SFC Netmod
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2015

R. Penno
P. Quinn
Cisco Systems
D. Zhou
J. Li
Intel Corporation

March 02, 2015

Yang Data Model for Service Function Chaining
draft-penno-sfc-yang-13

Abstract

This document defines a YANG data model that can be used to configure and manage Service Function Chains.

Requirements Language

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 RFC 2119 [RFC2119].

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 http://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 September 3, 2015.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage Service Function Chains

2. Definitions and Acronyms

The reader should be familiar with the terms contained in [I-D.ietf-sfc-architecture], [I-D.ietf-sfc-problem-statement] ,[I-D.ietf-sfc-architecture] and [I-D.quinn-vxlan-gpe]

3. Understanding SFC Yang Models

There are two main models in SFC: service-function (SF) and service-function-forwarder (SFF). Most other models are used or derived from those models. SF describes a service function like firewall, napt44, dpi, http-proxy, etc. SFF describes a forwarding element that moves packets along a service path. A SFF to function only needs to be able to associate a Service Path ID and SI to a next hop data plane locator.

The service-locator model provides a centralized place to register transport and endpoints used with SFFs and SFs. This allows reuse across a large number of other models since in networking usually data plane locators are widely used. Some examples of transport types are GRE, VXLAN-GPE and the data plane locator are IP:port, VLAN-ID and MPLS Label. This model is imported by SFF, SF and Rendered Service Path (RSP) models.

Service Function Type model serves as a registry for SF types. The model can be easily extended by anyone looking to define their own service type. This model is imported by SF and Service Function Chain (SFC). Since a SFC is an abstract order of service function

types, having a registry of types is important. Furthermore, when we instantiate a SFP and RSP from a SFC we need to choose the actual SFs that will be traversed by the packets and this requires us to know the type associated with a Service Function.

A service function path (SFP) is an intermediate step between SFC and RSP. It allows the user to provide input or constraints into the construction of a RSP. This input ranges from nothing to specifying the entire path. During RSP construction, the controller examines the SFP and 'fill in the blanks'.

One of the most important configuration aspects of a SF is the data plane locators. A SF's data plane locators indicates how the SF can be reached. A SF can have multiple data plane locators of different transport and types as specified in the service locator model.

A SFF has also can have multiple data plane locators that indicate how it can be reached. It is very important when constructing a RSP to pick SFFs that have data plane locators of the same transport and type so that the path works. A SFF has an additional very important configuration container, the service function dictionary. The service function dictionary stores the SFF's view of the Service Functions. It contains all SFs and their data plane locators.

Therefore the Service Function data plane locators and the SFF service function dictionary constitute two pieces of a puzzle. If they fit, it means they can be used in a path, otherwise they can not.

The RSP model is the result of creating a Service Function Chain, applying policies through the Service Function Path and finally choosing a collection of (SFF, SF) tuples that meet these criteria. The RSP is an operational model, meaning it can read but not changed.

4. Service Function (SF)

This module describe a Service Function, which is an essential building block of other modules.

4.1. Module Structure

```
module: service-function
   +--rw service-functions
   |  +--rw service-function* [name]
   |     +--rw name                          string
   |     +--rw type                          sfc-sft:service-function-type
   |     +--rw rest-uri?                     inet:uri
   |     +--rw ip-mgmt-address?              inet:ip-address
```



```
   |     +--rw request_reclassification?   boolean
   |     +--rw nsh-aware?                  boolean
   |     +--rw sf-data-plane-locator* [name]
   |        +--rw name                            string
   |        +--rw (locator-type)
   |        |  +--:(ip)
   |        |  |  +--rw ip?                             inet:ip-address
   |        |  |  +--rw port?                           inet:port-number
   |        |  +--:(lisp)
   |        |  |  +--rw eid?                            inet:ip-address
   |        |  +--:(mac)
   |        |  |  +--rw mac?                            yang:mac-address
   |        |  |  +--rw vlan-id?                        uint16
   |        |  +--:(function)
   |        |  |  +--rw function-name?                  string
   |        |  +--:(mpls)
   |        |     +--rw mpls-label?                     uint32
   |        +--rw transport?                       identityref
   |        +--rw service-function-forwarder?     string
   +--ro service-functions-state
      +--ro service-function-state* [name]
         +--ro name                string
```

```
         +--ro sf-service-path* [name]
            +--ro name    string
rpcs:
   +---x delete-all-service-function
   +---x put-service-function
   |  +--ro input
   |     +--ro name?                        string
   |     +--ro type                         sfc-sft:service-function-type
   |     +--ro rest-uri?                    inet:uri
   |     +--ro ip-mgmt-address?             inet:ip-address
   |     +--ro request_reclassification?    boolean
   |     +--ro nsh-aware?                   boolean
   |     +--ro sf-data-plane-locator* [name]
   |        +--ro name                              string
   |        +--ro (locator-type)
   |        |  +--:(ip)
   |        |  |  +--ro ip?                               inet:ip-address
   |        |  |  +--ro port?                             inet:port-number
   |        |  +--:(lisp)
   |        |  |  +--ro eid?                              inet:ip-address
   |        |  +--:(mac)
   |        |  |  +--ro mac?                              yang:mac-address
   |        |  |  +--ro vlan-id?                          uint16
   |        |  +--:(function)
   |        |  |  +--ro function-name?                    string
   |        |  +--:(mpls)
```



```
   |        |     +--ro mpls-label?                    uint32
   |        +--ro transport?                    identityref
   |        +--ro service-function-forwarder?   string
   +---x read-service-function
   |  +--ro input
   |  |  +--ro name    string
   |  +--ro output
   |     +--ro name?                        string
   |     +--ro type                         sfc-sft:service-function-type
   |     +--ro rest-uri?                    inet:uri
   |     +--ro ip-mgmt-address?             inet:ip-address
   |     +--ro request_reclassification?    boolean
   |     +--ro nsh-aware?                   boolean
   |     +--ro sf-data-plane-locator* [name]
   |        +--ro name                              string
```

```
|        +--ro (locator-type)
|        |  +--:(ip)
|        |  |  +--ro ip?                              inet:ip-address
|        |  |  +--ro port?                            inet:port-number
|        |  +--:(lisp)
|        |  |  +--ro eid?                             inet:ip-address
|        |  +--:(mac)
|        |  |  +--ro mac?                             yang:mac-address
|        |  |  +--ro vlan-id?                         uint16
|        |  +--:(function)
|        |  |  +--ro function-name?                   string
|        |  +--:(mpls)
|        |     +--ro mpls-label?                      uint32
|        +--ro transport?                         identityref
|        +--ro service-function-forwarder?       string
+---x delete-service-function
   +--ro input
      +--ro name    string
```

