

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 January 2026

C. Yu
Huawei Technologies
S. Belotti
Nokia
J.-F. Bouquier
Vodafone
F. Peruzzini
FiberCop
P. Bedard
Cisco
22 July 2025

A Base YANG Data Model for Network Inventory
draft-ietf-ivy-network-inventory-yang-08

Abstract

This document defines a base YANG data model for network inventory. The scope of this base model is set to be application- and technology-agnostic. The base data model can be augmented with application- and technology-specific details.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://ietf-ivy-wg.github.io/network-inventory-yang/draft-ietf-ivy-network-inventory-yang.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-ivy-network-inventory-yang/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-ivy-wg/network-inventory-yang>.

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 23 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Editorial Note (To be removed by RFC Editor)	4
2. Terminology and Notations	5
2.1. Requirements Notations	5
2.2. Terminology	5
2.3. Tree Diagrams	8
2.4. YANG Prefixes	8
3. YANG Data Model for Network Inventory Overview	8
3.1. Common Design for All Inventory Objects	10
3.2. Network Element	11
3.3. Components	12
3.3.1. Hardware Components	12
3.3.2. Software Components	14
3.3.3. Breakout ports	14
3.4. Changes Since RFC 8348	14
3.4.1. Part Number	14
3.4.2. Component identifiers	15
4. Network Inventory Tree Diagram	15
5. YANG Data Model for Network Inventory	17
6. Manageability Considerations	28
7. Security Considerations	28
8. IANA Considerations	28
9. References	29
9.1. Normative References	29
9.2. Informative References	30

Appendix A. Comparison With Openconfig-platform Data Model . . .	31
Appendix B. Terminology of Container	34
Appendix C. Efficiency Issue	34
Appendix D. Examples of ports, transceivers and port breakouts	35
D.1. JSON Examples	41
Appendix E. Example of multi-chassis network elements	45
E.1. JSON Examples	49
Acknowledgments	58
Contributors	58
Authors' Addresses	59

1. Introduction

This document defines a base network inventory YANG data model that is application- and technology-agnostic. The base data model can be augmented to describe application- and technology-specific information. Please note that the usage of term "network inventory", in the context of this I-D, is to indicate that it is describing "network-wide" scope inventory information.

Network inventory is a fundamental functional block in the overall network management which was specified many years ago. Network inventory management is a critical part for ensuring that the network remains healthy (e.g., auditing to identify faulty elements), well-planned (e.g., identify assets to upgrade or to decommission), and maintained appropriately to meet the performance objectives. Also, network inventory management allows operators to keep track of which devices are deployed in their networks, including relevant embedded software and hardware versions.

Exposing standard interfaces to retrieve network elements capabilities as maintained in an inventory are key enablers for many applications. For example, [I-D.ietf-teas-actn-poi-applicability] identifies a gap about the lack of YANG data models that could be used at Abstraction and Control of TE Networks (ACTN) Multi-Domain Service Coordinator-Provisioning Network Controller Interface (MPI) level to report whole or partial network hardware inventory information available at domain controller level towards upper layer systems (e.g., Multi-Domain Service Coordinator (MDSC) or Operations Support Systems (OSS) layers).

It is key for operators to coordinate with the industry towards the use of a standard YANG data model for Network Inventory data instead of using vendors proprietary APIs.

[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 towards the network elements rather than at the northbound interface of a network controller (e.g., toward an application or another hierarchical network controller). However, the YANG data model defined in [RFC8348] has been used as a reference for defining the YANG network inventory data model presented in this document.

Network Inventory is a collection of data for network devices and their components managed by a specific management system. Per the definition of [RFC8969], the network inventory model is a network model.

This document defines one YANG module "ietf-network-inventory" in Section 5.

This base data model is application- and technology-agnostic (that is, valid for IP/MPLS, optical, and microwave networks as well as optical local loops, access networks, core networks, data centers, etc.) and can be augmented to include required application- and technology-specific inventory details together with specific hardware or software component's attributes.

The YANG data model defined in the document is scoped to cover the common use cases for Inventory but at network-wide level, covering both hardware and base software information.

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

The YANG data model defined in the document is intended to only report actual inventory data which includes both applied configuration data and state data of the network elements and components actually installed within the network.

1.1. Editorial Note (To be removed by RFC Editor)

Note to the RFC Editor: This section is to be removed prior to publication.

