I2RS Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

Y. Zhuang
D. Shi
Huawei
R. Gu
China Mobile

September 29, 2016

A YANG Data Model for Fabric Topology in Data Center Network
draft-zhuang-i2rs-yang-dc-fabric-network-topology-01

Abstract

This document defines a YANG data model for fabric topology in Data Center Network.

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 April 2, 2017.

Table of Contents

1. Introduction

Normally, a data center network is composed of single or multiple fabrics which are also known as PODs (a Point Of Delivery). These fabrics may be heterogeneous due to implementation of different technologies while DC network upgrading or enrolling new techniques and features. For example, for a DC network, Fabric A may use VXLAN while Fabric B may use VLAN. Likewise, a legacy Fabric may use VXLAN while a new Fabric B implemented technique such as GPE[] may be built due to DC expansion. The configuration and management of such DC networks with heterogeneous fabrics will be sophisticated and complex especially.

Luckily, for a DC network, fabric can be considered as an atomic structure to provide network services and deploy network management, as well as expand network capacity. From this point of view, the miscellaneous DC network management can be decomposed to task of managing each fabric respectively along with their connections, which can make the entire management much concentrated and flexible, also easy to expand.

With this purpose, this document defines a YANG data model for the Fabric-based Data center topology by using YANG [6020][7950]. To do so, it augments the generic network and network topology data models defined in [I-D.ietf-i2rs-yang-network-topo] with information specific to Data Center application.

This model defines the generic configuration and operational state for a fabric-based network topology, which can be extended by vendors

with specific information. This model can then be used by network elements to expose to a network controller and capture their understanding of the overall network topology and self role in the overall network. Also, it can be used by a network controller to represent its view of the fabric topology that it controls and expose it to applications for DC network management.

With the context of topology architecture defined in [I-D.ietf-i2rs-yang-network-topo] and [I.D. draft-ietf-i2rs-usecase-reqs-summary], this model can also be treated as an application of I2RS network topology model [I-D.ietf-i2rs-yang-network-topo] in the scenario of Data center network management. Furthermore, it can also act as a service topology when mapping network elements at fabric layer to elements in other topologies, such as L3 topology defined in [I.D. draft-ietf-i2rs-yang-l3-topology-01.

By using this fabric topology model, people can focus on characteristics of fabrics (such as encapsulation type, gateway type, etc.) as well as their connections while putting the underlay topology aside. As such, clients can consume the topology information at fabric level, while no need to parse entire set of links and nodes in underlay networks. The mapping between fabric layer and underlay topologies are made when creating a fabric, while further management of the network will dynamically be mapped from nodes of the fabric layer onto the related nodes in the underlay network.

2. Definitions an Acronyms

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].

2.1. Terminology

DC Fabric: also known as POD, is a module of network, compute, storage, and application components that work together to deliver networking services. It is a repeatable design pattern, and its components maximize the modularity, scalability, and manageability of data centers.

2.2. Tree diagram

The following notations are used within the data tree and carry the meaning as below.

```
  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 notifications
    <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
```

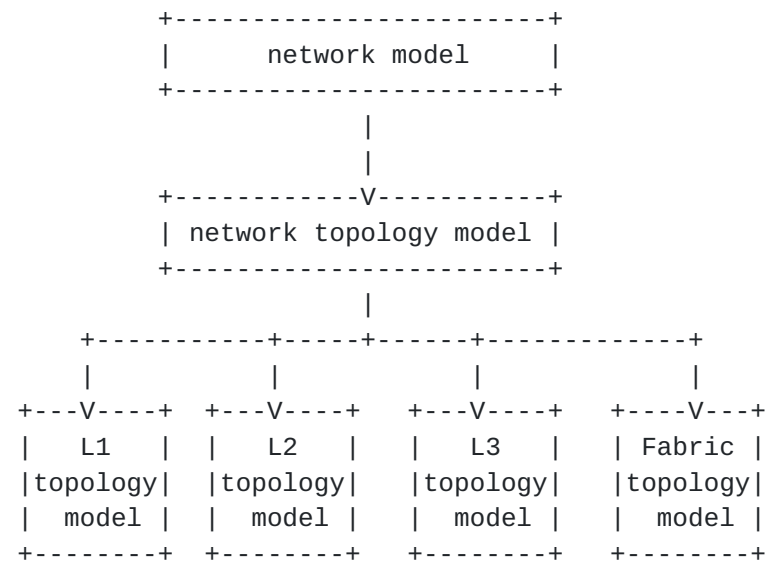
In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Model Overview