4.2. Service Function Configuration Module

```
<CODE BEGINS> file "service-function@2014-07-01.yang"

module service-function {

  namespace "urn:cisco:params:xml:ns:yang:sfc-sf";

  prefix sfc-sf;

  import ietf-inet-types { prefix inet; }
  import ietf-yang-types { prefix yang; }
  import service-function-type {prefix sfc-sft;}
  import service-locator {prefix sfc-sl;}
```



```
  organization "Cisco Systems, Inc.";
  contact "Reinaldo Penno <repenno@cisco.com>";


  description
    "This module contains a collection of YANG definitions for
     managing service function.
```

```
      It follows closely the constructs of
      http://tools.ietf.org/html/draft-ietf-netmod-interfaces-cfg-12

      Copyright (c) 2013 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).

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

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

      // RFC Ed.: update the date below with the date of RFC publication
      // and remove this note.

   revision 2014-07-01 {
     description
     "Changes based on Opendaylight Testing.";
   }


   typedef service-function-ref {
     type leafref {
       path "/sfc-sf:service-functions/sfc-sf:service-function/"
           + "sfc-sf:name";
     }
     description
       "This type is used by data models that need to reference
        configured service functions.";
   }

   grouping service-function-entry {
     description
```



```
       "This group bundles together all information related to a
```

```
   single service function";
  leaf name {
    type string;
    description
        "The name of the service function.";
  }
  leaf type {
    type sfc-sft:service-function-type;
    mandatory true;
    description
      "Service Function Type from service-function-type yang
       model";
  }
  leaf rest-uri {
    description "URI of REST based management";
    type inet:uri;
  }
  leaf ip-mgmt-address {
    type inet:ip-address;
    description
      "The IP and port used to configure this service-function";
  }
  leaf request_reclassification {
    description "This leaf determines whether SF can request
      reclassification by the SFF";
    type boolean;
  }
  leaf nsh-aware {
    type boolean;
    description "Whether this SF can process NSH headers";
  }

  list sf-data-plane-locator {
    description
      "A network data-plane locator";
    key "name";
    leaf name {
      type string;
      description
        "A unique string that represents this
         data-plane-locator";
    }
    uses sfc-sl:data-plane-locator;
    leaf service-function-forwarder {
      type string;
      description
        "The service function forwarder associated with this
```

```
            locator";
        }
      }
    }

    container service-functions {
      description
        "A function that is responsible for specific
        treatment of received packets.  A Service Function can act at
        various layers of a protocol stack (e.g., at the network layer
        or other OSI layers).  A Service Function can be a virtual
        element or be embedded in a physical network element.  One of
        multiple Service Functions can be embedded in the same network
        element.  Multiple occurrences of the Service Function can be
        enabled in the same administrative domain.

        One or more Service Functions can be involved in the delivery
        of added-value services.  A non-exhaustive list of Service
        Functions includes: firewalls, WAN and application
        acceleration, Deep Packet Inspection (DPI),a LI (Lawful
        Intercept) module, server load balancers, NAT44 [RFC3022],
        NAT64 [RFC6146], NPTv6 [RFC6296], HOST_ID injection, HTTP
        Header Enrichment functions, TCP optimizer, etc.

        An SF may be SFC encapsulation aware, that is it receives, and
        acts on information in the SFC encapsulation, or unaware in
        which case data forwarded to the service does not contain the
        SFC encapsulation.";

      list service-function {
        description
          "This list holds configuration data for all service functions
           in the domain";
        key "name";
        uses service-function-entry;
      }
    }

    container service-functions-state {
      description
        "This container hold operational state for all service
         functions";
      config false;
      list service-function-state {
        description
          "This list holds operational data for all service functions
```

```
          in the domain";
        key "name";
```



```
        leaf name {
          type string;
          description
            "the name of the service function";
        }
        list sf-service-path {
          key "name";
          leaf name {
            type string;
            description
              "The name of the Service Path";
          }
          description
          "A list of all service function paths that contain this
           service function";
        }
      }
    }

    rpc delete-all-service-function {
    description
      "Deletes all service functions";
    }

    rpc put-service-function {
      description
        "Creates a service function";
      input {
        uses service-function-entry;
      }
    }
    rpc read-service-function {
      description
        "Reads a service function";
      input {
        leaf name {
          type string;
          mandatory true;
          description "The name of the service function.";
```

```
      }
    }
    output {
      uses service-function-entry;
    }
  }
  rpc delete-service-function {
    description
      "Deletes a service function";
```



```
    input {
      leaf name {
        type string;
        mandatory true;
        description "The name of the service function.";
      }
    }
  }
}

</CODE ENDS>
```

5. Service Function Type (SFT)

This module holds one list for each service function type found in the system. Each one of these lists has the name of all service functions configured on the system of that particular type. This allows finding a service function of a given type simple.

5.1. Module Structure

```
module: service-function-type
   +--rw service-function-types
      +--rw service-function-type* [type]
         +--rw type                          service-function-type
         +--rw sft-service-function-name* [name]
            +--rw name    string
```

5.2. Service Function Type Configuration Model

<CODE BEGINS> file "service-function-type@2014-07-01.yang"

```
module service-function-type {

  namespace "urn:cisco:params:xml:ns:yang:sfc-sft";

  prefix sfc-sft;

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


  organization "Cisco Systems, Inc.";
  contact "Reinaldo Penno <repenno@cisco.com>";


  description
```



```
    "This module contains a collection of YANG definitions for
     managing service function types.

     It follows closely the constructs of
     http://tools.ietf.org/html/draft-ietf-netmod-interfaces-cfg-12

     Copyright (c) 2013 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).

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
```

```
  revision 2014-07-01 {
    description
    "Changes based on Opendaylight Testing.";
  }


  // Service Function

  // Service Function Type definitions

  identity service-function-type-identity {
    description
      "Base identity from which specific service function
       types are derived.";
  }

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

  identity dpi {
    base "service-function-type-identity";
```



```
    description "Deep Packet Inspection";
  }

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

  identity qos {
    base "service-function-type-identity";
    description "Quality of Service";
  }

  identity ids {
    base "service-function-type-identity";
    description "Intrusion Detection System";
  }
```

```
  identity http-header-enrichment {
    base "service-function-type-identity";
    description "HTTP services that add HTTP headers for charging and adding su
  }

  typedef service-function-type {
    type identityref {
      base "service-function-type-identity";
    }
    description "This type is used to reference all
      registered service function types";
  }

  container service-function-types {
    description
      "A list of Service function Types. For each type we keep
       a list of Service Functions";

    list service-function-type {
      key "type";
      leaf type {
        type service-function-type;
        description
            "The service function type";
      }
      list sft-service-function-name {
        key "name";
        leaf name {
          type string;
          description
```



```
            "The name of the service function.";
        }
        description
          "The list of all service functions of a specific type";
      }
      description
        "A list of all service types. Each service-type entry holds
         a list of all service functions of the type";
    }
  }
}
```

</CODE ENDS>

6. Service Function Chain (SFC)

This model describes a service function chain which is basically an ordered list of services. But a service function chain does not specify exactly which service (firewall1 vs. firewall2) will be used to actually process packets.