This document contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed.

Please apply the following replacements:

* XXXX --> the assigned RFC number for this I-D

2. Terminology and Notations

2.1. Requirements Notations

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.

2.2. Terminology

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

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

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

- * configuration data
- * state data

The following terms are defined in [RFC8342] and are not redefined here:

- * applied configuration

The following terms are defined in the description statements of the corresponding YANG identities, defined in [IANA_HW_YANG], and are not redefined here:

- * backplane
- * battery
- * container
- * cpu
- * chassis

- * fan
- * module
- * port
- * power supply
- * sensor
- * stack
- * storage device

Editors' Note: The chassis and port definition below needs to be moved to iana-hardware update

Chassis: A field replaceable equipment with a particular structural format and dimensions. A chassis can include spaces(called slots) to take cards.

Elsewhere, a chassis can be called shelf, sub-rack, etc.

Port: A component where networking traffic can be received and/or transmitted, e.g., by attaching networking cables.

In case of pluggable ports, the port may be empty when no transceiver module is plugged in.

Also, the document makes use of the following terms:

Network Inventory: A collection of data for network elements and their components with network-wide scope, managed by a specific management system.

Physical Network Element: An implementation or application specific group of components (e.g., hardware components).

Network Element: The generalization of the physical network element definition.

Hardware Component: The generalization of the hardware components defined in [IANA_HW_YANG] (e.g., backplane, battery, container, cpu, chassis, fan, module, port, power supply, sensor, stack, and storage device components).

The list of hardware components can be extended in future versions of [IANA_ENTITY_MIB] (and, consequently, of ([IANA_HW_YANG])).

Component: The generalization of the hardware component definition to include other inventory objects which can be managed, from an inventory perspective, like hardware components.

Card: A pluggable equipment with a particular structural format and dimensions which can be inserted into one or more slots (or sub-slots). A card can have spaces (called sub-slots) to take other cards.

Elsewhere, a card can be called board, module, circuit pack, etc..

Slot: A space in a chassis that can be equipped with one card, which may be chosen from a limited range of types of cards. A slot can be subdivided into smaller spaces (called sub-slots).

Physical interface: An interface associated to a port. A physical interface is always in the lowest layer of the interface stack.

Logical interface: An interface which is not associated to a port.

Editors' Note: check whether the definitions of physical and logical interfaces can be replaced by a normative reference to [RFC8343]

Breakout Port: A port is usually associated with a single physical interface. A breakout port is a port which is broken down and associated into multiple physical interfaces. Those physical interfaces can have the same or different speeds and the same or different number of breakout channels.

Trunk Port: A trunk port is a port which is associated with one and only physical interface.

Port Breakout: The configuration of a breakout port.

Note that interface channelization, when an interface (e.g., an Ethernet interface) is configured to support multiple logical sub-interfaces (e.g., VLAN interfaces), is different than port breakout and outside the scope of inventory models.

Breakout channel: An abstraction of the atomic resource elements into which a breakout port can be broken down: a physical interface can be associated with one or more breakout channels but no more than one physical interface can be associated with one breakout channel.

The physical elements abstracted as breakout channels are

implementation specific. Appendix D provides some examples of breakout ports configurations and implementations.

Container: A hardware component class that is capable of containing one or more removable physical entities (e.g., a slot in a chassis is containing a board).

Transceiver: A transceiver represents a transmitter/receiver (Tx/Rx) pair which is transmitting and receiving a signal from the media.

This definition generalizes the transceiver definition in [I-D.ietf-ccamp-optical-impairment-topology-yang] to model also non optical transceivers (e.g., electrical transceivers).

2.3. Tree Diagrams

The meanings of the symbols in the YANG tree diagrams are defined in [RFC8340].

2.4. YANG Prefixes

Table 1 list the prefixes of the modules that are used in this document.

Prefix	YANG Module	Reference
inet	ietf-inet-types	Section 4 of [RFC6991]
yang	ietf-yang-types	Section 3 of [RFC6991]
ianahw	iana-hardware	[IANA_HW_YANG]
nwi	ietf-network-inventory	RFC XXXX

Table 1: Prefixes and corresponding YANG modules

3. YANG Data Model for Network Inventory Overview

The base network inventory model, defined in this document, provides a list of network elements and of network element components:


```
+--rw network-inventory
  +--rw network-elements
    +--rw network-element* [ne-id]
      +--rw ne-id                string
      ...
    +--rw components
      +--rw component* [component-id]
        +--rw component-id      string
        ...
```

Figure 1: Overall Tree Structure

The network-inventory top level container has been defined to support reporting other types of network inventory objects, besides the network elements and network element components.

