I2RS
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

Z. Wang
S. Hares
Huawei
J. Tantsura
R. White
Ericsson
Q. Wu
L. Dunbar
Huawei

March 8, 2015

A YANG Data Model for Protocol Independent Service Topology: SFF Forwarder
draft-wang-i2rs-yang-sff-dm-00

Abstract

This document defines a YANG data model for Service Function Forward Topology. This I2RS yang data model is part of the I2RS protocol independent topology set of data models.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

An overlay network consists of tunnels established among designated nodes to traverse segments of networks.

This draft describes a protocol independent topology of service function forwarder nodes which augments the [I-D.clemm-i2rs-yang-network-topo] model as a specific service topology (SFF). Figure 1 shows how the SFF is an extension of the service forwarded nodes.

```
                 +---------------------------------------+
            /            _[X1]_          "Service"  /
           /           _/  :   \_        (SFF)     /
          /          _/     :    \_               /
         /         _/        :     \_            /
        /         /           :      \          /
       /       [X2]__________________[X3]      /
      +---------:--------------:------:-------+
                 :              :     :
             +----:--------------:----:--------------+
            /      :              :   :        "L3" /
           /        :              :  :            /
          /         :               : :           /
         /         [Y1]_____________[Y2] tunnels /
        /           *               * *         /
       /            *              *  *        /
      +--------------*-------------*--*-------+
                      *           *   *
             +--------*----------*----*--------------+
            /     [Z1]_______________[Z1] "Optical" /
           /         \_         *   _/             /
          /            \_      *  _/              /
         /               \_   * _/               /
        /                  \ * /                /
       /                    [Z]                /
      +---------------------------------------+
```

Figure 1

There can be many types of protocol independent service topologies such as: L2VPN, L3VPN, MPLS, EVPN, and others. The Service Function Chaining services consists of a topology of Service Function Forwarder nodes connected by links which are tunnels that connect the service nodes. Each Service Forwarder node has service functions attached to the Service Function Forwarder node.

The SFF topology is built on top of one or several underlying networks (see figure 1). In case multi-tenancy is needed, multiple SFF topologies can be built on top of the same underlying network. Each tenant can only see its own service topology. But all the tenant’s service topology can be mapped into the same L3 network topology.

The I2RS protocol independent topologies are abstractions created by the I2RS Client directly or by instructions to I2RS agent to import network topologies or aggregations of the network topology. The I2RS protocol independent L3 topology is created by the client or the clients instruction to import specific information from the I2RS Agent from static configuration or IGPs (E.g. OSPF or ISIS) or information passed in EGPs (e.g. [I-D.ietf-idr-ls-distribution]. Similarly, the protocol independent SFF topology is abstraction of network topology information. Since SFF has no another control plane protocol running on top of the underlying networks, this information will need to be gathered from other sources.

This document defines a Yang data model for the SFF protocol independent topology. This draft will need to be harmonized with the configuration and status yang modules from SFC WG. This draft has been reviewed against the following drafts: [I-D.penno-sfc-yang] and [I-D.xia-sfc-yang-oam].

2. Definitions and Acronyms

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

NETCONF: Network Configuration Protocol.

URI: Uniform Resource Identifier.

YANG: A data definition language for NETCONF.

Classification: Locally instantiated policy and customer/network/ service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

Classifier: An element that performs Classification.

Service Function Chain (SFC): A service function chain defines a set of abstract service functions and ordering constraints that must be applied to packets and/or frames selected as a result of classification.

Service Function (SF): A function that is responsible for specific treatment of received packets.

Service Function Forwarder (SFF): A service function forwarder is responsible for delivering traffic received from the network to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

Metadata: provides the ability to exchange context information between classifiers and SFs and among SFs.

Service Function Path (SFP): The SFP provides a level of indirection between the fully abstract notion of service chain as a sequence of abstract service functions to be delivered, and the fully specified notion of exactly which SFF/SFs the packet will visit when it actually traverses the network.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. SFF Topology Data Model

This section describe the architecture and the tree diagram of the service topology yang data model.

3.1. Model Overview

The abstract Topology yang Model contain a set of abstract nodes and a list of abstract links. Service Function Chain Topo yang model and other service topo model can be augumented from the abstract topology model with SFC base topology specifics.

