Workgroup: CCAMP Working Group
Internet-Draft:
draft-yg3bp-ccamp-network-inventory-yang-02
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: C. Yu
Huawei Technologies
I. Busi
Huawei Technologies
A. Guo
Futurewei Technologies
S. Belotti
Nokia
J.-F. Bouquier
Vodafone
F. Peruzzini
TIM
O. Gonzalez de Dios
Telefonica
V. Lopez
Nokia

A YANG Data Model for Network Hardware Inventory

Abstract

This document defines a YANG data model for network hardware inventory data information.

The YANG data model presented in this document is intended to be used as the basis toward a generic YANG data model for network hardware inventory data information which can be augmented, when required, with technology-specific (e.g., optical) inventory data, to be defined either in a future version of this document or in another document.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA).

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 https://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 27 April 2023.

Table of Contents

1. Introduction

Network hardware inventory management is a key component in operators' OSS architectures.

Network inventory is a fundamental functionality in network management and was specified many years ago. Given the emerging of data models and their deployment in operator's management and control systems, the traditional function of inventory management is also requested to be defined as a data model.

Network inventory management and monitoring is a critical part for ensuring the network stays healthy, well-planned, and functioning in the operator's network. Network inventory management allows the operator to keep track of what physical network devices are staying in the network including relevant software and hardware versions.

The network inventory management also helps the operator to know when to acquire new assets and what is needed, or to decommission old or faulty ones, which can help to improve network performance and capacity planning.

In [I-D.ietf-teas-actn-poi-applicability] a gap was identified regarding the lack of a YANG data model that could be used at ACTN MPI interface level to report whole/partial network hardware inventory information available at domain controller level towards north-bound systems (e.g., MDSC or OSS layer).

[RFC8345] initial goal was to make possible the augmentation of the YANG data model with network inventory data model but this was never developed and the scope was kept limited to network topology data only.

It is key for operators to drive the industry towards the use of a standard YANG data model for network inventory data instead of using vendors proprietary APIs (e.g., REST API).

In the ACTN architecture, this would bring also clear benefits at MDSC level for packet over optical integration scenarios since this would enable the correlation of the inventory information with the links information reported in the network topology model.

The intention is to define a generic YANG data model that would be as much as possible technology agnostic (valid for IP, optical and microwave networks) and that could be augmented, when required, to include some technology-specific inventory details.

[RFC8348] defines a YANG data model for the management of the hardware on a single server and therefore it is more applicable to the domain controller South Bound Interface (SBI) towards the network elements rather than at the domain controller's northbound. However, the YANG data model defined in [RFC8348] has been used as a reference for defining the YANG network hardware inventory data model presented in this draft.

For optical network hardware inventory, the network inventory YANG data model should support the use cases (4a and 4b) and requirements as defined in [ONF_TR-547], in order to guarantee a seamless integration at MDSC/OSS/orchestration layers.

The proposed YANG data model has been analysed at the present stage to cover the requirements and use cases for Optical Network Hardware Inventory.

Being based on [RFC8348], this data model should be a good starting point toward a generic data model and applicable to any technology. However, further analysis of requirements and use cases is needed to extend the applicability of this YANG data model to other types of networks (IP and microwave) and to identify which aspects are generic and which aspects are technology-specific for optical networks.

This document defines one YANG module: ietf-network-inventory.yang (Section 4).

Note: review in future versions of this document the related modules, depending on the augmentation relationship.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture [RFC8342].

1.1. Terminology and Notations

The following terms are defined in [RFC7950] and are not redefined here:

- *client
- *server
- *augment
- *data model
- *data node

The following terms are defined in [RFC6241] and are not redefined here:

- *configuration data
- *state data

The terminology for describing YANG data models is found in [RFC7950].

TBD: Recap the concept of chassis/slot/component/board/... in [TMF-MTOSI].

Following terms are used for the representation of the hierarchies in the network hardware inventory.

Network Element:

a device installed on one or several chassis and can afford some specific transmission function independently.

Rack:

a holder of the device and provides power supply for the device in it.

Chassis:

a holder of the device installation.

Slot:

a holder of the board.

Component:

holders and equipment of the network element, including chassis, slot, sub-slot, board and port.

Board/Card:

a pluggable equipment can be inserted into one or several slots/ sub-slots and can afford a specific transmission function independently.

Port:

an interface on board

1.2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Tree Diagram

A simplified graphical representation of the data model is used in Section 3 of this document. The meaning of the symbols in this diagram is defined in [RFC8340].

1.4. Prefix in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in the following table.

Prefix	Yang Module	Reference
ianahw	iana-hardware	[RFC8348]
ni	ietf-network-inventory	RFC XXXX
yang	ietf-yang-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

RFC Editor Note: Please replace XXXX with the RFC number assigned to this document. Please remove this note.

2. YANG Data Model for Network Hardware Inventory

2.1. YANG Model Overview

Based on TMF classification in [TMF-MTOSI], inventory objects can be divided into two groups, holder group and equipment group. The holder group contains rack, chassis, slot, sub-slot while the equipment group contains network-element, board and port. With the requirement of GIS and on-demand domain controller selection raised, the equipment room becomes a new inventory object to be managed besides TMF classification.