These additional types of network inventory objects can be defined, together with the associated YANG data model and the rationale for managing them as part of the network inventory, in other documents providing application- and technology-specific companion augmentation data models, such as [I-D.ietf-ivy-network-inventory-location]

The network element definition is generalized to support physical network elements and other types of components' groups that can be managed as physical network elements from an inventory perspective.

Physical network elements are usually devices such as hosts, gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management stations ([RFC1157]).

The "ne-type" is defined as a YANG identity to describe the type of the network element. This document defines only the "physical-network-element" identity.

Other types of network elements can be defined in other documents, together with the associated YANG identity and the rationale for managing them as network elements from an inventory perspective.

The component definition is also generalized to support any types of component inventory objects that can be managed as hardware components from an inventory perspective.

The data model for components defined in this document uses a list of components within each network element.

Different types of components can be distinguished by the class of component. The component "class" is defined as a union between the hardware class identity, defined in "iana-hardware", and the "non-hardware" identity, defined in this document.

Other types of components can be defined in other documents, together with the associated YANG identity and the rationale for managing them as components from an inventory perspective.

The identity definition of additional types of "ne-type" and "non-hardware" identity of component are outside the scope of this document and could be defined in application- and technology-specific companion augmentation data models, such as [I-D.ietf-ivy-network-inventory-software].

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

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 base inventory information that a controller discovers from all the network elements with network-wide scope 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 inventory model when the controller uses the YANG data model defined in [RFC8348] to retrieve the hardware from the network elements under its control.

3.1. Common Design for All Inventory Objects

For all the inventory objects, there are some common attributes Figure 2. Such as:

Identifier: The UUID format is used. Such identifiers are widely implemented with systems. It is guaranteed to be globally unique.

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

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

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

```

+--rw network-elements
  +--rw network-element* [ne-id]
    +--rw ne-id          string
    +--rw ne-type?       identityref
    +--rw uuid?          yang:uuid
    +--rw name?          string
    +--rw description?   string
    +--rw alias?         string
    ...
    +--rw components
      +--rw component* [component-id]
        +--rw component-id string
        +--rw uuid?       yang:uuid
        +--rw name?       string
        +--rw description? string
        +--rw alias?      string
        +--rw class       union
        ...

```

Figure 2: Common Attributes Between Network Elements and Components Subtree

3.2. Network Element

To be consistent with the component definition, some of the attributes defined in [RFC8348] for components are reused for network elements as shown in Figure 3.

```

+--rw network-elements
  +--rw network-element* [ne-id]
    ...
    +--rw hardware-rev?  string
    +--rw software-rev?  string
    +--rw mfg-name?      string
    +--rw mfg-date?     yang:date-and-time
    +--rw part-number?   string
    +--rw serial-number? string
    +--rw product-name?  string
    ...

```

Figure 3: Network Elements Subtree

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.

3.3. Components

The YANG data model for network 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, and port).

The component definition (Figure 4) is generalized to both hardware components and non-hardware components (e.g., software components).

```

+--rw components
  +--rw component* [component-id]
    +--rw component-id          string
    +--rw uuid?                 yang:uuid
    +--rw name?                 string
    +--rw description?          string
    +--rw alias?                string
    +--rw class                 union
    +--rw child-component-ref
    +--rw parent-rel-pos?       int32
    +--rw parent-component-ref
    +--rw hardware-rev?         string
    +--rw firmware-rev?        string
    +--rw software-rev?        string
    +--rw serial-num?          string
    +--rw mfg-name?            string
    +--rw part-number?         string
    +--rw asset-id?            string
    +--rw is-fru?              boolean
    +--rw mfg-date?            yang:date-and-time
    +--rw uri*                 inet:uri
```

Figure 4: Components Subtree

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

3.3.1. Hardware Components

Based on TMF classification in [TMF_SD2-20], hardware components 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.

Figure 5 describes the relationship between typical inventory objects in a physical network element.