This section provides an overview of the DC Fabric topology model and its relationship with other topology models.

3.1. Topology Model structure

The relationship of the DC fabric topology model and other topology models is shown in the following figure (dotted lines in the figure denote augmentations).



From the perspective of resource management and service provisioning for a Data Center network, the fabric topology model augments the basic network topology model with definitions and features specific to a DC fabric, to provide common configuration and operations for heterogeneous fabrics.

3.2. Fabric Topology Model

The fabric topology model is composed of two key modules, fabric-topology module and fabric-capable-device module, which are designed to be generic and can be applied to data center fabrics built with different technologies, such as VLAN, VXLAN etc al. The main purpose is to configure and manage fabrics and their connections.

3.2.1. Fabric Toplogy

In the fabric topology module, fabric is modeled as a node in the network, while the fabric-based Data center network consists of a set of fabric nodes and their connections. The following is the snatch of the definition to show the main structure of the model:

```
module: ietf-fabric-topology
augment /nw:networks/nw:network/nw:network-types:
   +--rw fabric-network!
augment /nw:networks/nw:network/nw:node:
   +--rw fabric-attribute
          +--rw name?          string
          +--rw type?          fabrictype:underlayer-network-type
          +--rw description?   string
          +--rw options
          +--...
augment /nw:networks/nw:network/nw:node/nt:termination-point:
   +--ro fport-attribute
          +--ro name?         string
          +--ro role?         fabric-port-role
          +--ro type?         fabric-port-type
```

The fabric topology module augments the generic ietf-network and ietf-network-topology modules as follows:

- A new topology type "ietf-fabric-topology" is introduced and added under the "network-types" container of the ietf-network module.

- Fabric is defined as a node under the network/node container. A new container of "fabric-attribute" is defined to carry attributes for a fabric network such as gateway mode, fabric types, involved device nodes and links etc al.

- Termination points (in network topology module) are augmented with fabric port attributes defined in a container. The "termination-point" here can represent the "port" of a fabric that provides connections to other nodes, such as device internally, another fabric externally and also end hosts.

Details of fabric node and fabric termination point extension will be explained in the following sections.

3.2.1.1. Fabric node extension

As a network, a fabric itself is composed of set of network elements i.e. devices, and related links. As stated previously, the configuration of a fabric is contained under the "fabric-attribute" container depicted as follows:

```
  +--rw fabric-attribute
        +--rw name?           string
        +--rw type?           fabrictype:underlayer-network-type
        +--rw description?    string
        +--rw options
        |  +--rw gateway-mode?           enumeration
        |  +--rw traffic-behavior?       enumeration
        |  +--rw capability-supported*   fabrictype:service-capabilities
        +--rw device-nodes* [device-ref]
        |  +--rw device-ref    fabrictype:node-ref
        |  +--rw role?         fabrictype:device-role
        +--rw device-links* [link-ref]
           +--rw link-ref    fabrictype:link-ref
```

As in the module, additional data objects for nodes are introduced by augmenting the "node" list of the network module. New objects include fabric name, type of the fabric, descriptions of the fabric as well as a set of options defined in an "options" container. The options container includes type of the gateway-mode (centralized or distributed) and traffic-behavior (whether acl needed for the traffic).