Logically, the relationship between these inventory objects can be described by Figure 1 below:

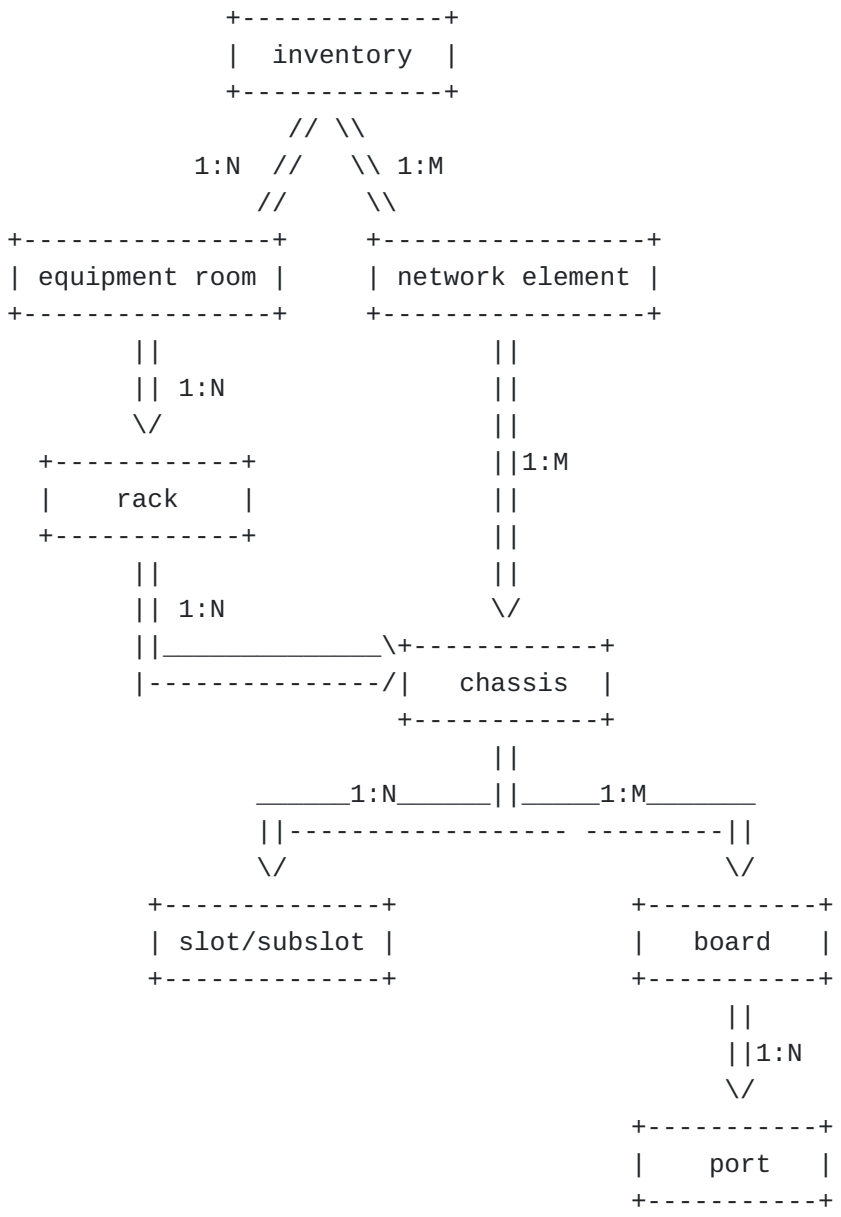

Figure 1: Relationship between inventory objects

In [RFC8348], rack, chassis, slot, sub-slot, board and port are defined as components of network elements with generic attributes.

Considering there are some special scenarios, there is no direct relationship between the rack and network element. In some cases, one network element contains multiple racks while in other cases one rack contains several shelves belonging to one or more network elements.

While [RFC8348] is used to manage the hardware of a single server (e.g., a network element), the Network Inventory YANG data model is used to retrieve the network hardware inventory information that a

controller discovers from all the network elements under its control.

However, the YANG data model defined in [RFC8348] has been used as a reference for defining the YANG network inventory data model. This approach can simplify the implementation of this network hardware inventory model when the controller uses the YANG data model defined in [RFC8348] to retrieve the hardware from the network elements under its control.

Note: review in future versions of this document whether to re-use definitions from [RFC8348] or use schema-mount.

```
+--ro network-inventory
   +--ro equipment-rooms
   |  +--ro equipment-room* [uuid]
   |     +--ro uuid        yang:uuid
   |     ...................................
   |     +--ro racks
   |        +--ro rack* [uuid]
   |           +--ro uuid           yang:uuid
   |           ....................................
   |           +--ro contained-chassis* [ne-ref component-ref]
   |              +--ro ne-ref?          leafref
   |              +--ro component-ref?   leafref
   +--ro network-elements
      +--ro network-element* [uuid]
         +--ro uuid          yang:uuid
         ...................................
         +--ro components
            +--ro component* [uuid]
               +--ro uuid              yang:uuid
               ....................................
```

2.1.1. Common Design for All Inventory Objects

For all the inventory objects, there are some common attributes existing. Such as:

Identifier: here we suggest to use uuid format which is widely implemented with systems. It is guaranteed to be globally unique.

Name: name is a human-readable label information which could be used to present on GUI. This name is suggested to be provided by server.