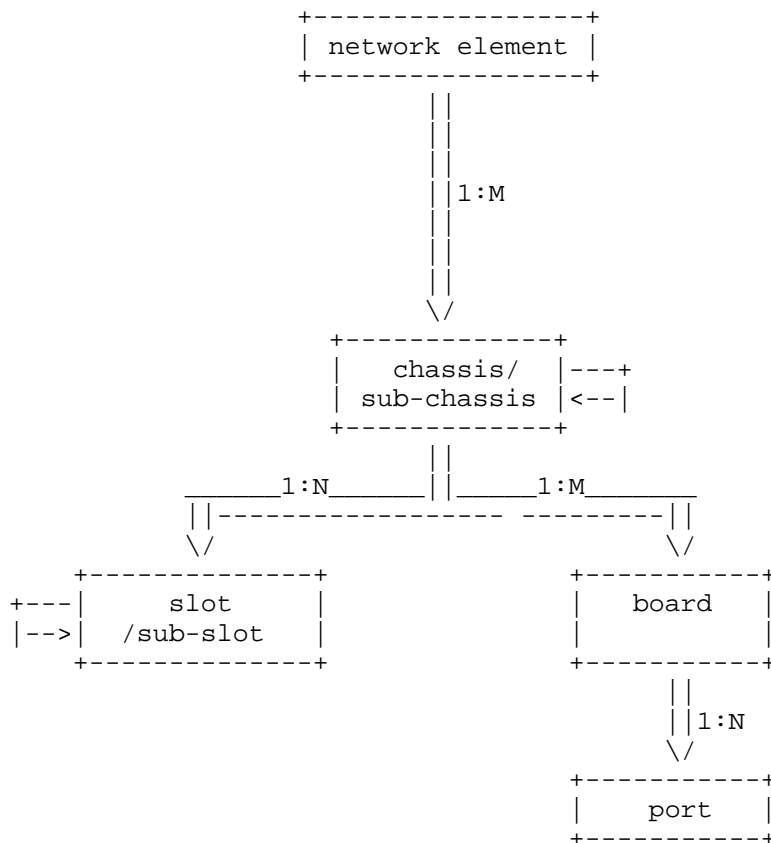


Figure 5: Relationship between typical inventory objects in physical network elements

The "iana-hardware" module [IANA_HW_YANG] defines YANG identities for the hardware component types in the IANA-maintained "IANA-ENTITY-MIB" registry.

Some of the definitions taken from [RFC8348] are based on the ENTITY-MIB [RFC6933].

Additional attributes of specific hardware, such as CPU, storage, port, or power supply are defined in the hardware extension.

3.3.2. Software Components

This document defines "software-rev" of NEs and components, which are basic software attributes of a Network Element.

The software and hardware components share the same attributes of the component and have similar replaceable requirements. Generally, the device also has other software data, for example, one or more software patch information.

The software components of other classes, such as platform software, BIOS, bootloader, and software patch information, are outside the scope of this document and defined other documents such as [I-D.ietf-ivy-network-inventory-software].

3.3.3. Breakout ports

The model defines the 'breakout-channels' presence container to indicate whether the port, which contains the transceiver module, can be configured as a breakout port or not.

The breakout channels represent the capability of the port to support port breakout, independently on how the port is configured (trunk or breakout).

It is assumed that a port which supports port breakout can be configured either as a trunk port or as a breakout port.

Reporting whether a port, which supports port breakout, is configured as a trunk or as a breakout port, is outside the scope of the base network inventory model. The model providing the mapping between the topology and the inventory models should provide sufficient information to identify how the port is configured and, in case of breakout configuration, which breakout channel is associated with which Link Termination Point (LTP), abstracting a device physical interface within the topology model.

3.4. Changes Since RFC 8348

This document re-defines some attributes listed in [RFC8348], based on some integration experience for network inventory data.

3.4.1. 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* [component-id]
    ...
    +--ro part-number?          string
    ...
```

3.4.2. Component identifiers

There are some use cases where the name of the components are assigned and changed by the operator. In these cases, the assigned names are also not guaranteed to be always unique.

In order to support these use cases, this model is not aligned with [RFC8348] in defining the component name as the key for the component list.

Instead the name is defined as an optional attribute and the component-id is defined as the key for the component list (in alignment with the approach followed for the network-element list).

4. Network Inventory Tree Diagram

Figure 6 below shows the tree diagram of the YANG data model defined in module "ietf-network-inventory" (Section 5).