The following Figure depicts the relationship of service topology yang model to the abstract topology yang model.

```
                   +-----------------+
                   |    Abstract     |
                   | Topology Model  |
                   |                 |
                   +--------|--------+
                            |
                  +-----------------------+
                  |          .....         |
                  |
           +------V----------+
           |     Service     |
           | Topology models |
           |                 |
           +--------|---------
                            |
           +----|------------------------|---+
                |                        |
       +--------V--------+      +--------V-----------+
       |Service Function |      | Another protocol   |
       |   Forwarder     |      | Independent Service|
       | Topology Models |      | Topology Models    |
       +-----------------+      +--------------------+
```

Figure 2

The relationship of service topology yang model to the abstract topology yang model

The following is the generic topology module

```
          module: network
   +--rw network* [network-id]
      +--rw network-id            network-id
      +--ro server-provided?      boolean
      +--rw network-types
      +--rw supporting-network* [network-ref]
      |  +--rw network-ref    leafref
      +--rw node* [node-id]
         +--rw node-id            node-id
         +--rw supporting-node* [network-ref node-ref]
            +--rw network-ref    leafref
            +--rw node-ref       leafref
```

The service modules augments the network types and this data structures. To provide context for this model, this sample augment for the types is provided (but not normative for this draft).

```
module: Service Topologies
augment /nt:network-topology/nt:topology/nt:topology-types
  +--rw Service-Topologies
     +--rw SFF-topology
     +--rw L3VPN-Service-topology
     +--rw EVPN-Service-topology
```

Figure 3: The structure of the abstract (base) network model

3.2. SFF Topology Yang

The following figure provide the structure of service topology yang model. Each node is printed as:

```
   <status> <flags> <name> <opts> <type>

        <status> is one of:
          +  for current
          x  for deprecated
          o  for obsolete

        <flags> is one of:
          rw  for configuration data
          ro  for non-configuration data
          -x  for rpcs
          -n  for notification

        <name> is the name of the node
         If the node is augmented into the tree from another module, its
      name is printed as <prefix>:<name>.

      <opts> is one of:
          ?  for an optional leaf or choice
          !  for a presence container
          *  for a leaf-list or list
          [<keys>] for a list's keys

      <type> is the name of the type for leafs and leaf-lists
```

Figure 4

3.3. SFF topology Model Description

SFF Topology Module

```
      module: SFF topology
      augment /nt:network-topology/nt:topology/nt:topology-types
        +--rw Service-Topologies
               +--SFF Topology!
           augment /nt:network-topology/nt:topology
        +--rw service-topo-id     network-id
        +--rw service-topology-attributes
           +--rw node-count          uint32      ! augments model
           +--rw topology-extension!
       augment /nt:network-topology/nt:topology/nt:node
            +--rw node-type!
           +--rw classifier-node?  string
           +--rw sf-node?        string
           +--rw sff-node?       string
        +--rw next-hop*[hop-id]
           +--hop-id           node-id
        +--rw node-extension!
           +--rw classifier-extension!
                       +--rw classifier-id  node-id
               +--rw sfc-policy    uint32
                  +--rw sfp!
                  +--rw sfp-id    uint32
                  +--rw sf-list*[sf-id]
                  +--rw sf-id  node-id
                  +--rw sff-list*[sff-id]
                  +--rw sff-id  node-id
           +--rw sf-node-extension!
                      +--rw sf-node-extension!
                 +--rw sf-id  node-id
                 +--rw sf-node-locator  uint32
                 +--rw sf-type?   sft:service-function-type
              +--rw sf-inventory-data!
           +--rw sff-node-extension!
                      +--rw sff-id   node-id
              +--rw (sffn-address)?
                 +--:(ipv4-address)
                 |  +--rw ipv4-address?  inet:ipv4-address
                 +--:(ipv6-address)
                 | +--rw ipv6-address?     inet:ipv6-address
                 +--rw sffn-virtual-context!
                 |  +--rw context-id          uint32
                 +--rw Attached-service-address!
                 |  +--rw service-node*[service-node-id]
                 |  |  +--rw service-node-id  node-id
                 |  +--rw host-system*[host-system-id]
                    |     +--rw host-system-id   uint32
                 +--rw customer-support*[customer-id]
                 |  +--rw customer-id  uint32
                 +--rw customer-service-resource*[customer-resource-id]
                 |  +--rw customer-resource-id  node-id
                 +--rw sffn-vntopo!
```