6.1. Module Structure


```
module: service-function-chain
   +--rw service-function-chains
   |  +--rw service-function-chain* [name]
   |     +--rw name                    string
```

```
      |     +--rw symmetric?              boolean
      |     +--rw sfc-service-function* [name]
      |        +--rw name     string
      |        +--rw type     sfc-sft:service-function-type
      |        +--rw order?   uint8
      +--ro service-function-chains-state
         +--ro service-function-chain-state* [name]
            +--ro name                          string
            +--ro sfc-service-function-path*    string
   rpcs:
      +---x instantiate-service-function-chain
      |  +--ro input
      |  |  +--ro name    string
      |  +--ro output
      |     +--ro name?   string
      +---x put-service-function-chains
         +--ro input
            +--ro service-function-chain* [name]
               +--ro name                     string
               +--ro symmetric?               boolean
               +--ro sfc-service-function* [name]
                  +--ro name     string
                  +--ro type     sfc-sft:service-function-type
                  +--ro order?   uint8
```

6.2. Service Function Chain Configuration Model

```
 <CODE BEGINS> file "service-function-chain@2014-07-01.yang"

 module service-function-chain {

   namespace "urn:cisco:params:xml:ns:yang:sfc-sfc";

   prefix sfc-sfc;

   import ietf-inet-types { prefix inet; }
   import ietf-yang-types { prefix yang; }
   import service-function {prefix sfc-sf; }
   import service-function-type {prefix sfc-sft;}


   organization "Cisco Systems, Inc.";
   contact "Reinaldo Penno <repenno@cisco.com>";
```

```
   description
     "This module contains a collection of YANG definitions for
      managing service function chains.

      Copyright (c) 2013 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).

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

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

      // RFC Ed.: update the date below with the date of RFC publication
      // and remove this note.

   revision 2014-07-01 {
     description
     "Revised based on Opendaylight Project feedback";
   }

   grouping service-function-chain-grouping {
     list service-function-chain {
       description
         "A list that holds all service function chains in the
          domain";
       key "name";
       leaf name {
         type string;
         description
           "the name of the service function chain";
       }
       leaf symmetric {
         type boolean;
         default false;
         description
           "If the chain is symmetric we will create two service
            paths, one ingress and another egress. Packets traverse
            the egress service path in the reverse order of the
            ingress path";
```

```
      }
```



```
      list sfc-service-function {
        key "name";
        leaf name {
          type string;
          description
            "A unique handle that describes the service function
             that will be chosen for this type, such as
             ingress-dpi. This is not the service function name";
        }
        leaf type {
          type sfc-sft:service-function-type;
          mandatory true;
          description
            "Service Function Type from service-function-type.yang";
        }
        leaf order {
          type uint8;
        }
        ordered-by user;
        description
          "A list of service functions that compose the service
           chain";
      }
    }
    description
      "This group bundles together all service function chains in the
       domain";
  }

  // Service Function Chains

  container service-function-chains {
    uses service-function-chain-grouping;
    description
      "A service Function chain defines an
       abstract set of service functions and their ordering
       constraints that must be applied to packets and/or frames
       selected as a result of classification.  The implied order
       may not be a linear progression as the architecture allows
       for nodes that copy to more than one branch, and also allows
```

```
        for cases where there is flexibility in the order in which
        services need to be applied. The term service chain is often
        used as shorthand for service function chain.";
    }

    container service-function-chains-state {
      config false;
      list service-function-chain-state {
```



```
        description
          "A list that contains operational service function
           chain state";
        key "name";
        leaf name {
          type string;
          description
            "the name of the service function chain";
        }
        leaf-list sfc-service-function-path {
          type string;
          description
          "A list of all service function paths instantiated from
           this chain";
        }
      }
      description
        "This containers holds operational service function
          chain state and their associated service path";
    }

  // Remote procedure calls

  // (main feature: instantiation of a SFC)

    rpc instantiate-service-function-chain {
      description
        "Instantiates a single service function";
      input {
        leaf name {
          type string;
            mandatory true;
            description
```

```
            "The name of the service function chain to be
             instantiated.";
        }
      }
      output {
        leaf name {
          type string;
          description
            "The name of the created service function path.";
        }
      }
    }

      // (RPC for testing)
    rpc put-service-function-chains {
```



```
      description
        "Creates Service-Functions";
      input {
        uses service-function-chain-grouping;
      }
    }
  }
```

</CODE ENDS>

7. Service Function Path (SFP)

A Service Function Path is an instantiation of a service function chain. It allows be user to provide constrains for the rendering of the service path such as specific service-hops that need to be visited, the transport encapsulation used in the overlay, whether paths should be symmetric, amongst others.

7.1. Module Structure

module: service-function-path

```
   +--rw service-function-paths
   |  +--rw service-function-path* [name]
   |     +--rw name                     string
   |     +--rw symmetric?               boolean
   |     +--rw classifier?              string
   |     +--rw symmetric-classifier?    string
   |     +--rw context-metadata?        sfc-md:context-metadata-ref
   |     +--rw variable-metadata?       sfc-md:variable-metadata-ref
   |     +--rw service-path-hop* [hop-number]
   |     |  +--rw hop-number                    uint8
   |     |  +--rw service-function-name?        string
   |     |  +--rw service-function-forwarder?   string
   |     |  +--rw service-index?                uint8
   |     +--rw service-chain-name      string
   |     +--rw starting-index?         uint8
   |     +--rw path-id?                uint32
   +--ro service-function-paths-state
      +--ro service-function-path-state* [name]
         +--ro name                          string
         +--ro sfp-rendered-service-path* [name]
            +--ro name    string
```


7.2. Service Function Path Configuration Model

```
 <CODE BEGINS> file "service-function-path@2014-07-01.yang"

 module service-function-path {

   namespace "urn:cisco:params:xml:ns:yang:sfc-sfp";

   prefix sfc-sfp;

   import ietf-inet-types { prefix inet; }
   import ietf-yang-types { prefix yang; }
   import service-function {prefix sfc-sf; }
   import service-function-metadata {prefix sfc-md; }


   organization "Cisco Systems, Inc.";
   contact "Reinaldo Penno <repenno@cisco.com>";
```

```
  description
    "This module contains a collection of YANG definitions for
     managing service function chains.

     Copyright (c) 2013 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).

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.

  revision 2014-07-01 {
    description
      "Changes based on Opendaylight Testing and IETF SFC ml.";
  }
```



```
  typedef service-function-path-ref {
    type leafref {
      path "/sfc-sfp:service-function-paths/" +
        "sfc-sfp:service-function-path/sfc-sfp:name";
    }
    description
      "This type is used by data models that need to reference
       configured service functions.";
  }

  // Service Function Path
```

```
   container service-function-paths {
     description
       "The SFP provides a level of indirection
        between the fully abstract notion of service chain as an
        abstract sequence of functions to be delivered, and the
        fully specified notion of exactly what SFF/SFs the packet
        will visit when it actually traverses the network.  By
        allowing the control components to specify the use of this
        level of indirection, the deployment may choose the degree
        of SFF/SF selection authority that is delegated to the
        network";
     list service-function-path {
       description
         "A list that holds configuration data for all SFPs in the
          domain";
       key "name";
       leaf name {
         type string;
         description
           "The name of this service function path";
       }
       leaf symmetric {
         type boolean;
         default false;
         description
           "If the chain is symmetric we will create two service
            paths, one ingress and another egress. Packets traverse
            the egress service path in the reverse order of the
            ingress path";
       }
       leaf classifier {
         type string;
         description "The classifier responsible for directing"
           + "packets to this service path";
       }
       leaf symmetric-classifier {
```



```
         type string;
         description "The classifier responsible for directing"
           + "packets to this service path";
       }
       leaf context-metadata {
```

```
      type sfc-md:context-metadata-ref;
      description
        "The name of the associated context metadata";
    }
    leaf variable-metadata {
      type sfc-md:variable-metadata-ref;
      description
        "The name of the associated variable metadata";
    }
    list service-path-hop {
      key "hop-number";
      leaf hop-number {
        type uint8;
        description
          "A Monotonically increasing number";
      }
      leaf service-function-name {
        type string;
        description
          "Service Function name";
      }
      leaf service-function-forwarder {
        type string;
        description
          "Service Function Forwarder name";
      }
      leaf service-index {
        type uint8;
        description
          "Provides location within the service path.
           Service index MUST be decremented by service functions
           or proxy nodes after performing required services.  MAY
           be used in conjunction with service path for path
           selection.  Service Index is also valuable when
           troubleshooting/reporting service paths.  In addition to
           location within a path, SI can be used for loop
           detection.";
      }
      ordered-by user;
      description
        "A list of service functions that compose the
         service path";
    }
```

```
    leaf service-chain-name {
      type string;
      mandatory true;
      description
        "The Service Function Chain used as blueprint for this
         path";
    }
    leaf starting-index {
      type uint8;
      description
        "Starting service index";
    }
    leaf path-id {
      type uint32 {
        range "0..16777216";
      }
      description
        "Identifies a service path.
         Participating nodes MUST use this identifier for path
         selection.  An administrator can use the service path
         value for reporting and troubleshooting packets along
         a specific path.";
    }
  }
}

container service-function-paths-state {
  description
    "This container hold operational state for all service
     function paths";
  config false;
  list service-function-path-state {
    description
      "This list holds operational data for all service function
       paths in the domain";
    key "name";
    leaf name {
      type string;
      description
        "The name of the service function path";
    }
    list sfp-rendered-service-path {
      key "name";
      leaf name {
        type string;
        description
          "The name of the Rendered Service Path";
      }
```

```
          description
          "A list of all rendered service paths instantiated
           from this service path";
        }
      }
    }

 }


 <CODE ENDS>
```