Alias: alias is also a human-readable label information which could be modified by user. It could also be present on GUI instead of name.

```
   Description: description is a human-readable information which could
   be also input by user. Description provides more detailed
   information to prompt users when performing maintenance operations.

   Location: location is a common management requirement of operators.
   This location could be an absolute position (e.g. mailing address),
   or a relative position (e.g. port index). Different types of
   inventory objects may require different types of position.

module: ietf-network-inventory
   +--ro network-inventory
      +--ro equipment-rooms
      |  +--ro equipment-room* [uuid]
      |     +--ro uuid           yang:uuid
      |     +--ro name?          string
      |     +--ro description?   string
      |     +--ro alias?         string
      |     +--ro location?      string
      |     ................................
      |     +--ro racks
      |        +--ro rack* [uuid]
      |           +--ro uuid                 yang:uuid
      |           +--ro name?                string
      |           +--ro description?         string
      |           +--ro alias?               string
      |           +--ro rack-location
      |           |  +--ro equipment-room-name?   leafref
      |           |  +--ro row-number?            uint32
      |           |  +--ro column-number?         uint32
      |           ................................
      +--ro network-elements
         +--ro network-element* [uuid]
            +--ro uuid             yang:uuid
            +--ro name?            string
            +--ro description?     string
            +--ro alias?           string
            +--ro ne-location
            |  +--ro equipment-room-name*   leafref
            ................................
            +--ro components
               +--ro component* [uuid]
                  +--ro uuid                    yang:uuid
                  +--ro name?                   string
                  +--ro description?            string
                  +--ro alias?                  string
                  +--ro location                string
                  ................................
```

2.1.2. Reference from RFC8348

The YANG data model for network hardware inventory mainly follows the same approach of [RFC8348] and reports the network hardware inventory as a list of components with different types (e.g., chassis, module, port).

```
+--ro components
   +--ro component* [uuid]
      +--ro uuid              yang:uuid
      +--ro name?             string
      +--ro description?      string
      +--ro class?            identityref
      +--ro contained-child*  -> ../uuid
      +--ro hardware-rev?     string
      +--ro firmware-rev?     string
      +--ro software-rev?     string
      +--ro serial-num?       string
      +--ro mfg-name?         string
      +--ro asset-id?         string
      +--ro is-fru?           boolean
      +--ro mfg-date?         yang:date-and-time
      +--ro uri*              inet:uri
```

For state data like admin-state, oper-state and so on, we consider they are related to device hardware management and not hardware inventory. Therefore, they are outside of scope of this document. Same for the sensor-data, they should be defined in some other performance monitoring data models instead of inventory data model.

We re-defined some attributes listed in [RFC8348], based on some integration experience for network wide inventory data.

2.1.3. Changes with respect to RFC8348

2.1.3.1. New Parent Identifiers' Reference

[RFC8348] provided a "parent-ref" attribute, which was an identifier reference to its parent component. When the MDSC or OSS systems want to find this component's grandparent or higher level component in the hierarchy, they need to retrieve this parent-ref step by step. To reduce this iterative work, we decided to provide a list of hierarchical parent components' identifier references.

```
+--ro components
   +--ro component* [uuid]
      .................................
      +--ro parent-references
      |  +--ro equipment-room-uuid?    leafref
      |  +--ro ne-uuid?                leafref
      |  +--ro rack-uuid?              leafref
      |  +--ro component-references
      |     +--ro component-reference* [index]
      |        +--ro index    uint8
      |        +--ro class?   leafref
      |        +--ro uuid?    leafref
      .................................
```

The hierarchical components' identifier could be found by the "component-reference" list. The "index" attribute is used to order the list by the hierarchical relationship from topmost component (with the "index" set to 0) to bottom component.

2.1.3.2. Component-Specific Info Design

According to the management requirements from operators, some important attributes are not defined in [RFC8348]. These attributes could be component-specific and are not suitable to define under the component list node. So, we defined a choice-case structure for this component-specific extension, as follows:

```
+--ro components
   +--ro component* [uuid]
      .................................
      +--ro (component-class)?
         +--:(chassis)
         |  +--ro chassis-specific-info
         +--:(container)
         |  +--ro slot-specific-info
         +--:(module)
         |  +--ro board-specific-info
         +--:(port)
            +--ro port-specific-info
      .................................
```

Note: The detail of each *-specific-info YANG container is still under discussion, and the leaf attributes will be defined in future.

2.1.3.3. Part Number

According to the description in [RFC8348], the attribute named "model-name" under the component, is preferred to have a customer-

visible part number value. "Model-name" is not straightforward to understand and we suggest to rename it as "part-number" directly.

```
+--ro components
   +--ro component* [uuid]
      ...................................
      +--ro part-number?           string
      ...................................
```

2.1.4. Equipment Room

Note: add some more attributes about equipment room in the future.

2.1.5. Rack

Besides the common attributes mentioned in above section, rack could have some specific attributes, such as appearance-related attributes and electricity-related attributes. The height, depth and width are described by the figure below (please consider that the door of the rack is facing the user):