```

module: ietf-network-inventory
  +--rw network-inventory
    +--rw network-elements
      +--rw network-element* [ne-id]
        +--rw ne-id string
        +--ro ne-type? identityref
        +--ro uuid? yang:uuid
        +--rw name? string
        +--rw description? string
        +--rw alias? string
        +--ro hardware-rev? string
        +--ro software-rev? string
        +--ro software-patch-rev* string
        +--ro mfg-name? string
        +--ro mfg-date? yang:date-and-time
        +--ro serial-number? string
        +--ro product-name? string
      +--rw components
        +--rw component* [component-id]
          +--rw component-id string
          +--ro class union
          +--ro uuid? yang:uuid
          +--rw name? string
          +--rw description? string
          +--rw alias? string
          +--ro hardware-rev? string
          +--ro software-rev? string
          +--ro software-patch-rev* string
          +--ro mfg-name? string
          +--ro mfg-date? yang:date-and-time
          +--ro serial-number? string
          +--ro product-name? string
          +--ro firmware-rev? string
          +--ro part-number? string
          +--ro asset-id? string
          +--ro is-fru? boolean
          +--ro uri* inet:uri
          +--ro child-component-ref
          +--ro parent-rel-pos? int32
          +--ro parent?
            | -> ../../component/component-id
          +--ro is-main? boolean
          +--ro breakout-channels!
            +--ro breakout-channel* [channel-id]
              +--ro channel-id uint8

```

Figure 6: Network inventory tree diagram

5. YANG Data Model for Network Inventory

```
<CODE BEGINS> file "ietf-network-inventory@2025-02-03.yang"
module ietf-network-inventory {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-inventory";
  prefix nwi;

  import iana-hardware {
    prefix ianahw;
    reference
      "https://www.iana.org/assignments/yang-parameters";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF IVY Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/ivy/>
    WG List:  <mailto:inventory-yang@ietf.org>