Figure 5

The service topo yang model contains a service-topology structure. Based on the generic topology model, this can be a list.

topology model

The generic model contains a topology leaf. The SFF augments the topology types leaf within this topology life with the SFF-topology type. The SFF module also augments the topology with topology-id leaf, and a topology attributes leaf that contains node count leaf and topology-extension container. The node-count leaf can be used to indicate the number of nodes which contained in the service-topology list. The topology-extension container can be used to augment the service topology model by topology specifics.

node structure

The generic topology structure also contains a node (nt:node), and this structure has been augmented by containers for node type, a next-hop container, and a node-extensions. The node-type container can used to indicate the type node, such classifier, a sf or a sff. The node-extension container can be used to augment the node list by node specifics, for example: classifier extension, sf extension, sff extension.

link structure

The generic link topology structure contains a link (nt:link) structure, and this generic link structure has been augmented to include a sff-link-type leaf, sff-direction container, and an segment-extension leaf. The segment-extension container can be used to augment the segment list by segment specifics. Such as netconf segment extension, i2rs segment extension.

classifier extension

In SFC, the classifier is used to locally instantiated policy and customer/network/service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

sf-node-extension

The sf is a function that is responsible for specific treatment of received packets. As a logical component.

sff-node-extension

The service function forwarder is responsible for delivering traffic received from the network to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

3.4. Comparison with SFC WG Yang Modules

The following entities have modules in [I-D.penno-sfc-yang]

- classifier-extensions - may link to Service Function Classifier (SCF), no equivalent policy structures exist in that module.

   o sff-node-extensions - may link to Service Function Forwarder
      (SFF), but it is difficult to determine if the sff-data-plane-
      locator or the ip-mgt-address could be used as the virtual
      address. All portions of this structure do not exist in the
      service functgion forwarder configuration and status structure.

   o service-function-type (SFT) has been utilized as a definition in
      the sf-node extension structure.

No appropriate entities were found in the SFC OAM draft [I-D.xia-sfc-yang-oam].