8. Service Function Forwarder (SFF)

This module describes the configuration a SFF needs to have in order to route packets to the service functions it serves. the SFF needs to have a table with service function name and associated locator. The locator could be an IP address and port, an internal function call or some other unique identifier.

8.1. Module Struture

```
module: service-function-forwarder
   +--rw service-function-forwarders
   |  +--rw service-function-forwarder* [name]
   |     +--rw name                              string
   |     +--rw service-node?                     string
   |     +--rw ip-mgmt-address?                  inet:ip-address
   |     +--rw rest-uri?                         inet:uri
   |     +--rw sff-data-plane-locator* [name]
   |     |  +--rw name                    string
   |     |  +--rw data-plane-locator
   |     |     +--rw (locator-type)
   |     |     |  +--:(ip)
   |     |     |  |  +--rw ip?                inet:ip-address
   |     |     |  |  +--rw port?              inet:port-number
   |     |     |  +--:(lisp)
   |     |     |  |  +--rw eid?               inet:ip-address
   |     |     |  +--:(mac)
```

```
   |     |     |  |  +--rw mac?             yang:mac-address
   |     |     |  |  +--rw vlan-id?         uint16
   |     |     |  +--:(function)
   |     |     |  |  +--rw function-name?   string
   |     |     |  +--:(mpls)
   |     |     |     +--rw mpls-label?      uint32
```



```
   |     |     +--rw transport?       identityref
   |     +--rw service-function-dictionary* [name]
   |        +--rw name                          string
   |        +--rw type?                         sfc-sft:service-function-type
   |        +--rw sff-sf-data-plane-locator
   |        |  +--rw (locator-type)
   |        |  |  +--:(ip)
   |        |  |  |  +--rw ip?              inet:ip-address
   |        |  |  |  +--rw port?            inet:port-number
   |        |  |  +--:(lisp)
   |        |  |  |  +--rw eid?             inet:ip-address
   |        |  |  +--:(mac)
   |        |  |  |  +--rw mac?             yang:mac-address
   |        |  |  |  +--rw vlan-id?         uint16
   |        |  |  +--:(function)
   |        |  |  |  +--rw function-name?   string
   |        |  |  +--:(mpls)
   |        |  |     +--rw mpls-label?      uint32
   |        |  +--rw transport?       identityref
   |        +--rw sff-interfaces* [sff-interface]
   |        |  +--rw sff-interface    string
   |        +--rw failmode?                     failmode-type
   +--ro service-function-forwarders-state
      +--ro service-function-forwarder-state* [name]
         +--ro name                string
         +--ro sff-service-path* [name]
            +--ro name    string
```

8.2. Service Function Forwarder Model

<CODE BEGINS> file "service-function-forwarder@2014-07-01.yang"

```
module service-function-forwarder {
```

```
  namespace "urn:cisco:params:xml:ns:yang:sfc-sff";

  prefix sfc-sff;

  import ietf-inet-types        { prefix inet;}
  import ietf-yang-types {
    prefix "yang";
    revision-date 2013-07-15;
  }
  import service-locator        { prefix sfc-sl; }
  import service-function-type {prefix sfc-sft;}

  organization "Cisco Systems, Inc.";
```



```
  contact "Reinaldo Penno <repenno@cisco.com>";


  description
    "This module contains a collection of YANG definitions for
     managing service function forwarders.

     Copyright (c) 2013 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).

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.

  revision 2014-07-01 {
      description
```

```
      "Revision based on Opendaylight project feedback";
  }

  // Failmode type definitions

  identity failmode-type-identity {
    description
      "Base identity from which specific failmode
       types are derived. Fail mode specifies the behavior
       when the interface does not have connectivity to the
       service node.";
  }

  typedef failmode-type {
    type identityref {
        base "failmode-type-identity";
    }
    description "This type is used to reference all
      registered failmode types";
  }
```



```
  identity close {
    base "failmode-type-identity";
    description "When service-function can not reach service
       function, packets will be dropped";
  }

  identity open {
    base "failmode-type-identity";
    description "When service-function can not reach service
       function, packets will be forwarded";
  }


  container service-function-forwarders {
    description
      "A service function forwarder is
       responsible for delivering traffic received from the SFC
       network forwarder to one or more connected service
       functions via information carried in the SFC encapsulation.
      ";
    list service-function-forwarder {
```

```
      description
        "A list that holds configuration of all SFFs in the domain";
      key "name";
      leaf name {
        type string;
        description
          "The unique name of this service function forwarder, such
           as SFF1";
      }

      leaf service-node {
        type string;
        description "The service node that hosts this SFF";
      }

      leaf ip-mgmt-address {
        type inet:ip-address;
        description
          "The IP and port used to configure this service-function-forwarder";
      }

      leaf rest-uri {
        description "URI of REST based management";
        type inet:uri;
      }

      list sff-data-plane-locator {
```



```
        description
          "A list of all data-plane-locators of this SFF.";
        key "name";
        leaf name {
          type string;
          description
            "A unique string that represents this data-plane-locator";
        }

        container data-plane-locator {
          description
            "This container holds configuration for the overlay data plane"
          + "locator used by this SFF. This could be VXLAN, GRE, etc";
          uses sfc-sl:data-plane-locator;
```

```
      }
    }

    list service-function-dictionary {
      key "name";
      leaf name {
        type string;
        description
            "The name of the service function.";
      }
      leaf type {
        type sfc-sft:service-function-type;
        description
          "Service Function Type from service-function-type yang
           model";
      }
      container sff-sf-data-plane-locator {
        description
          "The SFF uses this data plane locator when sending packets to the"
        + "associated service function";
        uses sfc-sl:data-plane-locator;
      }
      list sff-interfaces {
        key "sff-interface";
        leaf sff-interface {
          type string;
          description
            "An individual interface on the SFF connected to the
             SF";
        }
        description
          "A list of interfaces on the SFF which are connected to this SF,"
        + "usually one 1 or 2 elements";
      }
```



```
      leaf failmode {
        type failmode-type;
        description
          "This leaf defines what should the SFF do if it can not
           send packets to the SF";

      }
```

```
        description
          "A list of all Service Functions attached to this SFF.";
      }
    }
  }

  container service-function-forwarders-state {
    description
      "This container hold operational state for all service
       function forwarders";
    config false;
    list service-function-forwarder-state {
      description
        "This list holds operational data for all service functions
         forwarders in the domain";
      key "name";
      leaf name {
        type string;
        description
          "the name of the service function forwarder";
      }
      list sff-service-path {
        key "name";
        leaf name {
          type string;
          description
            "The name of the Service Path";
        }
        description
        "A list of all service function paths that contain this
         service function forwarder";
      }
    }
  }
}
```

<CODE ENDS>

9. Service Function Forwarder Open vSwitch (SFF-OVS)

This module augments the SFF model for Open vSwitch, meaning when Open vSwitche (OVS) bridges are used as SFF.

