Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 4, 2015

J. Dong
M. Chen
Huawei Technologies

March 3, 2015

A Data Model for Network Inventories
draft-dong-i2rs-network-inventory-00

Abstract

This document defines a YANG data model for network inventories.

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

Copyright Notice

Table of Contents

1. Introduction

This document defines the Yang [RFC6020] data model for network inventory information. It augments the abstract (base) network Yang model defined in [I-D.clemm-i2rs-yang-network-topo] with inventory information of the network entities. This inventory model also describes the protocol independent features, functions and capabilities of the network entities and their components. This network inventory data model may be augmented to describe detailed inventory information and characteristics.

This network inventory data model can be used by applications in several ways, such as:

- to obtain a complete view of the network inventory information;

- to enable/disable some protocol independent functions with the characteristics specified in the data model;

- to receive notifications of the changes in the network inventories.

The relationship between the inventory model and the abstract network model is shown in the diagram below:

```
                     +------------------------+
                     |                        |
                     | Abstract Network Model |
                     |                        |
                     +------------------------+
                                 |
                          +-------+-------+
                          |               |
                          V               V
                   +------------+  +------------+
                   |  Abstract  |  |  Network   |
                   |  Topology  |  | Inventory  |
                   |   Model    |  |   Model    |
                   +------------+  +------------+
```

Figure 1. Relationship of network inventory model and network model

2. Network Inventory Model Structure

Network inventory refers to the network entities used in building the network. Normally each network entity consists of a series of cards and interfaces, both the network entity and its components have a set of features and support a set of functions. Those protocol independent features and functions are described in this inventory model. For the interface part, some contents of the ietf-interfaces module as defined in [RFC7223] are reused.

The structure of "network-inventory" data model is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parentheses enclose choice and case nodes.

```
   module: network-inventory
   augment /ntw:network/ntw:node:
      +--rw node-name?              string
      +--rw node-description?       string
      +--rw hardware-version?       string
      +--rw software-version?       string
      +--ro capability-supported*   node-capability
      +--rw capability-enabled*     node-capability
      +--rw node-status?            node-status
      +--rw power-consumption?      uint32
      +--rw card* [card-id]
      |  +--rw card-id             inet:uri
      |  +--rw card-description?   string
      |  +--rw card-type?          identityref
      |  +--rw card-capability*    card-capability
      |  +--rw admin-status?       card-status
      +--rw interface* [if-name]
         +--rw if-name         if:interface-ref
         +--rw description?    string
         +--rw card-ref?       leafref
         +--rw if-type?        identityref
         +--rw phys-address?   yang:phys-address
         +--rw transceiver?    identityref
         +--rw throughput?     uint64
         +--rw admin-status?   enumeration
         +--ro oper-status?    enumeration
```