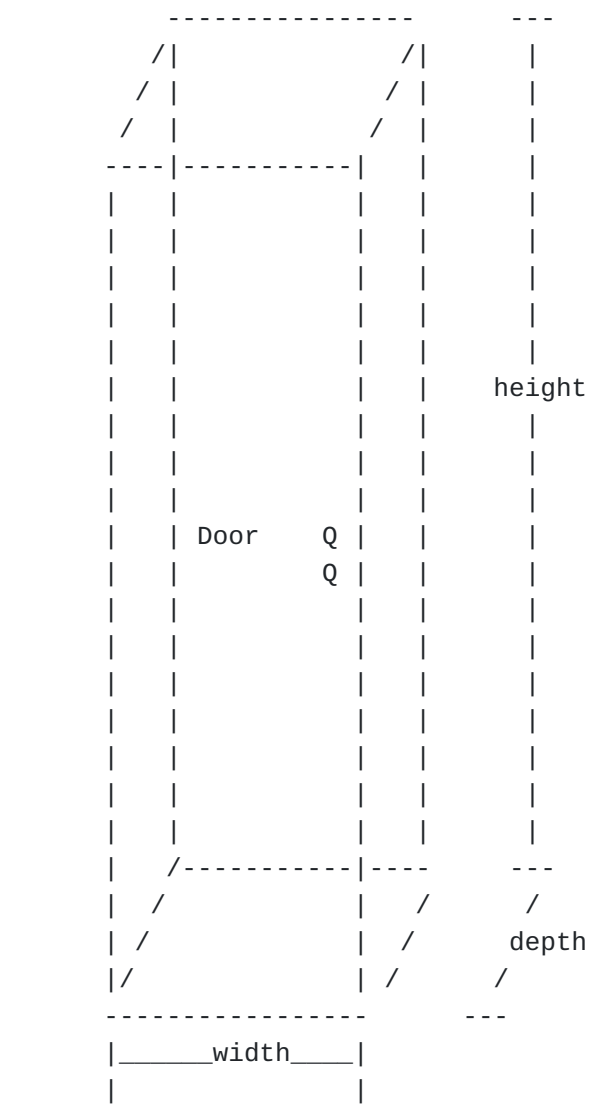

Figure 2: height, width and depth of rack

The rack attributes include:

```
+--ro racks
   +--ro rack* [uuid]
      .................................
      +--ro height?              uint16
      +--ro width?               uint16
      +--ro depth?               uint16
      +--ro max-voltage?         uint16
      .................................
```

Max-voltage: the maximum voltage supported by the rack.

2.1.6. Network Element

We consider that some of the attributes defined in [RFC8348] for components are also applicable for network element. These attributes include:

```
   +--ro network-elements
      +--ro network-element* [uuid]
         ...................................
         +--ro hardware-rev?    string
         +--ro firmware-rev?    string
         +--ro software-rev?    string
         +--ro mfg-name?        string
         +--ro mfg-date?        yang:date-and-time
         +--ro part-number?     string
         +--ro serial-number?   string
         +--ro product-name?    string
         ...................................
```

Note: Not all the attributes defined in [RFC8348] are applicable for network element. And there could also be some missing attributes which are not recognized by [RFC8348]. More extensions could be introduced in later revisions after the missing attributes are fully discussed.

2.2. Efficiency Issue

During the integration with OSS in some operators, some efficiency/scalability concerns have been discovered when synchronizing network inventory data for big networks. More discussions are needed to address these concerns.

Considering that relational databases are widely used by traditional OSS systems and also by some network controllers, the inventory objects are most likely to be saved in different tables. With the model defined in current draft, when doing a full synchronization, network controller needs to convert all inventory objects of each NE into component objects and combine them together into a single list, and then construct a response and send to OSS or MDSC. The OSS or MDSC needs to classify the component list and divide them into different groups, in order to save them in different tables. The combining-regrouping steps are impacting the network controller & OSS/MDSC processing, which may result in efficiency/scalability limitations in large scale networks.

An alternative YANG model structure, which defines the inventory objects directly, instead of defining generic components, has also been analyzed. However, also with this model, there still could be some scalability limitations when synchronizing full inventory

resources in large scale of networks. This scalability limitation is caused by the limited transmission capabilities of HTTP protocol. We think that this scalability limitation should be solved at protocol level rather than data model level.

The model proposed by this draft is designed to be as generic as possible so to cover future special types of inventory objects that could be used in other technologies, that have not been identified yet. If the inventory objects were to be defined directly with fixed hierarchical relationships in YANG model, this new type of inventory objects needs to be manually defined, which is not a backward compatible change and therefore is not an acceptable approach for implementation. With a generic model, it is only needed to augment a new component class and extend some specific attributes for this new inventory component class, which is more flexible. We consider that this generic data model, enabling a flexible and backward compatible approach for other technologies, represents the main scope of this draft. Solution description to efficiency/scalability limitations mentioned above is considered as out-of-scope.

2.3. Some Other Considerations

Note: review in future versions of this document whether the component list should be under the network-inventory instead of the network-element container.

Note that in [RFC8345], topology and inventory are two subsets of network information. However, considering the complexity of the existing topology models and having a better extension capability, we define a separate root for the inventory model. We will consider some other ways to do some associations between the topology model and inventory model in the future.

Note: review in future versions of this document whether network hardware inventory should be defined as an augmentation of the network model defined in [RFC8345] instead of under a new network-inventory root.

The proposed YANG data model has been analysed so far to cover the requirements and use cases for Optical Network Inventory.

Further analysis of requirements and use cases is needed to extend the applicability of this YANG data model to other types of networks (IP and microwave) and to identify which aspects are generic and which aspects are technology-specific for optical.