Also, it defines a list of device-nodes and related links as supporting-nodes to form a fabric network. These device nodes and links are leaf-ref of existing nodes and links in the physical topology. For the device-node, the "role" object is defined to represents the role of the device within the fabric, such as "SPINE" or "LEAF", which should work together with gateway-mode.

3.2.1.2. Fabric termination-point extension

Since the fabric is considered as a node, in this concept, "termination-points" can represent "ports" of a fabric that connects to other fabrics or end hosts, besides representing ports that connect devices inside the fabric itself.

As such, the "termination-point" in the fabric topology has three roles, including internal TP that connects to devices within a fabric, external TP that connects to outside network, as well as access TP to end hosts.

A set of "termination-point" indicates all connections of a fabric including its internal connections, interconnections with other fabrics and also connections to end hosts for a DC network.

The structure of fabric ports is as follows:

```
augment /nw:networks/nw:network/nw:node/nt:termination-point:
   +--ro fport-attribute
      +--ro name?          string
      +--ro role?          fabric-port-role
      +--ro type?          fabric-port-type
      +--ro device-port?   tp-ref
      +--ro (tunnel-option)?
         +--:(gre)
            +--ro src-ip?        inet:ip-prefix
            +--ro dest-ip?       inet:ip-address
```

It augments the termination points (in network topology module) with fabric port attributes defined in a container.

New nodes are defined for fabric ports which include name, role of the port within the fabric (internal port, external port to outside network, access port to end hosts), port type (l2 interface, l3 interface etc al). By using the device-port defined as a tp-ref, this fabric port can be mapped to a device node in the underlay network.

Also, a new container for tunnel-options is introduced as well to present the tunnel configuration on the port.

3.2.1.3. Operations for Fabric topology

In addition, we also define several rpc operations for fabric managements within data center networks to get and configure fabrics and obtain a response as depicted below.

```
rpcs:
   +---x get-all-fabric
   |  +--ro output
   |     +--ro fabric-id*   fabric-id
   +---x compose-fabric
   |  +---w input
   |  |  +---w fabric-network-attributes
   |  |     +---w name?          string
   |  |     +---w fabric-id?     fabric-id
   |  |     +---w type?          fabrictype:underlayer-network-type
   |  |     +---w description?   string
   |  |     +---w options
   |  |        +---w gateway-mode?       enumeration
   |  |        +---w traffic-behavior?   enumeration
   |  +--ro output
   |     +--ro fabric-id?   fabric-id
   +---x decompose-fabric
   |  +---w input
   |     +---w fabric-id?   fabric-id
   +---x add-node-to-fabric
   |  +---w input
   |     +---w fabric-id?   fabric-id
   |     +---w node-ref?    fabrictype:node-ref
   |     +---w role?        fabrictype:device-role
   +---x rm-node-from-fabric
   |  +---w input
   |     +---w fabric-id?     fabric-id
   |     +---w node-ref?      -> /nd:networks/network[nd:network-
id=current()/../network-ref]/node/node-id
   |     +---w network-ref?   -> /nd:networks/network/network-id
   +---x add-link-to-fabric
   |  +---w input
   |     +---w fabric-id?   fabric-id
   |     +---w link-ref?    fabrictype:link-ref
   +---x rm-link-from-fabric
      +---w input
         +---w fabric-id?     fabric-id
         +---w link-ref?      -> /nd:networks/network[nd:network-
id=current()/../network-ref]/lnk:link/link-id
         +---w network-ref?   -> /nd:networks/network/network-id
```

In general, the fabric topology module defines 7 rpc commands for this management, including getting all fabric information, composing a fabric, decomposing a fabric, adding a device node to a fabric, removing a device node from a fabric, adding a link to a fabric and removing a link from a fabric.

3.2.2. Fabric Device Module

Besides the fabric topology module, the fabric-capable-device module is defined to configure individual network elements i.e. devices of physical network with specific attributes for fabric network which acts as supporting nodes of a fabric that provide fabric services.