9.1. Module Structure

```
module: service-function-forwarder-ovs
augment /sfc-sff:service-function-forwarders/sfc-sff:service-function-forwarder
   +--rw ovs-bridge
      +--rw bridge-name?    string
      +--rw uuid?           yang:uuid
      +--rw external-ids* [name]
      |  +--rw name     string
      |  +--rw value?   string
      +--rw rest-uri?       inet:uri
augment /sfc-sff:service-function-forwarders/sfc-sff:service-function-forwarder
   +--rw ovs-bridge
      +--rw bridge-name?    string
      +--rw uuid?           yang:uuid
      +--rw external-ids* [name]
      |  +--rw name     string
      |  +--rw value?   string
      +--rw rest-uri?       inet:uri
```

9.2. Service Function Forwarder OVS Model

<CODE BEGINS> file "service-function-forwarder-ovs@2014-07-01.yang"

```
module service-function-forwarder-ovs {
  yang-version 1;

  namespace "urn:cisco:params:xml:ns:yang:sfc-sff-ovs";

  prefix sfc-sff-ovs;

  import service-function-forwarder {
    prefix "sfc-sff";
  }

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

```
  import ietf-yang-types {
    prefix "yang";
    revision-date 2013-07-15;
  }

  organization "Cisco Systems, Inc.";
  contact "Reinaldo Penno <repenno@cisco.com>";

  revision 2014-07-01 {
    description "Augmentation of SFF for OVS";
  }

  grouping bridge {
    container ovs-bridge {
      leaf bridge-name {
        description "Open vSwitch bridge name. In Openstack it is usually"
          + "br-tun or br-int depending if it is used in the overlay or"
          + "facing virtual machines respectively";
        type string;
      }
      leaf uuid {
        description "Open vSwitch bridge UUID";
        type yang:uuid;
      }
      list external-ids {
        description
          "The list of external ids associated with this
           bridge";
        key "name";
        leaf name {
          type string;
          description
            "A unique string that represents this external-id such
             as attached-mac";
        }
        leaf value {
          type string;
          description
            "A unique string that represents the value of the
             external-id such as fa:16:3e:56:47:10";
        }
      }
    }
```

```
  }

  augment "/sfc-sff:service-function-forwarders/"
        + "sfc-sff:service-function-forwarder/"
        + "sfc-sff:sff-data-plane-locator" {
```



```
    uses bridge;
  }

  augment "/sfc-sff:service-function-forwarders/"
        + "sfc-sff:service-function-forwarder/"
        + "sfc-sff:service-function-dictionary/"
        + "sfc-sff:sff-sf-data-plane-locator" {

    uses bridge;
  }
}
```

<CODE ENDS>

10. Service Locator (SL)

This module provides a single point of registration for all network locators types used in Services Function Chaining. the model can be augmented at will with locators appropriate for each use-case.

10.1. Module Structure

10.2. Service Locator Module

<CODE BEGINS> file "service-locator@2014-07-01.yang"

```
module service-locator {

  namespace "urn:cisco:params:xml:ns:yang:sfc-sl";

  prefix sfc-sl;

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

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

organization "Cisco Systems, Inc.";
contact "Reinaldo Penno <repenno@cisco.com>";


description
  "This module contains a collection of YANG definitions for
   managing service locators. Service locators are used as
   data plane network destinations for Service Functions and
   Service Function Forwarders
```



```
   It follows closely the constructs of
   http://tools.ietf.org/html/draft-ietf-netmod-interfaces-cfg-12

   Copyright (c) 2013 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).

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

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

   // RFC Ed.: update the date below with the date of RFC publication
   // and remove this note.

revision 2014-07-01 {
  description
  "Changes based on Opendaylight Testing.";
}

// Locator definitions
```

```
// Transport type definitions
identity transport-type {
  description
    "Base identity from which specific transport types are
     derived.";
}

identity vxlan-gpe {
  base "sl-transport-type";
  description "Programmable vxlan transport type";
}

identity gre {
  base "sl-transport-type";
  description "GRE transport type";
}

identity mpls {
  base "sl-transport-type";
```



```
  description "Multi Protocol Label Switching transport type";
}

identity other {
  base "sl-transport-type";
  description "unspecified transport type";
}

identity sl-transport-type {
  base "transport-type";
  description
    "This identity is used as a base for all transport
     types";

}

grouping lisp-locator {
  description
    "Data plane-locator: Lisp Eid(IP)";
  leaf eid {
    type inet:ip-address;
    description "Data-plane IP address";
```

```
    }
  }

  grouping ip-port-locator {
    description
      "Data plane-locator: IP address and L4 port";
    leaf ip {
      type inet:ip-address;
      description "Data-plane IP address";
    }
    leaf port {
      type inet:port-number;
      description "Data-plane port number";
    }
  }

  grouping mac-address-locator {
    description
      "Data plane-locator: mac address and optional vlan-id";
    leaf mac {
      type yang:mac-address;
      description "Data-plane MAC address";
    }
    leaf vlan-id {
      type uint16 {
        range "1..4094";
```



```
      }
      description "Data-plane VLAN ID address";
    }
  }

  grouping mpls-locator {
    description
      "Data plane-locator: MPLS";
    leaf mpls-label {
      type uint32 {
        range "1..1048575";
      }
      description "MPLS Label, 20 bits";
    }
  }
```

```
  grouping function-locator {
    description
      "When SF is co-located with SFF, this could be the name of a
       function or method.";
    leaf function-name {
      type string;
      description "Function or method name";
    }
  }

  grouping data-plane-locator {
    description
      "This group presents data-plane locator
       information for service function elements in the domain";
    choice locator-type {
      mandatory true;
      case ip {
        uses ip-port-locator;
      }
      case lisp {
        uses lisp-locator;
      }
      case mac {
        uses mac-address-locator;
      }
      case function {
        uses function-locator;
      }
      case mpls {
        uses mpls-locator;
      }
      description "The collection of all possible data-plane
```



```
         locators. Only one can be chosen";
    }
    leaf transport {
      type identityref {
        base sfc-sl:sl-transport-type;
      }
    }
  }
```

```
 }


 <CODE ENDS>
```

11. Rendered Service Path (RSP)

This module holds the actual service-hops a packet will traverse when forwarded through a specific service path.

11.1. Module Structure