4. Service Topology YANG Module

```
<CODE BEGINS> file "service-topo.yang"
module service-topo {
     yang-version 1;
     namespace "urn:TBD:params:xml:ns:yang:service-topo";
     prefix stopo;

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

         import ietf-service-function-type {
           prefix sft;
         ;

     organization "IETF I2RS Working Group";
     contact
       "wangzitao@huawei.com";
            "shares@ndzh.com";

     description
       "This module defines service topology yang data model";
     typedef node-id {
       type inet:uri;
     }
     typedef network-id {
       type inet:uri;
     }
     typedef link-id {
       type inet:uri;
     }
container service-topologies {
```

```
description
"Contains configuration related data. Within the container
is list of service topologies.";

list service-topology{
key "service-topo-id";
description
"Define the list of service-topology within the network";

leaf service-topo-id{
type network-id;
 description
"the identifier for a service topology";}

leaf node-count{
type uint32;
description
"Number of nodes within a service topology.";}

container topology-type{
  description
 " This container contains several leaf to specify the topology type,
   such as NETCONF or I2RS. A service topology can even have
   multiple types simultaneously. And this container can
   be used to augment the service topology model by topology specifics.";}

container topology-extension{
description
 " The topology-extension container can be used to augment
the service topology model by topology specifics.";}

list underlay-topologies {
key "topology-id";
   description
"Define the list of underlay-topologies within the service-topology list";

leaf topology-id {
 type network-id;
   description
 "It is presumed that a datastore will contain many topologies. To
  distinguish between topologies it is vital to have UNIQUE
  topology identifiers.";}
}

list node {
key "node-id";
description
"Define the list of node within the service-topology list";
```

```
leaf node-id {
type node-id;
description
"The identifier of a node in the service topology.";}

list termination-point{
key "termination-point-id";
description
"Define the list of termination-point within the node list";

leaf termination-point-id {
type node-id;
description
"The identifier of the termination point of the node";}

list support-termination-points{
key "support-point-id";
description
"Define the list of support-point-id within the termination-point list";

leaf support-point-id {
type node-id;
description
"the identifier of the support node of the termination point";}
}

container termination-point-extension{
description
"The termination-point-extension container can be used to augment
the service topology model by topology specifics.";}
}

container node-type{
description
"The node-type container can used to indicate the type node,
such classifier, a sf or a sff. And this container can be used
to augment the service topology model by topology specifics. ";

leaf classifier-node {
type string;
description
"To indicate the node is a classifier";}

leaf sf-node {
type string;
description
"To indicate the node is a service function(sf)";}
leaf sff-node {
```

```
type string;
description
" To indicate the node is a service function forward(sff)";}
}

list next-hop{
key "hop-id";
description
"Define the list of next-hop within the node list";

leaf hop-id {
type node-id;
description
"The identifier of the next hop of the node";}
}

container node-extension{
description
" The node-extension container can be used to augment
the service topology model by topology specific nodes.";

container classifier-extension {
description
"The classifier-extension container contains the extensions
of the classifier";}

container sf-node-extension {
description
"The sf-extension container contains the extensions of the sf.";}

container sff-node-extension {
description
" The sff-extension container contains the extensions of the sff. ";}
}
} //end the node list

list segment{
key "segment-id";
description
"Define the list of segment within the service-topology list.";

leaf segment-id{
type link-id;
description
" The segment-id leaf can be used to distinguish
the different service segment.";}

container source{
```

```
description
"The source container contains the source node
and the termination point reference list.";

leaf source-id{
type node-id;
description
"The identifier of the source node of the segment.";}
list termination-point-reference{

key "source-support-id";
leaf source-support-id{
type node-id;
description
"The identifier of the termination point of the source node.";}
}//end the termination-point-reference list
}//end the source container

container destination{
description
"The destination container contains the source node
and the termination point reference list.";

leaf destination-id{
type node-id;
description
"The identifier of the destination node of the segment.";}
list termination-point-reference{
key "destination-support-id";
leaf destination-support-id{
type node-id;
description
"The identifier of the termination point of the source node.";}
}//end the termination-point-reference list
}//end the destination container

container direction{
leaf unidirection{
type boolean;
description
"Indicates whether the segment is unidirection or not.";}
leaf bidirection{
type boolean;
description
"Indicates whether the segment is bidirection or not.";}
} //end the direction container

container segment-extension{
```

```
description
"The segment-extension container can be used to augment
the segment list by segment specifics. Such as
netconf segment extension, i2rs segment extension.";

container netconf-segment-extension {
description
"To contain the netconf segment extension.";}
container i2rs-segment-extension {
description
"To contain the i2rs segment extension.";}
} //end the segment-extension container
}//end the segment list
}//end the service-topology list
}
}
<CODE ENDS>
```