3. Network Hardware Inventory Tree Diagram

Figure 3 below shows the tree diagram of the YANG data model defined in module ietf-network-inventory.yang (Section 4).

```
module: ietf-network-inventory
  +--ro network-inventory
     +--ro equipment-rooms
     |  +--ro equipment-room* [uuid]
     |     +--ro uuid           yang:uuid
     |     +--ro name?          string
     |     +--ro description?   string
     |     +--ro alias?         string
     |     +--ro location?      string
     |     +--ro racks
     |        +--ro rack* [uuid]
     |           +--ro uuid                 yang:uuid
     |           +--ro name?                string
     |           +--ro description?         string
     |           +--ro alias?               string
     |           +--ro rack-location
     |           |  +--ro equipment-room-name?   leafref
     |           |  +--ro row-number?            uint32
     |           |  +--ro column-number?         uint32
     |           +--ro rack-number?         uint32
     |           +--ro height?              uint16
     |           +--ro width?               uint16
     |           +--ro depth?               uint16
     |           +--ro max-voltage?         uint16
     |           +--ro contained-chassis* [ne-ref component-ref]
     |              +--ro ne-ref           leafref
     |              +--ro component-ref    leafref
     +--ro network-elements
        +--ro network-element* [uuid]
           +--ro uuid             yang:uuid
           +--ro name?            string
           +--ro description?     string
           +--ro alias?           string
           +--ro ne-location
           |  +--ro equipment-room-name*   leafref
           +--ro hardware-rev?    string
           +--ro firmware-rev?    string
           +--ro software-rev?    string
           +--ro mfg-name?        string
           +--ro mfg-date?        yang:date-and-time
           +--ro part-number?     string
           +--ro serial-number?   string
           +--ro product-name?    string
           +--ro components
              +--ro component* [uuid]
                 +--ro uuid                          yang:uuid
                 +--ro name?                         string
                 +--ro description?                  string
                 +--ro alias?                        string
```

```
+--ro location?                       string
+--ro class?                          identityref
+--ro contained-child*                -> ../uuid
+--ro parent-rel-pos?                 int32
+--ro parent-references
|  +--ro equipment-room-uuid?    leafref
|  +--ro ne-uuid?                leafref
|  +--ro rack-uuid?              leafref
|  +--ro component-references
|     +--ro component-reference* [index]
|        +--ro index    uint8
|        +--ro class?   -> ../../../../class
|        +--ro uuid?    -> ../../../../uuid
+--ro hardware-rev?                   string
+--ro firmware-rev?                   string
+--ro software-rev?                   string
+--ro serial-num?                     string
+--ro mfg-name?                       string
+--ro part-number?                    string
+--ro asset-id?                       string
+--ro is-fru?                         boolean
+--ro mfg-date?
|       yang:date-and-time
+--ro uri*                            inet:uri
+--ro (component-class)?
   +--:(chassis)
   |  +--ro chassis-specific-info
   +--:(container)
   |  +--ro slot-specific-info
   +--:(module)
   |  +--ro board-specific-info
   +--:(port)
      +--ro port-specific-info
```