The structure of "ietf-fabric-capable-device" data model is depicted in the following diagram.

```
                module: ietf-fabric-capable-device
  augment /nw:networks/nw:network/nw:node:
     +--rw supported-fabric*       identityref
     +--rw capability-supported*   fabrictype:service-capabilities
     +--ro attributes
     |  +--ro fabric-id?    nw:node-id
     |  +--ro fabric-ref?   fabrictype:node-ref
     +--rw config
        +--rw bridge-domain* [id]
        |  +--rw id         string
        |  +--rw flood?     boolean
        |  +--rw arp?       boolean
        |  +--rw segment?   uint32
        +--rw bd-port* [bd-port-id]
        |  +--rw bd-port-id     string
        |  +--rw bdid?          string
        |  +--rw ref-tp-id?     nw:node-id
        |  +--rw access-type?   fabrictype:access-type
        |  +--rw access-tag?    uint32
        |  +--rw fabric-acl* [fabric-acl-name]
        |     +--rw fabric-acl-name    string
        +--rw vrf* [id]
        |  +--rw id         string
        |  +--rw name?      string
        |  +--rw vrf-ctx?   int32
        +--rw bdif* [id]
           +--rw id               string
           +--rw bdid?            string
           +--rw vrf?             int32
           +--rw ip-address?      inet:ip-address
           +--rw mask?            int32
           +--rw mac-address?     yang:mac-address
           +--rw fabric-acl* [fabric-acl-name]
           |  +--rw fabric-acl-name    string
           +--rw port-function
              +--rw (function-type)?
                 +--:(ip-mapping)
                    +--rw ip-mapping-entry* [external-ip]
                       +--rw external-ip    inet:ipv4-address
                       +--rw internal-ip?   inet:ipv4-address
  augment /nw:networks/nw:network/nw:node/nt:termination-point:
     +--rw port-role?   fabrictype:fabric-port-role
     +--rw port-ref?    fabrictype:tp-ref
  augment /nw:networks/nw:network/nw:node/nt:termination-point:
     +--rw bd-ids*   instance-identifier
```

The ietf-fabric-capable-device augments the generic ietf-network and ietf-network-topology modules with specific attributes for fabric services as follows:

- Additional data objects for nodes are introduced by augmenting the "node" list of the network module to carry specific fabric features for a fabric capable device. New objects include type of supported fabric such as VXLAN fabric or VLAN fabric, supported capability such as whether the device supports Network Address Transformation or Service Function Chain functions. Also, the "host" fabric is indicated to show which fabric the device belonged to.

- Besides, new objects for configuring the device are contained in the "config" container, which includes the L2 attributes and L3 attributes of the bridge.

- Termination points are also augmented with attributes to indicate the role of the device port in a fabric and its role in the bridge network.