5. SFC Topology YANG Module

```
<CODE BEGINS>file "sfc-topo@2013-12-23.yang"
module sfc-topo {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:sfc-topology";
    prefix "sfc-topo";
    import service-topo {
        prefix "stopo";
    }
    import ietf-inet-types {
        prefix "inet";
    }
organization "IETF I2RS Working Group";
     contact
      "wangzitao@huawei.com";
     description
       "This module defines sfc topology yang data model";
     typedef node-id {
       type inet:uri;
     }
augment "/stopo:service-topologies/stopo:service-topology/stopo:node/stopo:node-
extension/stopo:classifier-extension"{
 leaf classifier-id{
  type node-id;
  description
  "The identifier of the classifier.";}
 leaf sfc-policy{
  type uint32;
  description
  "Indicate the policy of sfc.";}
 container sfp{
  description
  "contains several sfps.";
  leaf sfp-id{
   type uint32;
   description
   "The identifier of the sfp.";}
  list sf-list{
   key "sf-id";
   leaf sf-id{
    type node-id;
    description
    "The identifier of the sf which include in the sfp.";}
  }
  list sff-list{
   key "sff-id";
   leaf sff-id{
    type node-id;
    description
    "The identifier of the sff which include in the sfp.";}
  }
 }//end the sfp container
}//end the classifier-extension

augment "/stopo:service-topologies/stopo:service-topology/stopo:node/stopo:node-
extension/stopo:sf-node-extension"{
 leaf sf-id{
  type node-id;
  description
  "The identifier of the service function(sf).";}
 leaf sf-node-locator{
  type sft:service-function-type;
  description
  "To indicate the service function (sf) locator";}
  container sf-type{
   leaf firewall{
    type sft:-service-function-type;
    description
    "To indicate the service function (sf) is firewall.";}
   leaf loadbalancer{
    type sft:-service-function-type;
    description
    "To indicate the service function (sf) is loadbalancer.";}
   leaf NAT44{
    type sft:-service-function-type;
    description
    "To indicate the service function (sf) is NAT44.";}
   leaf NAT64{
    type sft:-service-function-type;
    description
    "To indicate the service function (sf) is NAT64.";}
   leaf DPI{
    type sft:-service-function-type;
    description
    "To indicate the service function (sf) is DPI.";}
  }//end the sf-type container
  container sf-inventory-data{
   description
   "The container of the inventory data of service function (sf).";
  }
 }//end the sf-node-extension

augment "/stopo:service-topologies/stopo:service-topology/stopo:node/stopo:node-
extension/stopo:sff-node-extension"{
 leaf sff-id{
  type node-id;
  description
  "The identifier of the service function forward (sff).";}
 choice sffn-address{
  description
  "The address of the service function forward (sff) node";
   case ipv4-address{
    leaf ipv4-address{
     type inet:ipv4-address;}
   }
   case ipv6-address{
    leaf ipv6-address{
     type inet:ipv6-address;}
   }
  }//end the choice sffn-address
  container sffn-virtual-context{
   leaf context-id{
   type uint32;
   description
   "the identifier of the sffn virtual context.";}
  }
  container Attached-service-address{
   list service-node{
    key "service-node-id";
    leaf service-node-id{
     type node-id;
     description
     "The identifier of the service node.";}
   }//end the service-node list
   list host-system{
    key "host-system-id";
    leaf host-system-id{
     type uint32;
     description
     "The identifier of the host system.";}
   }//end the service-node list
  } //end the attached-service-address container
  list customer-support{
   key "customer-id";
   leaf customer-id{
    type uint32;
    description
    "The identifier of the customer.";}
  }//end the customer-support list
  list customer-service-resource{
   key "customer-resource-id";
   leaf customer-resource-id{
    type node-id;
    description
    "The identifier of the customer resource.";}
  }//end the customer-service-resource list
  container sffn-vntopo{
   description
   "This container can be use to contain the virtual network topology of
    Sffn. And it can be augment by specific virtual network topology.";
   }//end the sffn-vntopo container
  }//end the sff-node-extension
 }
<CODE ENDS>
```

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Normative References

[I-D.clemm-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Tkacik, T., Bahadur, N., and H. Ananthakrishnan, "A Data Model for Network Topologies", draft-clemm-i2rs-yang-network-topo-03 (work in progress), March 2015.

[I-D.ietf-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-10 (work in progress), January 2015.

[I-D.penno-sfc-yang]
Penno, R., Quinn, P., Zhou, D., and J. Li, "Yang Data Model for Service Function Chaining", draft-penno-sfc-yang-13 (work in progress), March 2015.

[I-D.xia-sfc-yang-oam]
Xia, L., Wu, Q., Kumar, D., Boucadair, M., and Z. Wang, "YANG Data Model for SFC Operations, Administration, and Maintenance (OAM)", draft-xia-sfc-yang-oam-02 (work in progress), March 2015.

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241’, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[draft-ietf-sfc-architecture-02]
Halpern, J., "Service Function Chaining (SFC) Architecture", ID draft-ietf-sfc-architecture-02, May 2014.

Authors’ Addresses

Zitao(Michael) Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Susan Hares
Huawei Technologies,Co.,Ltd
7453 Hickory Hill
Saline, MI 48176
USA

Email: Email: shares@ndzh.com

Jeff Tantsura
Ericsson

Email: Jeff Tantsura jeff.tantsura@ericsson.com

Russ White
Ericsson

Email: russw@riw.us

Qin Wu
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Phone: +86 25 56623633
Email: bill.wu@huawei.com

Linda Dunbar
Huawei Technologies,Co.,Ltd

Email: Email: Linda.Dunbar@huawei.com