SFC Netmod
Internet-Draft
Intended status: Standards Track
Expires: January 22, 2015

R. Penno
P. Quinn
Cisco Systems

July 21, 2014

Yang Data Model for Service Function Chaining
draft-penno-sfc-yang-06

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 January 22, 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.quinn-sfc-arch], [I-D.ietf-sfc-problem-statement] ,[I-D.quinn-sfc-nsh] and [I-D.quinn-vxlan-gpe]

3. VXLAN-GPE

This model describes the VXLAN-GPE encapsulation when used as a overlay mechanism to create service function paths. VXLAN is one of many transport protocols that can be used to setup service chaining overlays.

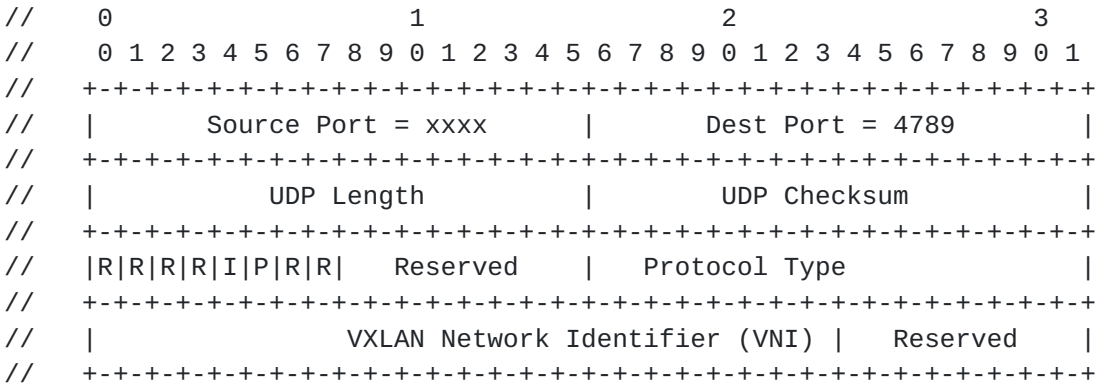
3.1. Module Structure

```
module: vxlan-gpe
   +--rw vxlan-gpe-header
      +--rw gpe-header-flag-value?   vxlan-gpw-header-flag-type
      +--rw reserved?                uint8
      +--rw protocol-type?           uint16
      +--rw vni*                     uint8
      +--rw reserved2?               uint8
```

3.2. VXLAN -GPE Configuration Model

```
<CODE BEGINS> file "vxlan-gpe@2013-12-04.yang"


module vxlan-gpe {

  namespace "urn:cisco:params:xml:ns:yang:vxlan-gpe";

  prefix vxlan-gpe;

  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 chains.
```

```
    // 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 2013-11-26 {
    description
    "Initial revision.";
  }
```




```
  typedef vxlan-gpw-header-flag-type {
    type bits {
      bit r1 {
        position 0;
        description "reserved";
      }
      bit r2 {
        position 1;
        description "reserved";
      }
      bit r3 {
```

```
        position 2;
        description "reserved";
      }
      bit r4 {
        position 3;
        description "reserved";
      }
      bit i {
        position 5;
        description "Some description";
      }
      bit p {
        position 6;
        description "Some description";
      }
      bit r7 {
        position 7;
        description "reserved";
      }
      bit r8 {
        position 8;
        description "reserved";
      }
    }
    description "vxlan-gpe Header Flags";
    reference "http://tools.ietf.org/html/draft-quinn-vxlan-gpe-01";
  }

  container vxlan-gpe-header {
    description "Network Service Base header";

    leaf gpe-header-flag-value {
        type vxlan-gpw-header-flag-type;
    }

    leaf reserved {
      default 0;
      type uint8;
    }

    leaf protocol-type {
      type uint16;
      // Reinaldo: Another option is to import Opendaylight L2 Types so have
ethertype
    }

    leaf vni-low {
      type uint8;
```

```
    }
```

```
    leaf vni-med {
      type uint8;
    }

    leaf vni-hi {
      type uint8;
    }

    leaf reserved2 {
      default 0;
      type uint8 {
        range "0 .. 255";
      }
      description "Reserved field";
    }
  }
}

</CODE ENDS>
```