Figure 3: Network inventory tree diagram

4. YANG Model for Network Hardware Inventory

```
<CODE BEGINS> file "ietf-network-inventory@2022-07-11.yang"

module ietf-network-inventory {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-inventory";
  prefix ni;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC6991: Common YANG Data Types.";
  }

  import iana-hardware {
    prefix ianahw;
    reference
      "RFC 8348: A YANG Data Model for Hardware Management.";
  }

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF CCAMP Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/ccamp/>
     WG List:  <mailto:ccamp@ietf.org>

     Editor:   Chaode Yu
               <yuchaode@huawei.com>

     Editor:   Italo Busi
               <italo.busi@huawei.com>

     Editor:   Aihua Guo
               <aihuaguo.ietf@gmail.com>

     Editor:   Sergio Belotti
               <sergio.belotti@nokia.com>

     Editor:   Jean-Francois Bouquier
               <jeff.bouquier@vodafone.com>

     Editor:   Fabio Peruzzini
               <fabio.peruzzini@telecomitalia.it>";

  description
    "This module defines a model for retrieving network inventory.
```

```
  The model fully conforms to the Network Management
  Datastore Architecture (NMDA).

  Copyright (c) 2022 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 Revised BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

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

  The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
  NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
  'MAY', and 'OPTIONAL' in this document are to be interpreted as
  described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
  they appear in all capitals, as shown here.";

// 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 2022-07-11 {
  description
    "version 3.0.0";
  reference
    "draft-yg3bp-ccamp-inventory-yang-01: A YANG Data
    Model for Network Inventory.";
}

revision 2022-03-04 {
  description
    "version 3.0.0";
  reference
    "draft-yg3bp-ccamp-inventory-yang-00: A YANG Data
    Model for Network Inventory.";
}

revision 2021-11-09 {
  description
    "version 2.0.0";
  reference
    "draft-yg3bp-ccamp-optical-inventory-yang-00: A YANG Data
    Model for Optical Network Inventory.";
}
```

```
revision 2021-10-25 {
  description
    "Initial revision.";
  reference
    "draft-yg3bp-ccamp-optical-inventory-yang-00: A YANG Data
    Model for Optical Network Inventory.";
}

container network-inventory {
  config false;
  description
    "The top-level container for the network inventory
    information.";
  uses equipment-rooms-grouping;
  uses network-elements-grouping;
}

grouping common-entity-attributes {
  description
    "A set of attributes which are common to all the entities
    (e.g., component, equipment room) defined in this module.";
  leaf uuid {
    type yang:uuid;
    description
      "Uniquely identifies an entity (e.g., component).";
  }
  leaf name {
    type string;
    description
      "A name for an entity (e.g., component), as specified by
      a network manager, that provides a non-volatile 'handle'
      for the entity and that can be modified anytime during the
      entity lifetime.

      If no configured value exists, the server MAY set the value
      of this node to a locally unique value in the operational
      state.";
  }
  leaf description {
    type string;
    description "a textual description of inventory object";
  }
  leaf alias {
    type string;
    description
    "a alias name of inventory objects. This alias name can be
    specified by network manager.";
  }
```

```
}

grouping network-elements-grouping {
  description
    "The attributes of the network elements.";
  container network-elements {
    description
      "The container for the list of network elements.";
    list network-element {
      key uuid;
      description
        "The list of network elements within the network.";
      uses common-entity-attributes;
      container ne-location {
        description
          "To be added.";
        leaf-list equipment-room-name {
          type leafref {
            path "/ni:network-inventory/ni:equipment-rooms/" +
                 "ni:equipment-room/ni:name";
          }
          description
            "Names of equipment rooms where the NE is located.
            Please note that a NE could be located in several
            equipment rooms.";
        }
      }
      uses ne-specific-info-grouping;
      uses components-grouping;
    }
  }
}

grouping ne-specific-info-grouping {
  description
    "To be added.";
  leaf hardware-rev {
    type string;
    description
      "The vendor-specific hardware revision string for the NE.";
  }
  leaf firmware-rev {
    type string;
    description
      "The vendor-specific firmware revision string for the NE.";
  }
  leaf software-rev {
    type string;
    description
```

```
      "The vendor-specific software revision string for the NE.";
  }
  leaf mfg-name {
    type string;
    description "The name of the manufacturer of this NE";
  }
  leaf mfg-date {
    type yang:date-and-time;
    description "The date of manufacturing of the NE.";
  }
  leaf part-number {
    type string;
    description
      "The vendor-specific model name identifier string associated
       with this NE.  The preferred value is the customer-visible
       part number, which may be printed on the NE itself.";
  }
  leaf serial-number {
    type string;
    description
      "The vendor-specific serial number string for the NE";
  }
  leaf product-name {
    type string;
    description
      "indicates the vendor-spefic device type infomation.";
  }
}

grouping equipment-rooms-grouping {
  description
    "The attributes of the equipment rooms.";
  container equipment-rooms {
    description
      "The container for the list of equipment rooms.";
    list equipment-room {
      key uuid;
      description
        "The list of equipment rooms within the network.";
      uses common-entity-attributes;
      leaf location {
        type string;
        description
          "compared with the location information of the other
          inventory objects, a GIS address is preferred for
          equipment room";
      }
      container racks {
        description
```

```
          "To be added.";
        list rack {
          key uuid;
          description
            "The list of racks within an equipment room.";
          uses common-entity-attributes;
          uses rack-specific-info-grouping;
          list contained-chassis {
            key "ne-ref component-ref";
            description
              "The list of chassis within a rack.";
            leaf ne-ref {
              type leafref {
                path "/ni:network-inventory/ni:network-elements"
                + "/ni:network-element/ni:uuid";
              }
              description
                "The reference to the network element containing
                the chassis component.";
            }
            leaf component-ref {
              type leafref {
                path "/ni:network-inventory/ni:network-elements"
                + "/ni:network-element[ni:uuid"
                + "=current()/../ne-ref]/ni:components"
                + "/ni:component/ni:uuid";
              }
              description
                "The reference to the chassis component within
                the network element and contained by the rack.";
            }
          }
        }
      }
    }
  }
}

grouping rack-specific-info-grouping {
  description
    "To be added.";
  container rack-location {
    description
      "To be added.";
    leaf equipment-room-name {
      type leafref {
        path "/ni:network-inventory/ni:equipment-rooms"
        + "/ni:equipment-room/ni:name";
      }
```

```
      description
        "Name of equipment room where this rack is located.";
      }
      leaf row-number {
        type uint32;
        description
          "Identifies the row within the equipment room where
          the rack is located.";
      }
      leaf column-number {
        type uint32;
        description
          "Identifies the physical location of the rack within
          the column.";
      }
    }
    leaf rack-number {
      type uint32;
      description
        "An integer identifier of rack.";
    }
    leaf height {
      type uint16;
      units millimeter;
      description
        "To be added.";
    }
    leaf width {
      type uint16;
      units millimeter;
      description
        "To be added.";
    }
    leaf depth {
      type uint16;
      units millimeter;
      description
        "To be added.";
    }
    leaf max-voltage {
      type uint16;
      units volt;
      description