```
module: rendered-service-path
```

```
   +--ro rendered-service-paths
      +--ro rendered-service-path* [name]
         +--ro name                              string
         +--ro parent-service-function-path?     string
         +--ro context-metadata?                 sfc-md:context-metadata-ref
         +--ro variable-metadata?                sfc-md:variable-metadata-ref
         +--ro rendered-service-path-hop* [hop-number]
         |  +--ro hop-number                      uint8
         |  +--ro service-function-name?          string
         |  +--ro service-function-forwarder?     string
         |  +--ro service-index?                  uint8
         +--ro service-chain-name                string
         +--ro starting-index?                   uint8
         +--ro path-id?                          uint32
rpcs:
   +---x delete-rendered-path
   |  +--ro input
   |  |  +--ro name?   string
   |  +--ro output
   |     +--ro result?   boolean
   +---x create-rendered-path
      +--ro input
      |  +--ro name?                            string
      |  +--ro parent-service-function-path?    string
      |  +--ro symmetric?                       boolean
      |  +--ro classifier?                      string
      |  +--ro symmetric-classifier?            string
      +--ro output
         +--ro result?   boolean
```

11.2. Rendered Service Path Module

```
 <CODE BEGINS> file "rendered-service-path.yang"

 module rendered-service-path {

   namespace "urn:cisco:params:xml:ns:yang:sfc-rsp";

   prefix sfc-rsp;

   import ietf-inet-types { prefix inet; }
   import ietf-yang-types { prefix yang; }
   import service-function {prefix sfc-sf; }
   import service-function-metadata {prefix sfc-md; }
   import service-function-forwarder {prefix sfc-sff; }
   import service-locator {prefix sfc-sl;}
```

```
   organization "Cisco Systems, Inc.";
   contact "Reinaldo Penno <repenno@cisco.com>";


   description
     "This module contains a collection of YANG definitions to
      manage Rendered Service Paths.

      Copyright (c) 2013 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).

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

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

      // RFC Ed.: update the date below with the date of RFC publication
      // and remove this note.

   revision 2014-07-01 {
     description
       "Changes based on Opendaylight Testing and IETF SFC ml.";
   }

   typedef rendered-service-path-ref {
     type leafref {
       path "/sfc-rsp:service-function-paths/" +
         "sfc-rsp:rendered-service-path/sfc-rsp:name";
     }
     description
       "This type is used by data models that need to reference
        rendered service paths.";
   }

   // Rendered Service Path

   container rendered-service-paths {
     config false;
```

```
     description
       "The SFP provides a level of indirection
```



```
        between the fully abstract notion of service chain as an
        abstract sequence of functions to be delivered, and the
        fully specified notion of exactly what SFF/SFs the packet
        will visit when it actually traverses the network.  By
        allowing the control components to specify the use of this
        level of indirection, the deployment may choose the degree
        of SFF/SF selection authority that is delegated to the
        network";
      list rendered-service-path {
        description
          "A list that holds operational data for all RSPs in the
           domain";
        key "name";
        leaf name {
          type string;
          description
            "The name of this rendered function path. This is the same
             name as the associated SFP";
        }
        leaf parent-service-function-path {
          type string;
          description
            "Service Function Path from which this RSP was instantiated";

        }
        leaf context-metadata {
          type sfc-md:context-metadata-ref;
          description
            "The name of the associated context metadata";
        }
        leaf variable-metadata {
          type sfc-md:variable-metadata-ref;
          description
            "The name of the associated variable metadata";
        }
        list rendered-service-path-hop {
          key "hop-number";
          leaf hop-number {
            type uint8;
```

```
          description
            "A Monotonically increasing number";
        }
        leaf service-function-name {
          type string;
          description
            "Service Function name";
        }
        leaf service-function-forwarder {
```



```
          type string;
          description
            "Service Function Forwarder name";
        }
        leaf service-function-forwarder-locator {
          type sfc-sff:service-function-forwarder-locator-ref;
          description
            "The name of the SFF data plane locator";
        }
        leaf service-index {
          type uint8;
          description
            "Provides location within the service path.
             Service index MUST be decremented by service functions
             or proxy nodes after performing required services.  MAY
             be used in conjunction with service path for path
             selection.  Service Index is also valuable when
             troubleshooting/reporting service paths.  In addition to
             location within a path, SI can be used for loop
             detection.";
        }
        ordered-by user;
        description
          "A list of service functions that compose the
           service path";
      }
      leaf service-chain-name {
        type string;
        mandatory true;
        description
          "The Service Function Chain used as blueprint for this
           path";
```

```
      }
      leaf starting-index {
        type uint8;
        description
          "Starting service index";
      }
      leaf path-id {
        type uint32 {
          range "0..16777216";
        }
        description
          "Identifies a service path.
           Participating nodes MUST use this identifier for path
           selection.  An administrator can use the service path
           value for reporting and troubleshooting packets along
           a specific path.";
```



```
      }
    }
  }

  rpc delete-rendered-path {
    description
      "Delete a Rendered Service Path";
    input {
      leaf name {
        type string;
        description
          "The name of this rendered function path. This is the same
           name as the associated SFP";
      }
    }
    output {
      leaf result {
        type boolean;
      }
    }
  }

  rpc create-rendered-path {
    description
      "Created a Rendered Service Path";
```

```
    input {
      leaf name {
        type string;
        description
          "The name of this rendered function path. This is the same
           name as the associated SFP";
      }
      leaf parent-service-function-path {
        type string;
        description
          "Service Function Path from which this RSP was instantiated";
      }
      leaf symmetric {
        type boolean;
        default false;
        description
          "If the chain is symmetric we will create two service
           paths, one ingress and another egress. Packets traverse
           the egress service path in the reverse order of the
           ingress path";
      }
      leaf classifier {
        type string;
```



```
        description "The classifier responsible for directing"
          + "packets to this service path";
      }
      leaf symmetric-classifier {
        type string;
        description "The classifier responsible for directing"
          + "packets to this service path";
      }
    }
    output {
      leaf result {
        type boolean;
      }
    }
  }

  container rendered-service-path-first-hop {
    description
```

```
      "Provides all necessary information for a system to construct
      a NSH header and associated overlay packet to target the first
      service hop of a Rendered Service Path";
    leaf starting-index {
      type uint8;
      description
          "Starting service index";
    }
    leaf symmetric-path-id {
      type uint32 {
        range "0..16777216";
      }
      description
        "Identifies the associated symmetric path, if any.";
    }
    leaf path-id {
      type uint32 {
        range "0..16777216";
      }
      description
        "Identifies a service path.
         Participating nodes MUST use this identifier for path
         selection.  An administrator can use the service path
         value for reporting and troubleshooting packets along
         a specific path.";
    }

    leaf transport-type {
      type sfc-sl:sl-transport-type-def;
    }
```



```
    uses sfc-sl:ip-port-locator;
    uses sfc-sl:mpls-locator;
  }


}
```

<CODE ENDS>

12. Service Function Classifier (SCF)

This module contains a collection of YANG definitions for managing service classification functions.

12.1. Module Structure


```
module: rendered-service-path
   +--ro rendered-service-paths
      +--ro rendered-service-path* [name]
         +--ro name                              string
```

```
         +--ro parent-service-function-path?   string
         +--ro context-metadata?               sfc-md:context-metadata-ref
         +--ro variable-metadata?              sfc-md:variable-metadata-ref
         +--ro rendered-service-path-hop* [hop-number]
         |  +--ro hop-number                    uint8
         |  +--ro service-function-name?        string
         |  +--ro service-function-forwarder?   string
         |  +--ro service-index?                uint8
         +--ro service-chain-name              string
         +--ro starting-index?                 uint8
         +--ro path-id?                        uint32
rpcs:
   +---x delete-rendered-path
   |  +--ro input
   |  |  +--ro name?   string
   |  +--ro output
   |     +--ro result?   boolean
   +---x create-rendered-path
      +--ro input
      |  +--ro name?                           string
      |  +--ro parent-service-function-path?   string
      |  +--ro symmetric?                      boolean
      |  +--ro classifier?                     string
      |  +--ro symmetric-classifier?           string
      +--ro output
         +--ro result?   boolean
```

12.2. Service Function Classifier Module