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                          string
      |     +--rw ip-mgmt-address?              inet:ip-address
      |     +--rw sf-data-plane-locator
      |     |  +--rw (locator-type)
      |     |     +--:(ip)
      |     |        +--rw ip?     inet:ip-address
      |     |        +--rw port?   inet:port-number
      |     +--rw service-function-forwarder    string
      +--ro service-functions-state
         +--ro service-function-state* [name]
            +--ro name                        string
            +--ro sf-service-function-path*   string
   rpcs:
      +---x delete-all-service-function
      +---x put-service-function
      |  +--ro input
      |     +--ro name?                         string
      |     +--ro type                          string
      |     +--ro ip-mgmt-address?              inet:ip-address
      |     +--ro sf-data-plane-locator
      |     |  +--ro (locator-type)
      |     |     +--:(ip)
      |     |        +--ro ip?     inet:ip-address
      |     |        +--ro port?   inet:port-number
      |     +--ro service-function-forwarder    string
      +---x read-service-function
      |  +--ro input
      |  |  +--ro name    string
      |  +--ro output
      |     +--ro name?                         string
      |     +--ro type                          string
      |     +--ro ip-mgmt-address?              inet:ip-address
      |     +--ro sf-data-plane-locator
      |     |  +--ro (locator-type)
      |     |     +--:(ip)
      |     |        +--ro ip?     inet:ip-address
      |     |        +--ro port?   inet:port-number
      |     +--ro service-function-forwarder    string
      +---x delete-service-function
         +--ro input
            +--ro name    string
```

4.2. Service Function

```
<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 {
    leaf name {
      type string;
      description
          "The name of the service function.";
    }
    leaf type {
      type string;
      mandatory true;
      description
        "Service Function Type from service-function-type.yang.";
    }
    leaf ip-mgmt-address {
      type inet:ip-address;
    }
    container sf-data-plane-locator {
      uses sfc-sl:data-plane-locator;
    }

    leaf service-function-forwarder {
      type string;
      mandatory true;
      description
        "The service function forwarders associated with this Service
Function";
    }
  }

  container service-functions {
    description
      "A network or application based packet
      treatment, application, compute or storage resource, used
      singularly or in concert with other service functions within a
      service chain to enable a service offered by an operator.

      A non-exhaustive list of Service Functions includes: firewalls,
```

WAN and application acceleration, Deep Packet Inspection (DPI),

```
      server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HOST_ID
      injection, HTTP Header Enrichment functions, TCP optimizer, etc.";

    list service-function {
      key "name";
      uses service-function-entry;
    }
  }

  container service-functions-state {
    config false;
    list service-function-state {
      description
        "A service chain defines the required functions and
        associated order (service-function1 --> service-function 2) that
        must be applied to packets and/or frames.  A service chain does
        not specify the network location or specific instance of service
        functions (e.g. firewall1 vs. firewall2).";
      key "name";
      leaf name {
        type string;
        description
          "the name of the service function";
      }
      leaf-list sf-service-function-path {
        type string;
        description
        "A list of all service function paths that contain this service
function";
      }
    }
  }

  rpc delete-all-service-function {
  }

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

</CODE ENDS>

5. 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 (firewal1 vs. firewall2) will be used to actually process packets.

5.1. Module Structure

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

5.2. 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; }


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


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

```
    // 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 service chain defines the required functions and
        associated order (service-function1 --> service-function 2) that
        must be applied to packets and/or frames.  A service chain does
        not specify the network location or specific instance of service
        functions (e.g. firewall1 vs. firewall2).";
      key "name";
      leaf name {
        type string;
        description
          "the name of the service function chain";
      }
      list sfc-service-function {
        key "name";
        leaf name {
          type string;
          description
            "The name of the service function if known,
            otherwise a generic unique string";
        }
        leaf type {
          type string;
```

```
          mandatory true;
          description
            "Service Function Type from service-function-type.yang.";
        }
        ordered-by user;
        description
          "A list of service functions that compose the service chain";
      }
    }
  }

  // Service Function Chains

  container service-function-chains {
    uses service-function-chain-grouping;
  }

  container service-function-chains-state {
    config false;
    list service-function-chain-state {
      description
        "A service chain defines the required functions and
        associated order (service-function1 --> service-function 2) that
        must be applied to packets and/or frames.  A service chain does
        not specify the network location or specific instance of service
        functions (e.g. firewall1 vs. firewall2).";
      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";
      }
    }
  }

    // Remote procedure calls

    // (main feature: instantiation of a SFC)

    rpc instantiate-service-function-chain {
        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 {
    input {
      uses service-function-chain-grouping;
    }
  }
}
```