4. Fabric YANG Modules

```
<CODE BEGINS> file "ietf-fabric-types@2016-09-29.yang"
module ietf-fabric-types {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-types";
    prefix fabrictypes;

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

        import ietf-network-topology { prefix nt;  }

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
    "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
     WG List:   <mailto:i2rs@ietf.org>

     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair:  Russ White
                <mailto:russ@riw.us>
```

```
     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
                <mailto:shidanian@huawei.com>";

    description
        "This module contains a collection of YANG definitions for Fabric.";

    revision "2016-09-29" {
        description
            "Initial revision of faas.";
                reference
                        "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
    }

    identity fabric-type {
                description
                        "base type for fabric networks";
    }

    identity vxlan-fabric {
        base fabric-type;
                description
                        "vxlan fabric";
    }

    identity vlan-fabric {
        base fabric-type;
                description
                        "vlan fabric";
    }

    typedef service-capabilities {
        type enumeration {
            enum ip-mapping {
                                description "NAT";
                        }
            enum acl-redirect{
                                description "acl redirect, which can provide
SFC function";
                        }
            enum dynamic-route-exchange{
                                description "dynamic route exchange";
                        }
        }
                description
                        "capability of the device";
```

```
    }
```

```
    /*
     * Typedefs
     */
    typedef node-ref {
        type instance-identifier;
        description "A reference to a node in topology";
    }

    typedef tp-ref {
        type instance-identifier;
        description "A reference to a termination point in topology";
    }

    typedef link-ref {
                type instance-identifier;
        description "A reference to a link in topology";
    }

    typedef device-role {
        type enumeration {
            enum SPINE {
                description "a spine node";
            }
            enum LEAF {
                description "a leaf node";
            }
            enum BORDER {
                description "a border node";
            }
        }
        default "LEAF";
        description "device role type";
    }

    typedef fabric-port-role {
        type enumeration {
            enum internal {
                description "the port used for devices to access each other.";
            }
            enum external {
                description "the port used for fabric to access outside
network.";
            }
            enum access {
                description "the port used for Endpoint to access fabric.";
            }
            enum reserved {
                description " not decided yet. ";
```

```
            }
```

```
        }
                description "the role of the physical port ";
    }

    typedef fabric-port-type {
        type enumeration {
            enum layer2interface {
                                description "l2 if";
            }
            enum layer3interface {
                                description "l3 if";
            }
            enum layer2Tunnel {
                                description "l2 tunnel";
            }
            enum layer3Tunnel {
                                description "l3 tunnel";
            }
        }
                description
                        "fabric port type";
    }

    typedef underlayer-network-type {
        type enumeration {
            enum VXLAN {
                description "vxlan";
            }
            enum TRILL {
                description "trill";
            }
            enum VLAN {
                description "vlan";
            }
        }
                description "";
    }

    typedef layer2-protocol-type-enum {
        type enumeration {
            enum VLAN{
                                description "vlan";
                        }
            enum VXLAN{
                                description "vxlan";
                        }
            enum TRILL{
                                description "trill";
```

```
                        }
            enum NvGRE{
                                description "nvgre";
                        }
        }
                description "";
    }

    typedef access-type {
        type enumeration {
            enum exclusive{
                                description "exclusive";
                        }
            enum vlan{
                                description "vlan";
                        }
        }
        description "";
    }

    grouping fabric-port {
                description
                        "attributes of a fabric port";
        leaf name {
            type string;
                        description "name of the port";
        }
        leaf role {
            type fabric-port-role;
                        description "role of the port in a fabric";
        }
        leaf type {
            type fabric-port-type;
                        description "type of the port";
        }
        leaf device-port {
            type tp-ref;
                        description "the device port it mapped to";
        }
        choice tunnel-option {
                        description "tunnel options";

            case gre {
                 leaf src-ip {
                    type inet:ip-prefix;
                                        description "source address";
                }
                leaf dest-ip {
```

```
                    type inet:ip-address;
                                        description "destination address";
                }
            }
        }
    }

    grouping route-group {
                description
                        "route attributes";
        list route {
            key "destination-prefix";
                        description "route list";

            leaf description {
                type string;
                description "Textual description of the route.";
            }
            leaf destination-prefix {
                type inet:ipv4-prefix;
                mandatory true;
                description "IPv4 destination prefix.";
            }
            choice next-hop-options {
                                description "choice of next hop options";
                case simple-next-hop {
                    leaf next-hop {
                        type inet:ipv4-address;
                        description "IPv4 address of the next hop.";
                    }
                    leaf outgoing-interface {
                        type nt:tp-id;
                        description "Name of the outgoing interface.";
                    }
                }
            }
        }
    }

    grouping port-functions {
                description
                        "port functions";

        container port-function {
                        description "port functions";
            choice function-type {
                                description "type of functions";
                case ip-mapping {
```

```
                    list ip-mapping-entry {
                                                key "external-ip";
                                                description "list of NAT
entry";
                        leaf external-ip {
                            type inet:ipv4-address;
                                                        description "external
address";
                        }
                        leaf internal-ip {
                            type inet:ipv4-address;
                                                        description "internal
address";
                        }
                    }
                }
            }
        }
    }
    grouping acl-list {
                description "acl list";
        list fabric-acl {
                        key fabric-acl-name;
                        description "fabric acl list";
            leaf fabric-acl-name {
                type string;
                                description "acl name";
            }
        }
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-topology@2016-09-29.yang"
module ietf-fabric-topology {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-topology";
    prefix fabric;

        import ietf-network { prefix nw;  }
        import ietf-network-topology { prefix nt;  }
    import ietf-fabric-types { prefix fabrictype; revision-date "2016-09-29"; }

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
```