```
<CODE BEGINS> file "service-function-classifier@2014-07-01.yang"

module service-function-classifier {
  yang-version 1;

  namespace "urn:cisco:params:xml:ns:yang:sfc-scf";

  prefix sfc-scf;

  import ietf-acl {prefix "ietf-acl";}

  organization "Cisco Systems, Inc.";
  contact "Reinaldo Penno <repenno@cisco.com>";

  description
```

```
    "This module contains a collection of YANG definitions for
     managing service classification functions.";

  revision 2014-07-01 {
    description
    "Revised based on Opendaylight Project feedback";
  }

  grouping attachment-point {
    description
      "Reusable group of all possible attachment point types";
    choice attachment-point-type {
      description
        "Provides a choice between access list
         attachment point types";
      case bridge {
        leaf bridge {
          type string;
          description
            "OVS bridge as an attachment point";
        }
      }
      case interface {
        leaf interface {
          type string;
          description
            "interface name as attachment point";
        }
      }
    }
  }

  container service-function-classifiers {
    description
    "Classifier container which represents the ACL being applied,
     attachment point and the chain associated with that ACL.";
    list service-function-classifier {
      description
        "A list that holds all service function classifiers";
      key "name";
      leaf name {
        type string;
         description
         "Classification function name";
      }
      leaf access-list {
        description
```

```
          "The ACL name associated with this classifier";
```



```
        type string;
      }

      leaf rendered-service-path {
        description
           "The classifier will direct packets to the SFP
          specified here";
        type string;
      }
      list scl-service-function-forwarder {
        description
          "The classifier will be attached to these SFFs";
        key "name";
        leaf name {
          description
          "The classifier will be attached to this SFF";
          type string;
        }
        uses attachment-point {
          description "Classifier attachment point";
        }
      }
    }
  }

  container service-function-classifiers-state {
    description
      "This container hold operational state for all service
       function classifiers";
    config false;
    list service-function-classifier-state {
      description
        "This list holds operational data for all service function
         classifiers in the domain";
      key "name";
      leaf name {
        type string;
        description
          "The name of the service function classifier";
      }
```

```
      list scl-rendered-service-path {
        key "name";
        leaf name {
          type string;
          description
            "The name of the Rendered Service Path";
        }
        description
```



```
        "A list of all rendered service paths that use this classifier";
      }
    }
  }
}
```

<CODE ENDS>

13. Service Function Description Monitor Report (SF-DESC-MON-RPT)

This module for entities implementing the network service functions for Service Function Chaining

13.1. Module Structure

```
module: service-function-description-monitor-report
   +--rw service-function
      +--rw description-info
      |  +--rw number-of-dataports?   uint32
      |  +--rw capabilities
      |     +--rw supported-packet-rate?   uint32
      |     +--rw supported-bandwidth?     uint32
      |     +--rw supported-ACL-number?    uint32
      |     +--rw RIB-size?                uint32
      |     +--rw FIB-size?                uint32
      |     +--rw ports-bandwidth
      |        +--rw port-bandwidth* [port-id]
      |           +--rw port-id                uint32
      |           +--rw ipaddress?             inet:ipv4-address
      |           +--rw macaddress?            yang:mac-address
```

```
|           +--rw supported-bandwidth?   uint32
+--rw monitoring-info
   +--rw liveness?               boolean
   +--rw resource-utilization
      +--rw packet-rate-utilization?          uint32
      +--rw bandwidth-utilization?            uint32
      +--rw CPU-utilization?                  uint32
      +--rw memory-utilization?               uint32
      +--rw available-memory?                 uint32
      +--rw RIB-utilization?                  uint32
      +--rw FIB-utilization?                  uint32
      +--rw power-utilization?                uint32
      +--rw SF-ports-bandwidth-utilization
         +--rw port-bandwidth-utilization* [port-id]
            +--rw port-id                   uint32
```



```
                  +--rw bandwidth-utilization?   uint32
rpcs:
   +---x get-SF-description
   |  +--ro output
   |     +--ro description-info
   |        +--ro number-of-dataports?   uint32
   |        +--ro capabilities
   |           +--ro supported-packet-rate?   uint32
   |           +--ro supported-bandwidth?     uint32
   |           +--ro supported-ACL-number?    uint32
   |           +--ro RIB-size?                uint32
   |           +--ro FIB-size?                uint32
   |           +--ro ports-bandwidth
   |              +--ro port-bandwidth* [port-id]
   |                 +--ro port-id                uint32
   |                 +--ro ipaddress?             inet:ipv4-address
   |                 +--ro macaddress?            yang:mac-address
   |                 +--ro supported-bandwidth?   uint32
   +---x get-SF-monitoring-info
      +--ro output
         +--ro monitoring-info
            +--ro liveness?               boolean
            +--ro resource-utilization
               +--ro packet-rate-utilization?          uint32
               +--ro bandwidth-utilization?            uint32
               +--ro CPU-utilization?                  uint32
```

```
                  +--ro memory-utilization?              uint32
                  +--ro available-memory?                uint32
                  +--ro RIB-utilization?                 uint32
                  +--ro FIB-utilization?                 uint32
                  +--ro power-utilization?               uint32
                  +--ro SF-ports-bandwidth-utilization
                     +--ro port-bandwidth-utilization* [port-id]
                        +--ro port-id                  uint32
                        +--ro bandwidth-utilization?   uint32
```

13.2. Service Function Description Monitor Report Module

<CODE BEGINS> file "service-function-description-monitor-report.yang"

```
module service-function-description-monitor-report {
  namespace "urn.intel.params:xml:ns:sf-desc-mon-rpt";
  prefix sf-desc-mon-rpt;

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

  organization "Intel Inc.";
```



```
  contact "honglix.chen@intel.com";
  description
      "The module for entities implementing the network service
       functions for Service Function Chaining";

  revision 2014-11-05 {
      description "Initial revision.";
  }

  grouping SF-description {
    leaf number-of-dataports {
        type uint32;
        description "Number of dataports";
    }
    container capabilities {
      leaf supported-packet-rate {
          type uint32;
          description "Maximum Mpps supported";
      }
```

```
      leaf supported-bandwidth {
          type uint32;
          description "Maximum Mbps supported ";
      }
      leaf supported-ACL-number {
          type uint32;
          description "ACL number of the service function";
      }
      leaf RIB-size {
          type uint32;
          description "Number of Routing Information Table's entries.";
      }
      leaf FIB-size {
          type uint32;
          description "Number of Forwarding Information Table's entries.";
      }
      container ports-bandwidth {
        list port-bandwidth {
          key port-id;
          leaf port-id {
              type uint32;
              description "The id of the port ";
          }
          leaf ipaddress {
              type inet:ipv4-address;
              description "IP address of the port";
          }
          leaf macaddress {
             type yang:mac-address;
```



