```

        /sr-svc-pgm:behaviour = 'dynamic-proxy';
    description
        "Parameters related to dynamic service proxy";
    uses service-proxy-parameters;
}
}

grouping masquerading-service-parameters {

    leaf next-hop {
        type yang:mac-address;
        description
            "Nexthop Ethernet address";
    }
    uses service-proxy-parameters;
}

grouping masquerading-service-proxy {
    container masquerading-proxy {
        description
            "Parameters related to masquerading service proxy";

        when "/rt:routing/sr:segment-routing
            /sr-svc-pgm:service-programming
            /sr-svc-pgm:service-program
            /sr-svc-pgm:service-programming-info
            /sr-svc-pgm:dataplane = 'srv6' and /rt:routing
            /sr:segment-routing/sr-svc-pgm:service-programming
            /sr-svc-pgm:service-program
            /sr-svc-pgm:service-programming-info
            /sr-svc-pgm:behaviour = 'Masquerading-proxy'";

        uses masquerading-service-parameters;
    }
}

grouping service-proxy-programming {
    container service-proxy {

        choice proxy-type {
            mandatory true;
            case static {
                uses static-service-proxy;
            }
            case dynamic {
                uses dynamic-service-proxy;
            }
        }
    }
}

```

```
        case masquerading {
            uses masquerading-service-proxy;
        }
    }
}

}

augment "/rt:routing/sr:segment-routing/
    sr-svc-pgm:service-programming/
    sr-svc-pgm:service-program/
    sr-svc-pgm:service-programming-info/
    sr-svc-pgm:sr-services" {
    description
        "Augmenting the segment-routing bindings to add SR-unaware
        service programming";

    uses service-proxy-programming;
}

grouping static-mpls-service-proxy-programming {
    container static-mpls-service-proxy {

        choice proxy-type {
            mandatory true;
            case static {
                uses static-service-proxy;
            }
            case dynamic {
                uses dynamic-service-proxy;
            }
        }
    }
}

}

augment "/rt:routing/sr:segment-routing/
    sr-mpls:sr-mpls/sr-mpls:bindings/
    sr-svc-pgm:mpls-static-service-programming/
    sr-svc-pgm:service-program/
    sr-svc-pgm:service-programming-info/
    sr-svc-pgm:sr-services" {
    uses static-mpls-service-proxy-programming;
}

}
```

Figure 11: ietf-sr-service-programming-proxy.yang

5. 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 [[I-D.ietf-spring-sr-service-programming](#)].

6. IANA Considerations

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

Internet-Draft YANG Data Model for SR Service Programming January 2022

URI	Registrant	XML
urn:ietf:params:xml:ns:yang:ietf-service-function-types	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-types	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-sr-service-programming	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-proxy	The IESG	N/A

Table 1

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

Name	Namespace	Prefix	Reference
ietf-service-function-types	urn:ietf:params:xml:ns:yang:ietf-service-function-types	service-function-types	This document
ietf-sr-service-programming-types	urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-types	ietf-sr-service-programming-types	This document
ietf-sr-service-programming	urn:ietf:params:xml:ns:yang:ietf-sr-service-programming	ietf-sr-service-programming	This document
ietf-sr-service-programming-proxy	urn:ietf:params:xml:ns:yang:ietf-sr-service-programming-proxy	ietf-sr-service-programming-proxy	This document

Table 2

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

7. Acknowledgments

The authors would like to acknowledge Francois Clad, Ketan Talaulikar, and Darren Dukes for their review of some of the contents in this document.

8. Normative References

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", Work in Progress, Internet-Draft, [draft-ietf-spring-segment-routing-policy-14](https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-policy-14), 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-policy-14.txt>>.

Rajamanickam, et al.

Expires 29 July 2022

[Page 41]

Internet-Draft YANG Data Model for SR Service Programm January 2022

[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", Work in Progress, Internet-Draft, [draft-ietf-spring-sr-service-programming-05](https://www.ietf.org/archive/id/draft-ietf-spring-sr-service-programming-05), 10 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-spring-sr-service-programming-05.txt>>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](https://www.rfc-editor.org/info/rfc3688), [RFC 3688](https://www.rfc-editor.org/info/rfc3688), 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>>.
- [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>>.
- [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>>.
- [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>>.

[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>>.

[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>>.

Authors' Addresses

Jaganbabu Rajamanickam
Cisco Systems

Email: jrajaman@cisco.com

Kamran Raza
Cisco Systems

Email: skraza@cisco.com

Daniel Bernier
Bell Canada

Email: daniel.bernier@bell.ca

Gaurav Dawra
LinkedIn

Email: gdawra.ietf@gmail.com

Cheng Li
Huawei

Email: c.l@huawei.com