"WG Web: <http://tools.ietf.org/wg/i2rs/ >
WG List: <mailto:i2rs@ietf.org>

```
     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair:  Russ White
                <mailto:russ@riw.us>

     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
                <mailto:shidanian@huawei.com>";


    description
        "This module contains a collection of YANG definitions for Fabric.

                Copyright (c) 2016 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
                draft-zhuang-i2rs-yang-dc-fabric-network-topology;
                see the RFC itself for full legal notices.";

    revision "2016-09-29" {
        description
            "Initial revision of fabric topology.";
                reference
                        "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
    }

    identity fabric-context {
                description
                        "identity of fabric context";
    }

    typedef fabric-id {
        type nw:node-id;
        description
            "An identifier for a fabric in a topology.
            The identifier is generated by compose-fabric RPC.";
    }
```

```
        //grouping statements
        grouping fabric-network-type {
      description "Identify the topology type to be fabric.";
      container fabric-network {
        presence "indicates fabric Network";
        description
        "The presence of the container node indicates
         fabric Topology";
      }
    }

    grouping fabric-options {
                description "options for a fabric";

        leaf gateway-mode {
            type enumeration {
                enum centralized {
                    description "centerilized gateway";
                }
                enum distributed {
                    description "distributed gateway";
                }
            }
            default "distributed";
                        description "gateway mode";
        }

        leaf traffic-behavior {
            type enumeration {
                enum normal {
                    description "normal";
                }
                enum policy-driven {
                    description "policy driven";
                }
            }
            default "normal";
                        description "traffic behavior of the fabric";
        }

        leaf-list capability-supported {
            type fabrictype:service-capabilities;
                        description
                                "supported services of the fabric";
        }
    }

    grouping device-attributes {
```

```
                description "device attributes";
        leaf device-ref {
            type fabrictype:node-ref;
                        description
                                "the device it includes to";
        }
        leaf role {
            type fabrictype:device-role;
            default "LEAF";
                        description
                                "role of the node";
        }
    }

    grouping link-attributes {
                description "link attributes";
        leaf link-ref {
            type fabrictype:link-ref;
                        description
                                "the link it includes";
        }
    }

    grouping fabric-attributes {
                description "attributes of a fabric";
        leaf name {
            type string;
                        description
                                "name of the fabric";
        }

        leaf type {
            type fabrictype:underlayer-network-type;
            description
                                "The type of physical network that implements
this fabric.Examples are vlan, and trill.";
        }

        leaf description {
            type string;
                        description
                                "description of the fabric";
        }

        container options {
                        description "options of the fabric";
            uses fabric-options;
        }
```

```
        list device-nodes {
                        key device-ref;
                        description "include device nodes in the fabric";
            uses device-attributes;
        }

        list device-links {
                        key link-ref;
                        description "include device links within the fabric";
            uses link-attributes;
        }

    }

        // augment statements

        augment "/nw:networks/nw:network/nw:network-types" {
    description
      "Introduce new network type for Fabric-based logical topology";

                uses fabric-network-type;
        }

    augment "/nw:networks/nw:network/nw:node" {
        when "/nw:networks/nw:network/nw:network-types/fabric-network" {
        description
          "Augmentation parameters apply only for networks
           with fabric topology";
      }
        description "Augmentation for fabric nodes created by faas.";

        container fabric-attribute {
                        description
                        "attributes for a fabric network";

            uses fabric-attributes;
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
        when "/nw:networks/nw:network/nw:network-types/fabric-network" {
        description
          "Augmentation parameters apply only for networks
           with fabric topology";
      }
        description "Augmentation for port on fabric.";

        container fport-attribute {
```

```
            config false;
                        description