```
             description "MAC address of the the port";
          }
          leaf supported-bandwidth {
              type uint32;
              description "Maximum Mbps supported by the port";
          }
        }
      }
    }
  }

  grouping SF-monitoring-info {
```

```
    leaf liveness {
      type boolean;
      description "Liveness flag of the service function";
    }
    container resource-utilization {
      leaf packet-rate-utilization {
        type uint32;
        description "Percentage of current package rate utilization.";
      }
      leaf bandwidth-utilization {
        type uint32;
        description "Percentage of bandwidth utilization.";
      }
      leaf CPU-utilization {
        type uint32;
        description "Percentage of CPU utilization.";
      }
      leaf memory-utilization {
        type uint32;
        description "Percentage of memory utilization.";
      }
      leaf available-memory {
        type uint32;
        description "Available memory size of the service function in MB.";
      }
      leaf RIB-utilization {
        type uint32;
        description "Percentage of Routing Information Table utilization.";
      }
      leaf FIB-utilization {
        type uint32;
        description "Percentage of Forwarding Information Table utilization.";
      }
      leaf power-utilization {
        type uint32;
        description "power utilization in W.";
```



```
      }
      container SF-ports-bandwidth-utilization {
          list port-bandwidth-utilization {
              key port-id;
              leaf port-id {
```

```
                type uint32;
                description "The id of the port ";
             }
             leaf bandwidth-utilization {
                type uint32;
                description "Percentage of the port's supported bandwidth util
            }
         }
      }
    }
  }

  container service-function {
    container description-info {
        uses SF-description;
    }
    container monitoring-info {
        uses SF-monitoring-info;
    }
  }

  rpc get-SF-description {
    description
        "Get service function description information.";
  output {
      container description-info {
          uses SF-description;
      }
    }
  }

  rpc get-SF-monitoring-info {
    description
        "Get current service function monitoring information.";
    output {
      container monitoring-info {
          uses SF-monitoring-info;
      }
    }
  }
}
```

<CODE ENDS>

14. Service Function Description Monitor (SF-DESC-MON)

This module is used to create description and monitoring information of Service Function extensions to service-function model

14.1. Module Structure

```
module service-function-description-monitor {
  namespace "urn.intel.params:xml:ns:sf-desc-mon";
  prefix sf-desc-mon;

  import service-function { prefix sfc-sf;}
  import service-function-description-monitor-report { prefix sf-desc-mon-rpt;}

  organization "Intel Inc.";
  contact "honglix.chen@intel.com";
  description
      "Create description and monitoring information of Service
       Function extensions to service-function model";

  revision 2014-12-01 {
      description "Initial revision.";
  }

  augment "/sfc-sf:service-functions-state/sfc-sf:service-function-state" {
    description "Service function description and monitoring information";
    container sfc-sf-desc-mon {
      container description-info {
        uses sf-desc-mon-rpt:SF-description;
      }
      container monitoring-info {
        uses sf-desc-mon-rpt:SF-monitoring-info;
      }
    }
  }
}
```

14.2. Service Function Description Monitor Report Module

<CODE BEGINS> file "service-function-description-monitor.yang"

```
module service-function-description-monitor {
  namespace "urn.intel.params:xml:ns:sf-desc-mon";
  prefix sf-desc-mon;

  import service-function { prefix sfc-sf;}
  import service-function-description-monitor-report { prefix sf-desc-mon-rpt;}

  organization "Intel Inc.";
  contact "honglix.chen@intel.com";
  description
      "Create description and monitoring information of Service
       Function extensions to service-function model";

  revision 2014-12-01 {
      description "Initial revision.";
  }

  augment "/sfc-sf:service-functions-state/sfc-sf:service-function-state" {
    description "Service function description and monitoring information";
    container sfc-sf-desc-mon {
      container description-info {
        uses sf-desc-mon-rpt:SF-description;
      }
      container monitoring-info {
        uses sf-desc-mon-rpt:SF-monitoring-info;
      }
    }
  }
}
```

<CODE ENDS>

15. IANA Considerations

TBD

16. Security Considerations

17. Acknowledgements

Thanks to Jan Medved, Ron Parker, Jan Lindblad, David Goldberg, Vina Ermagan, Sam Hague and Vinayak Joshi and for reviews and suggestions.

18. Changes

-11

- Added new co-authors
- changed RSP and SFP models to allow multiple encap paths.
- Added the "need reclassififcation" leaf to allow a Service Function to tell a SFF that packet reclassification is needed
- Added RSP first hop container to allow SFC applications to request the ingress hop to a RSP. These applciations (such as dynamic classifier) treat RSP as a black box and only need ingress locator.

-10

This new revision comes after considerable control and dataplane interop testing. The new changes reflect what we found necessary for building a well-rounded solution.

- Added Rendered Service Path Model
- Added Service Function Description Monitor and Report Models
- Updated Service Function Path (It allow users to control certain aspects of RSPs, new symmetric and metadata leaves)
- Updated Service Function Forwarder model
- Updated Service Function Model
- Updated Service Function Type Model (HTTP Header Enrichment service)

- Update Service Locator Model (MPLS encap)
- Removed Service Node
- Others

-09

- Modifed Service Function Forwarder OVS model based on OVS/ Openstack deployment experience

-08

- Removed VXLAN-GPE model
- Added Service Function Forwarder OVS model
- Added metadata reference to Service Function Path

-07

- All models that need data plane locators reference service locator model
- Service Locator module has locators for IP:port, VLAN:MAC, LISP
- A SF can have multiple data place locators
- SF and SFF are decoupled and have their own views of the network
- Service Function Chains and derived path can be symmtric (bi-dir) or not
- Service Function Types separated into a model
- Service Function Path is a collection of service hops. This allows hops such as SFF + classifier.

-06

- Introduced operational tree in some models based on testing and

user feedback.

- Introduced RPCs in some models
- Service Function Path needs SFC from which it will be instantiated
- Updated all module structures
- Introduced Service Locator module

-05

Changes based on Opendaylight Implementation Testing and Sfc-dev mailing list feedback

- Service Node becomes a container for Service Functions. Moved data plane items to SFF.
- Fixed Service Function Forwarders into a list so we cna have multiple in a system

- Fixed Service Function Chain so it becomes a list of lists.
- Created RPCs for Service Functions and Service Chain

-04

- Fixed list inside Service Function Chain to read service-function-type
- Small comment fixes

-03

- Revision dates consistent
- Service function chain to container + list in order to allow multiple
- Service Function Path to cotainer + list
- VXLAN-gpe vni to multiple 8-bit fields

- Consistent typeref use
- Other consistency fixes

-02

- After Opendaylight Testing converted multiple leafs to lists throughout all models
- Removed transport dependency. Transport could be layer-2, layer-3, etc
- Used pathrefs similar to ietf-interfaces to reference configuration names
- Other consistency fixes

19. References

19.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

19.2. Informative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-05 (work in progress), February 2015.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13

(work in progress), February 2015.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Surendra, S., Smith, M., Henderickx, W., Nadeau, T., Agarwal, P., Manur, R., Chauhan, A., Halpern, J., Majee, S., Elzur, U., Melman, D., Garg, P., McConnell, B., Wright, C., and K. Kevin, "Network Service Header", draft-quinn-sfc-nsh-07 (work in progress), February 2015.

[I-D.quinn-vxlan-gpe]
Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X., Elzur, U., Garg, P., and D. Melman, "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-04 (work in progress), February 2015.

Authors' Addresses

Reinaldo Penno
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: repenno@cisco.com

Paul Quinn
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: paulq@cisco.com

Danny Zhou
Intel Corporation
2200 Mission College Blvd.
Santa Clara CA
USA

Email: danny.zhou@intel.com

Johnson Li
Intel Corporation
2200 Mission College Blvd.
Santa Clara CA
USA

Email: johnson.li@intel.com