3. Network Inventory Yang Module

```
module network-inventory {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:network-inventory";
    // replace with IANA namespace when assigned

    prefix "netinv";

    import ietf-yang-types {
      prefix yang;
    }

    import ietf-inet-types {
      prefix inet;
    }

    import ietf-interfaces {
      prefix if;
    }
    import iana-if-type {
      prefix ift;
    }

    import network {
      prefix ntw;
    }

    organization "TBD";
    contact "I-D editor: jie.dong@huawei.com";
    description
    "This module defines a model for the network inventory.
     Key design considerations are as follows:
     A network consists of a set of network nodes.
     A network node can support a set of functions & capabilities.
     A network node consists of a set of cards and interfaces.
     A network node has a set of parameters to reflect its status.";

    revision "2015-02-10" {
        description "Initial revision";
        reference "TBD";
    }

   /*
    * Typedefs
    */

    typedef node-capability {
        type enumeration {
            enum ip-unicast {
                value 1;
            }
            enum ip-multicast {
                value 2;
            }
            enum mpls {
                value 3;
            }
            enum mpls-traffic-engineering {
                value 4;
            }
            enum ethernet-bridging {
                value 5;
            }
            enum cgn {
                value 6;
            }
        }
    }

    typedef node-status {
        type enumeration {
            enum other {
                value 0;
            }
            enum operating {
                value 1;
            }
            enum idle {
                value 2;
            }
            enum dormant {
                value 3;
            }
        }
    }

    typedef card-capability {
        type enumeration {
            enum ip-forwarding {
                value 1;
            }
            enum ethernet-bridging {
                value 2;
            }
            enum mpls-forwarding {
                value 3;
            }
            enum traffic-accounting {
                value 4;
            }
            enum caching {
                value 5;
            }
            enum cgn {
                value 6;
            }
        }
    }

    typedef card-status {
        type enumeration {
            enum operating {
                value 1;
            }
            enum idle {
                value 2;
            }
            enum dormant {
                value 3;
            }
        }
    }


   /*
    * Features
    */


   /*
    * Identities
    */

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

    identity control-card {
        base card-type;
        description
          "control-card";
    }

    identity line-card {
        base card-type;
        description
          "line-card";
    }

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

    identity transceiver-type {
        description
          "Base identity from which specific
           transceiver types are derived.";
    }
   /*
    * Data nodes
    */

    augment "/ntw:network/ntw:node" {

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

       leaf node-description {
           type string;
       }

       leaf hardware-version {
           type string;
       }

       leaf software-version {
           type string;
       }

       leaf-list capability-supported {
           config false;
           type node-capability;
           description
            "A list of capabilities the node can support.";
       }

       leaf-list capability-enabled {
           type node-capability;
           description
             "A list of capabilities that has enabled.";
       }

       leaf node-status {
           type node-status;
       }

       leaf power-consumption {
           type uint32;
           units "watt";
       }

       list card {
           key "card-id";
```

```
            leaf card-id {
                type inet:uri;
            }

            leaf card-description {
                type string;
                description
                  "The description of the card";
            }

            leaf card-type {
                type identityref {
                    base card-type;
                }
            }

            leaf-list card-capability {
                type card-capability;
                description
                  "A list of capabilities the card supports.";
            }

            leaf admin-status {
                type card-status;
            }
        } //list-card

        list interface {
            key "if-name";

            leaf if-name {
                type if:interface-ref;
                description
                  "A reference to an interface";
            }

            leaf description {
                type string;
            }

            leaf card-ref {
                type leafref {
                    path "/ntw:network/ntw:node/card/netinv:card-id";
                }
                description
                  "A reference to the card which the interface belongs to.";
            }
```

```
           leaf if-type {
               type identityref {
                   base ift:iana-interface-type;
               }
           }

           leaf phys-address {
               type yang:phys-address;
               description
                 "The inteface’s physical address.";
           }

           leaf transceiver {
               type identityref {
                   base transceiver-type;
               }
               description
                 "The interface’s transceiver type";
           }

           leaf throughput {
               description
                 "The maximum throughput of the interface.";
               type uint64;
               units "bits/second";
           }

           leaf admin-status {
               type enumeration {
                   enum up {
                     value 1;
                     description
                       "Ready to pass packets.";
                   }
                   enum down {
                     value 2;
                     description
                       "Not ready to pass packets and not in test mode.";
                   }
                   enum testing {
                       value 3;
                       description
                         "In some test mode.";
                   }
               }
           }

           leaf oper-status {
               config false;
               type enumeration {
                   enum up {
                     value 1;
                     description
                       "Ready to pass packets.";
                   }
                   enum down {
                     value 2;
                     description
                       "The interface does not pass any packets.";
                   }
                   enum testing {
                     value 3;
                     description
                       "In some test mode.  No operational packets can
                        be passed.";
                   }
                   enum unknown {
                     value 4;
                     description
                       "Status cannot be determined for some reason.";
                   }
                   enum dormant {
                     value 5;
                     description
                       "Waiting for some external event.";
                   }
                   enum not-present {
                     value 6;
                     description
                       "Some component (typically hardware) is missing.";
                   }
                   enum lower-layer-down {
                     value 7;
                     description
                       "Down due to state of lower-layer interface(s).";
                   }
               }
           }

       }//list interface
    }


   /*
    * Notifications: to be added
    */
```

```
} //module
```

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

The transport protocol used for sending information of this data model MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

6. Acknowledgements

TBD

7. 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-02 (work in progress), December 2014.

[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.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

Authors’ Addresses

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: mach.chen@huawei.com