                        "attributes for fabric ports";
            uses fabrictype:fabric-port;
        }
    }

    /
*****************************RPC*********************************************/
    rpc get-all-fabrics {
                description "get all fabrics";
        output {
            leaf-list fabric-id {
               type fabric-id;
                           description
                                "fabric id list";
            }
        }
    }

    rpc compose-fabric {
                description "compose a fabric";
        input {
            uses fabric-attributes;
        }
        output {
            leaf fabric-id {
                type fabric-id;
                                description
                                        "fabric id";
            }
        }
    }

    rpc decompose-fabric {
                description "decompose a fabric";
        input {
            leaf fabric-id {
                type fabric-id;
                                description
                                        "fabric id";
            }
        }
    }

    rpc add-node-to-fabric {
                description "add nodes to a fabric";
        input {
```

```
            leaf fabric-id {
```

```
                type fabric-id;
                                description
                                        "fabric id";
            }
            leaf node-ref {
                type fabrictype:node-ref;
                                description
                                        "node reference";
            }
            leaf role {
                type fabrictype:device-role;
                                description
                                        "role of the node";
            }
        }
    }

    rpc rm-node-from-fabric {
                description "remove nodes from a fabric";
        input {
            leaf fabric-id {
                type fabric-id;
                                description
                                        "fabric id";
            }
            leaf node-ref {
                type fabrictype:node-ref;
                                description
                                        "node reference";
            }
        }
    }

    rpc add-link-to-fabric {
                description "add links to a fabric";
        input {
            leaf fabric-id {
                type fabric-id;
                                description
                                        "fabric id";
            }
            leaf link-ref {
                type fabrictype:link-ref;
                                description
                                        "link reference";
            }
        }
    }
```

```
    rpc rm-link-from-fabric {
                description "remove link from a fabric";
        input {
            leaf fabric-id {
                type fabric-id;
                                description
                                        "fabric id";
            }
            leaf link-ref {
                type fabrictype:link-ref;
                                description
                                        "link reference";
            }
        }
    }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-capable-device@2016-09-29.yang"
module ietf-fabric-capable-device {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-capable-device";
    prefix fabric-device;

    import ietf-inet-types { prefix "inet"; revision-date "2013-07-15"; }
    import ietf-network { prefix "nw"; }
        import ietf-network-topology { prefix nt; }
    import ietf-yang-types { prefix "yang"; revision-date "2013-07-15"; }
    import ietf-fabric-types { prefix fabrictype; revision-date "2016-09-29"; }

        organization
    "IETF I2RS (Interface to the Routing System) Working Group";

        contact
    "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
     WG List:   <mailto:i2rs@ietf.org>

     WG Chair:  Susan Hares
                <mailto:shares@ndzh.com>

     WG Chair:  Russ White
                <mailto:russ@riw.us>

     Editor:    Yan Zhuang
                <mailto:zhuangyan.zhuang@huawei.com>

     Editor:    Danian Shi
```

```
                <mailto:shidanian@huawei.com>";

    description
        "This module contains a collection of YANG definitions for Fabric.";

    revision "2016-09-29" {
        description
            "Initial revision of faas.";
                reference
                        "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
    }

    identity fabric-capable-device-context {
                description
                        "identity of fabric capable device";
    }

    grouping fabric-capable-device-attribute {
                description
                        "attributes of fabric capable device";

        leaf fabric-id {
            type nw:node-id;
                        description "fabric id";
        }
        leaf fabric-ref {
            type fabrictype:node-ref;
                        description "fabric reference";
        }
    }

    grouping fabric-capable-device-config {
                description
                        "fabric service configuration of fabric capable
device";

        list bridge-domain {
            key id;
                        description "list of bridge domains";
            leaf id {
                type string;
                                description "id of the device";
            }
            leaf flood {