</CODE ENDS>

6. Service Node (SN)

A Service Node is a virtual or physical element that houses one or more service functions. A Service node might contain an entire service function chain or be part of a larger service function chain.

6.1. Module Structure

```
module: service-node
   +--rw service-nodes
      +--rw service-node* [name]
         +--rw name                string
         +--rw ip-mgmt-address?    inet:ip-address
         +--rw service-function*   sfc-sf:service-function-ref
rpcs:
   +---x put-service-node
      +--ro input
         +--ro name?               string
         +--ro ip-mgmt-address?    inet:ip-address
         +--ro service-function*   sfc-sf:service-function-ref
```

6.2. Service Node Configuration Model

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

```
module service-node {

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

  prefix sfc-sn;

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



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

```
    // Service Nodes

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


  grouping service-node-grouping {
      leaf name {
        type string;
        description
             "The name of the service node.";
      }

      leaf ip-mgmt-address {
        type inet:ip-address;
      }
      leaf-list service-function {
        type sfc-sf:service-function-ref;
        description
            "A list of service functions resident in this service node";
      }
  }

  container service-nodes {
    description
        "Physical or virtual element that hosts one or
        more service functions and has one or more network locators
        associated with it for reachability and service delivery.";
    list service-node {
      key "name";
      uses service-node-grouping;
    }
  }

    // Remote procedure calls (for testing)

    rpc put-service-node {
        input {
            uses service-node-grouping;
    }
  }
}
```

</CODE ENDS>

7. Service Function Path (SFP)

A Service Function Path is an instantiation of a service function chain. It specifies the actual service functions (e.g. firewall1) and the transport encapsulation used in the overlay.

7.1. Module Structure

```
module: service-function-path
   +--rw service-function-paths
      +--rw service-function-path* [name]
         +--rw name                    string
         +--rw sfp-service-function* [name]
         |  +--rw name                          string
         |  +--rw service-function-forwarder?   string
         +--rw service-chain-name      string
         +--rw service_index?          uint8
         +--rw path-id?                uint32
         +--rw path-id-low?            uint8
         +--rw path-id-med?            uint8
         +--rw path-id-hi?             uint8
```

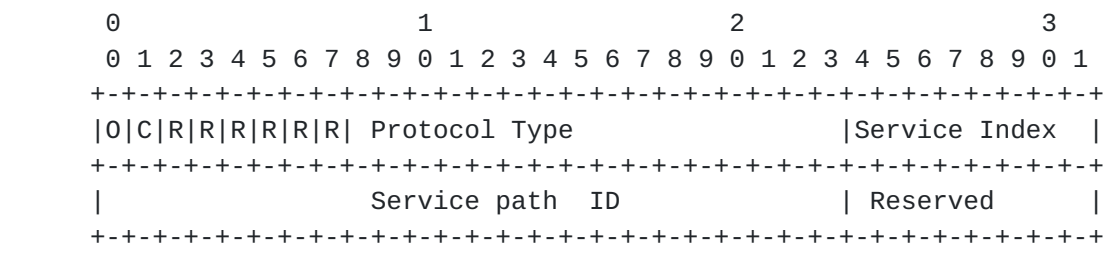
7.2. Service Function Path Configuration Model