        "The maximum voltage could be supported by the rack.";
    }
  }

  grouping components-grouping {
    description
```

```
  "The attributes of the hardware components.";
container components {
  description
    "The container for the list of components.";
  list component {
    key uuid;
    description
      "The list of components within a network element.";
    uses common-entity-attributes;
    leaf location {
      type string;
      description
        "To be added.

        In optical transport network, the location string is
        using the following pattern:
          '/ne=<nw-ne-name>[/r=<r_index>][/sh=<sh_index>
          [/s_sh=<s_sh_index> ...]][[/sl=<sl_index>
          [/s_sl=<s_sl_index> ...]][/p=<p_index> …]]'
        ";
    }
    leaf class {
      type identityref {
        base ianahw:hardware-class;
      }
      description
        "An indication of the general hardware type of the
         component.";
      reference
        "RFC 8348: A YANG Data Model for Hardware Management.";
    }
    leaf-list contained-child {
      type leafref {
        path "../ni:uuid";
      }
      description
        "The child components' identifier that are physically
        contained by this component.";
    }
    leaf parent-rel-pos {
      type int32 {
        range "0 .. 2147483647";
      }
      description
        "To be added.";
      reference
        "RFC 6933: Entity MIB (Version 4) -
                   entPhysicalParentRelPos";
    }
```

```
container parent-references {
  description
    "To be added.";
  leaf equipment-room-uuid {
    type leafref {
      path "/ni:network-inventory/ni:equipment-rooms/" +
           "ni:equipment-room/ni:uuid";
    }
    description
      "To be added.";
  }
  leaf ne-uuid {
    type leafref {
      path "/ni:network-inventory/ni:network-elements/" +
           "ni:network-element/ni:uuid";
    }
    description
      "To be added.";
  }
  leaf rack-uuid {
    type leafref {
      path "/ni:network-inventory/ni:equipment-rooms/" +
           "ni:equipment-room/ni:racks/ni:rack/ni:uuid";
    }
    description
      "To be added.";
  }
  container component-references {
    description
      "To be added.";
    list component-reference {
      key index;
      description
        "this list object is used to indicate its
        hierarchial parent components' identifier.
        This hierarchial relation can be found by index
        parameter. The topest parent component should be
        0-index.";
      leaf index {
        type uint8;
        description
          "To be added.";
      }
      leaf class {
        type leafref {
          path "../../../../ni:class";
        }
        description
          "To be added.";
```

```
      }
      leaf uuid {
        type leafref {
          path "../../../../ni:uuid";
        }
        description
          "To be added.";
      }
    }
  }
}
leaf hardware-rev {
  type string;
  description
    "The vendor-specific hardware revision string for the
     component.  The preferred value is the hardware revision
     identifier actually printed on the component itself (if
     present).";
  reference
    "RFC 6933: Entity MIB (Version 4) -
               entPhysicalHardwareRev";
}
leaf firmware-rev {
  type string;
  description
    "The vendor-specific firmware revision string for the
     component.";
  reference
    "RFC 6933: Entity MIB (Version 4) -
               entPhysicalFirmwareRev";
}
leaf software-rev {
  type string;
  description
    "The vendor-specific software revision string for the
     component.";
  reference
    "RFC 6933: Entity MIB (Version 4) -
               entPhysicalSoftwareRev";
}
leaf serial-num {
  type string;
  description
    "The vendor-specific serial number string for the
     component.  The preferred value is the serial number
     string actually printed on the component itself (if
     present).";
  reference
    "RFC 6933: Entity MIB (Version 4) -
```

```
    entPhysicalSerialNum";
}
leaf mfg-name {
  type string;
  description
    "The name of the manufacturer of this physical component.
     The preferred value is the manufacturer name string
     actually printed on the component itself (if present).

     Note that comparisons between instances of the
     'model-name', 'firmware-rev', 'software-rev', and
     'serial-num' nodes are only meaningful amongst
     components with the same value of 'mfg-name'.

     If the manufacturer name string associated with the
     physical component is unknown to the server, then this
     node is not instantiated.";
  reference
    "RFC 6933: Entity MIB (Version 4) - entPhysicalMfgName";
}
leaf part-number {
  type string;
  description
    "The vendor-specific model name identifier string
     associated with this physical component.  The preferred
     value is the customer-visible part number, which may be
     printed on the component itself.

     If the model name string associated with the physical
     component is unknown to the server, then this node is
     not instantiated.";
  reference
    "RFC 6933: Entity MIB (Version 4) -
    entPhysicalModelName";
}
leaf asset-id {
  type string;
  description
    "This node is a user-assigned asset tracking identifier
     for the component.

     A server implementation MAY map this leaf to the
     entPhysicalAssetID MIB object.  Such an implementation
     needs to use some mechanism to handle the differences in
     size and characters allowed between this leaf and
     entPhysicalAssetID.  The definition of such a mechanism
     is outside the scope of this document.";
  reference
    "RFC 6933: Entity MIB (Version 4) - entPhysicalAssetID";
```

```
      }
      leaf is-fru {
        type boolean;
        description
          "This node indicates whether or not this component is
           considered a 'field-replaceable unit' by the vendor.  If
           this node contains the value 'true', then this component
           identifies a field-replaceable unit.  For all components
           that are permanently contained within a
           field-replaceable unit, the value 'false' should be
           returned for this node.";
        reference
          "RFC 6933: Entity MIB (Version 4) - entPhysicalIsFRU";
      }
      leaf mfg-date {
        type yang:date-and-time;
        description
          "The date of manufacturing of the managed component.";
        reference
          "RFC 6933: Entity MIB (Version 4) - entPhysicalMfgDate";
      }
      leaf-list uri {
        type inet:uri;
        description
          "This node contains identification information about the
           component.";
        reference
          "RFC 6933: Entity MIB (Version 4) - entPhysicalUris";
      }
      uses component-specific-info-grouping;
    }
  }
}

grouping component-specific-info-grouping {
  description
    "In case if there are some missing attributes of component not
    defined by RFC8348. These attributes could be
    component-specific.
    Here we provide a extension structure for all the components
    we recognized. We will enrich these component specifc
    containers in the future.";
  choice component-class {
    description
      "To be added.";
    case chassis {
      when "./class = 'ianahw:chassis'";
      container chassis-specific-info {
        description
```

```
          "This container contains some attributes belong to
          chassis only.";
        uses chassis-specific-info-grouping;
      }
    }
    case container {
      when "./class = 'ianahw:container'";
      container slot-specific-info {
        description
          "This container contains some attributes belong to
          slot or sub-slot only.";
        uses slot-specific-info-grouping;
      }
    }
    case module {
      when "./ni:class = 'ianahw:module'";
      container board-specific-info {
        description
          "This container contains some attributes belong to
          board only.";
        uses board-specific-info-grouping;
      }
    }
    case port {
      when "./ni:class = 'ianahw:port'";
      container port-specific-info {
        description
          "This container contains some attributes belong to
          port only.";
        uses port-specific-info-grouping;
      }
    }
  //TO BE ADDED: transceiver
  }
}

grouping chassis-specific-info-grouping {
//To be enriched in the future.
  description
    "To be added.";
}

grouping slot-specific-info-grouping {
//To be enriched in the future.
  description
    "To be added.";
}

grouping board-specific-info-grouping {
```

```
  //To be enriched in the future.
    description
      "To be added.";
  }

  grouping port-specific-info-grouping {
  //To be enriched in the future.
    description
      "To be added.";
  }
}

<CODE ENDS>
```