                type boolean;
                default false;
                                description "flood";
            }
```

```
            leaf arp {
```

```
                type boolean;
                default false;
                                description "arp";
            }
            leaf segment {
                        type uint32;
                                description "segment";
            }
        }

        list bd-port {
                        key bd-port-id;
            description "bridge port list";
                        leaf bd-port-id {
                type string;
                                description "bridge port id";
            }
            leaf bdid {
                type string;
                                description "bridge id";
            }
            leaf ref-tp-id {
                type nt:tp-id;
                                description "reference of termination port id";
            }
            leaf access-type {
                type fabrictype:access-type;
                                description "access type";
            }
            leaf access-tag {
                type uint32;
                                description "access tag";
            }
            uses fabrictype:acl-list;
        }

        list vrf {
                        key id;
            description "vrf list";
                        leaf id {
                type string;
                                description "vrf id";
            }
            leaf name {
                type string;
                                description "name";
            }
            leaf vrf-ctx {
```

```
                type int32;
                                description "context";
            }
        }

        list bdif {
                        key id;
            description "bridge interface list";
                        leaf id {
                type string;
                                description "bridge interface id";
            }
            leaf bdid {
                type string;
                                description "bridge id";
            }
            leaf vrf {
                type int32;
                                description "vrf";
            }
            leaf ip-address {
                type inet:ip-address;
                                description "ip address";
            }
            leaf mask {
                type int32;
                                description "mask";
            }
            leaf mac-address {
                type yang:mac-address;
                                description "mac address";
            }
            uses fabrictype:acl-list;
            uses fabrictype:port-functions;
        }
    }

    augment "/nw:networks/nw:network/nw:node" {
                description "augmentation for fabric capable device nodes";
        leaf-list supported-fabric {
            type identityref {
                base fabrictype:fabric-type;
            }
                        description "supported fabric types";
        }

        leaf-list capability-supported {
                        type fabrictype:service-capabilities;
```

```
                        description "service capability list";
        }

        container attributes {
            config false;
                        description "attributes of the device";
            uses fabric-capable-device-attribute;
        }

        container config {
                        description "configuration of the device";
            uses fabric-capable-device-config;
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
                description
                        "References a termination point to a tp in a fabric";

        leaf port-role {
            type fabrictype:fabric-port-role;
                        description "role in a fabric";
        }
        leaf port-ref {
            type fabrictype:tp-ref;
                        description "port reference in fabric";
        }
    }

    augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
                description
                        "References a termination point to a tp in bridges";
        leaf-list bd-ids {
            type instance-identifier;
                        description "bridge id list";
        }
    }
}
<CODE ENDS>
```

5. Security Consideration

TBD

6. Acknowledgements

7. References

7.1. Informative References

[I-D.draft-ietf-i2rs-yang-l3-topology]
Clemm, A., Medved, J., Tkacik, T., Liu, X., Bryskin, I., Guo, A., Ananthakrishnan, H., Bahadur, N., and V. Beeram, "A YANG Data Model for Layer 3 Topologies", I-D draft-ietf-i2rs-yang-network-topo-02, December 2015.

[I-D.draft-ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N., and H. Ananthakrishnan, "A YANG Data Model for Network Topologies", I-D draft-ietf-i2rs-yang-network-topo-02, December 2015.

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

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

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, Auguest 2016.

7.2. Informative References

[I-D.draft-ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", I-D draft-ietf-i2rs-usecase-reqs-summary-01, May 2015.

Authors' Addresses

Yan Zhuang
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhuangyan.zhuang@huawei.com

Danian Shi
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: shidanian@huawei.com

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: gurong_cmcc@outlook.com