```
<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; }


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


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

```
    // 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 {
    list service-function-path {
      description
        "A Service Function Path is an instantiation of a Service Chain. It
         specifies the actual firewall (say, firewall-3) that will be traversed
by
         the packets. The Service Path needs to be known before hand or
stitched
         run-time (given the dynamic LB decision) since a forwarding decision
need
         to be made regardless.";
      key "name";
      leaf name {
```

```
        type string;
        description
          "the name of this service function path";
```

```
      }
      list sfp-service-function {
        key "name";
        leaf name {
          type string;
          description
            "Service Function name";
        }
        leaf service-function-forwarder {
          type string;
          description
            "Service Function Forwarder name";
        }
        ordered-by user;
        min-elements 1;
        description
          "A list of service function instances that compose the service path";
      }
      leaf service-chain-name {
        mandatory true;
        type string;
        description
          " The Service Function Chain used as blueprint for this path";
      }
      leaf service_index {
        type uint8;
      }
      leaf path-id {
        type uint32 {
          range "0..16777216";
        }
      }
      leaf path-id-low {
        type uint8;
      }

      leaf path-id-med {
        type uint8;
      }

      leaf path-id-hi {
        type uint8;
      }
    }
  }

/*
   Base Service Header:
```



Flags: 8
Protocol Type (PT): 16
Service Index: 8
Service path: 24
Reserved: 8
*/

}

<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 transport?                        transport-type
            +--rw path-id?                          uint32
            +--rw sff-data-plane-locator
            |  +--rw (locator-type)
            |     +--:(ip)
            |        +--rw ip?     inet:ip-address
            |        +--rw port?   inet:port-number
            +--rw service-function-dictionary* [name]
               +--rw name                             string
               +--rw type                             string
               +--rw ip-mgmt-address?                 inet:ip-address
               +--rw sf-data-plane-locator
               |  +--rw (locator-type)
               |     +--:(ip)
               |        +--rw ip?     inet:ip-address
               |        +--rw port?   inet:port-number
               +--rw service-function-forwarder    string
               +--rw failmode?                     failmode-type
```

8.2. Service Function Forwarder Model

<CODE BEGINS> file "service-function-forwarder@2014-06-05.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; }
  import service-function       { prefix sfc-sf; }
  import service-function-path  { prefix sfc-sfp; }
  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 forwarders.
```

```
    // 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";
  }

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

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

  typedef transport-type {
    type identityref {
        base "transport-type-identity";
    }
  }

  // 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";
    }
  }

  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";
  }


  // Service Function Forwarding Map

  container service-function-forwarders {
    description
        "This container holds the configuration for a service function
forwarder. ";
    list service-function-forwarder {
      key "name";
      leaf name {
        type string;
        description
          "The name of this service function forwarder, such as ovs-switch-1";
      }
      leaf transport {
        type transport-type;
      }

      leaf path-id {
        type uint32 {
          range "0..16777216";
        }
      }

      container sff-data-plane-locator {
        uses sfc-sl:data-plane-locator;
        description
```

```
            "The data plane network locator of this SFF, such as IP:port.";
        }
```

```
      list service-function-dictionary {
        key "name";
        uses sfc-sf:service-function-entry;
        leaf failmode {
            type failmode-type;
        }
      }
    }
  }
}
```

<CODE ENDS>

9. Service Function Locator

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.

9.1. Service Locator Module

```
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.
```

```
    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

  grouping ip-port-locator {
    description
      "Data plane-locator. The Service Function Forwarder uses the network
locator
      to send packets to this Service Function.";
    leaf ip {
      type inet:ip-address;
    }
    leaf port {
      type inet:port-number;
    }
  }

  grouping data-plane-locator {
    choice locator-type {
      mandatory true;
      case ip {
        uses ip-port-locator;
      }
    }
  }
}
```

10. IANA Considerations

TBD

11. Security Considerations

12. Acknowledgements

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

13. Changes

-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

14. References

14.1. 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.

14.2. Informative References

[I-D.ietf-sfc-problem-statement] Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-07 (work in progress), June 2014.

[I-D.quinn-sfc-arch]
Quinn, P. and J. Halpern, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-05 (work in progress), May 2014.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Fernando, R., Surendra, S., Smith, M., Yadav, N., Agarwal, P., Manur, R., Chauhan, A., Elzur, U., Garg, P., McConnell, B., and C. Wright, "Network Service Header", draft-quinn-sfc-nsh-03 (work in progress), July 2014.

[I-D.quinn-vxlan-gpe]
Quinn, P., Agarwal, P., Fernando, R., Lewis, D., Kreeger, L., Smith, M., Yadav, N., Yong, L., Xu, X., Elzur, U., and P. Garg, "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-03 (work in progress), July 2014.

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