Figure 4: Network inventory YANG module

5. Manageability Considerations

<Add any manageability considerations>

6. Security Considerations

<Add any security considerations>

7. IANA Considerations

<Add any IANA considerations>

8. References

8.1. Normative References

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

[TMF-MTOSI] TM Forum (TMF), "TMF MTOSI 4.0 Equipment Model", TMF SD2-20_EquipmentModel , 2008, <https://www.tmforum.org/resources/suite/mtosi-4-0/>.

8.2. Informative References

[I-D.ietf-teas-actn-poi-applicability] Peruzzini, F., Bouquier, J., Busi, I., King, D., and D. Ceccarelli, "Applicability of Abstraction and Control of Traffic Engineered Networks (ACTN) to Packet Optical Integration (POI)", Work in Progress, Internet-Draft, draft-ietf-teas-actn-poi-applicability-07, 10 July 2022, <https://www.ietf.org/archive/id/draft-ietf-teas-actn-poi-applicability-07.txt>.

[ONF_TR-547] Open Networking Foundation (ONF), "TAPI v2.1.3 Reference Implementation Agreement", ONF TR-547 TAPI RIA v1.0 , July 2020, <https://opennetworking.org/wp-content/uploads/2020/08/TR-547-TAPI-v2.1.3-Reference-Implementation-Agreement-1.pdf>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

Appendix A. Appendix

A.1. Comparison With Openconfig-platform Data Model

Since more and more devices can be managed by domain controller through OpenConfig, to avoid that our inventory data model cannot cover these devices' inventory data, we have compared our inventory data model with the openconfig-platform.yang which is the data model used to manage inventory information in OpenConfig.

Openconfig-platform data model is NE-level and uses a generic component concept to describe its inner devices and containers, which is similar to ietf-hardware model in [RFC8348]. Since we have also reused the component concept of [RFC8348] in our inventory data model, we can compare the component's attributes between openconfig-platform and our model directly , which is stated below:

Attributes in oc-platform	Attributes in our model	remark
name	name	
type	class	
id	uuid	
location	location	
description	description	
mfg-name	mfg-name	
mfg-date	mfg-date	
hardware-version	hardware-rev	
firmware-version	firmware-rev	
software-version	software-rev	
serial-no	serial-num	
part-no	part-number	
clei-code		TBD
removable	is-fru	
oper-status		state data
empty	contained-child?	If there is no contained child, it is empty.
parent	parent-references	
redundant-role		TBD
last-switchover-reason		state data
last-switchover-time		state data
last-reboot-reason		state data
last-reboot-time		state data
switchover-ready		state data

Attributes in oc-platform	Attributes in our model	remark
temperature		performance data
memory		performance data
allocated-power		TBD
used-power		TBD
pcie		alarm data
properties		TBD
subcomponents	contained-child	
chassis	chassis-specific-info	
port	port-specific-info	
power-supply		TBD
fan		Fan is considered as a specific board. And no need to define as a single component
fabric		TBD
storage		For Optical and IP technology, no need to manage storage on network element
cpu		For Optical and IP technology, no need to manage CPU on network element
integrated-circuit	board-specific-info	
backplane		Backplane is considered as a part of board. And no need to define as a single component
software-module		TBD
controller-card		Controller card is considered as a specific functional board. And no need to define as a single component

Table 2: Comparison between openconfig-platform and inventory data model

As it mentioned in Section 2.1.2 that state data and performance data are out of scope of our data model, it is same for alarm data and it should be defined in some other alarm data models separately. And for some component specific structures in openconfig-platform, we consider some of them can be contained by our existing structure, such as fan, backplane, and controller-card. And for some of them, there is no need to manage for operators, such as storage and cpu.

Mostly, our inventory data model can cover the attributes from OpenConfig.

Acknowledgments

The authors of this document would like to thank the authors of [I-D.ietf-teas-actn-poi-applicability] for having identified the gap and requirements to trigger this work.

This document was prepared using kramdown.

Authors' Addresses

Chaode Yu
Huawei Technologies

Email: yuchaode@huawei.com

Italo Busi
Huawei Technologies

Email: italo.busi@huawei.com

Aihua Guo
Futurewei Technologies

Email: aihuaguo.ietf@gmail.com

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com

Jean-Francois Bouquier
Vodafone

Email: jeff.bouquier@vodafone.com

Fabio Peruzzini
TIM

Email: fabio.peruzzini@telecomitalia.it

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Victor Lopez
Nokia

Email: victor.lopez@nokia.com