    Editor:   Chaode Yu
              <yuchaode@huawei.com>

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

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

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

    Editor:   Phil Bedard
              <phbedard@cisco.com>";
  description
    "This module defines a base model for retrieving network
    inventory."
```

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

Copyright (c) 2025 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.: update the date below with the date of RFC publication
// and remove this note.
```

```
revision 2025-07-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: A YANG Data Model for Network Inventory.";
}

/*
 * Identities
 */

identity non-hardware-component-class {
  description
    "Base identity for non hardware components (e.g., software
    components) in a managed device.";
}

identity ne-type {
  description
    "Base identity for Network Element (NE) types.";
}

identity ne-physical {
```

```
    base nwi:ne-type;
    description
        "A physical network element (NE). ";
}

/*
 * Editors' Note: This identity may need to be moved to
 *                 iana-hardware update
 */

identity transceiver-module {
    base ianahw:hardware-class;
    description
        "This identity is applicable if the hardware class is some sort
        of self-contained sub-system which contains one or more
        transceivers (e.g., optical or electrical transceivers).

        A transceiver-module component can only be contained by a
        port component.";
}

/*
 * Types
 */

typedef ne-ref {
    type leafref {
        path "/nwi:network-inventory/nwi:network-elements"
            + "/nwi:network-element/nwi:ne-id";
    }
    description
        "This type is intended to be used by data models that need to
        reference Network Element.";
}

/*
 * Groupings
 */

grouping port-ref {
    description
        "This grouping is intended to be used by data models that need
        to reference a port component within a Network Element.";
    leaf ne-ref {
        type nwi:ne-ref;
        description
            "The reference to the Network Element which contains the
            port to be referenced.";
    }
}
```

```
}
leaf port-ref {
  type leafref {
    path "/nwi:network-inventory/nwi:network-elements/"
      + "nwi:network-element[nwi:ne-id=current()/../ne-ref]"
      + "/nwi:components/nwi:component/nwi:component-id";
  }
  must "derived-from-or-self (/nwi:network-inventory/
    nwi:network-elements/nwi:network-element
    [nwi:ne-id=../ne-ref]/nwi:components/nwi:component
    [nwi:component-id=current()]/nwi:class, 'ianahw:port')";
  description
    "The reference to the port component.";
}
}

grouping channel-ref {
  description
    "This grouping is intended to be used by data models that need
    to reference one or more breakout channels within a
    transceivers module component.";
  leaf ne-ref {
    type nwi:ne-ref;
    description
      "The reference to the Network Element which contains the
      transceiver module to be referenced.";
  }
  leaf transceiver-module-ref {
    type leafref {
      path "/nwi:network-inventory/nwi:network-elements/"
        + "nwi:network-element[nwi:ne-id=current()/../ne-ref]"
        + "/nwi:components/nwi:component/nwi:component-id";
    }
    must "derived-from-or-self (/nwi:network-inventory/
      nwi:network-elements/nwi:network-element
      [nwi:ne-id=../ne-ref]/nwi:components/nwi:component
      [nwi:component-id=current()]/nwi:class,
      'nwi:transceiver-module')";
    description
      "The reference to the transceivers module component.";
  }
}
leaf-list channel-ref {
  type leafref {
    path "/nwi:network-inventory/nwi:network-elements"
      + "/nwi:network-element[nwi:ne-id=current()/../ne-ref]/"
      + "nwi:components/"
      + "nwi:component[nwi:component-id="
      + "current()/../transceiver-module-ref]/"
```

```
        + "nwi:breakout-channels/nwi:breakout-channel/"
        + "nwi:channel-id";
    }
    description
        "The references to the breakout channels.";
}
}

grouping component-attributes {
    description
        "The set of common attributes of a component.

        This grouping is intended also to be re-used by data models
        that need to report the common attributes of a component.";
    leaf component-id {
        type string;
        description
            "An identifier that uniquely identifies the component
            in a node.";
    }
    leaf class {
        type union {
            type identityref {
                base ianahw:hardware-class;
            }
            type identityref {
                base non-hardware-component-class;
            }
        }
        config false;
        mandatory true;
        description
            "The type of the component.";
    }
    uses ne-component-common-entity-attributes {
        refine "hardware-rev" {
            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";
        }
        refine "software-rev" {
            reference
                "RFC 6933: Entity MIB (Version 4) -
```

```
        entPhysicalSoftwareRev";
    }
    refine "mfg-name" {
        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-number' 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";
    }
    refine "mfg-date" {
        description
            "The date of manufacturing of the managed
            component.";
        reference
            "RFC 6933: Entity MIB (Version 4) -
            entPhysicalMfgDate";
    }
}
leaf firmware-rev {
    type string;
    config false;
    description
        "The vendor-specific firmware revision string for the
        component.";
    reference
        "RFC 6933: Entity MIB (Version 4) -
        entPhysicalFirmwareRev";
}
leaf part-number {
    type string;
    config false;
    description
        "The vendor-specific part number of the component
        type. It is expected that vendors assign unique part
        numbers to different component types within the
```

```
        scope of the vendor.";
    }
    leaf asset-id {
        type string;
        config false;
        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;
        config false;
        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-list uri {
        type inet:uri;
        config false;
        description
            "This node contains identification information about
            the component.";
        reference
            "RFC 6933: Entity MIB (Version 4) - entPhysicalUris";
    }
}

grouping basic-common-entity-attributes {
```

```
description
  "The set of basic attributes which are common to all the
   entities (e.g., component, network elements, location, passive
   entities) defined in this module and in other inventory
   modules.";
leaf uuid {
  type yang:uuid;
  config false;
  description
    "The Universally Unique Identifier of the entity
     (e.g., component).";
}
leaf name {
  type string;
  description
    "The name of the 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
    "The textual description of the entity (e.g., component).";
}
leaf alias {
  type string;
  description
    "The alias name of the entity (e.g., component). This alias
     name can be specified by network manager.";
}
}

grouping ne-component-common-entity-attributes {
  description
    "The set of attributes which are common to all the entities
     (e.g., component, network elements) defined in this module.";
  uses basic-common-entity-attributes;
  leaf hardware-rev {
    type string;
    config false;
    description
      "The vendor-specific hardware revision string for the entity
       (e.g., component).";
```



```
}
leaf software-rev {
    type string;
    config false;
    description
        "The vendor-specific software revision string for the entity
        (e.g., component).";
}
leaf-list software-patch-rev {
    type string;
    config false;
    description
        "The vendor-specific patch software revision string for the
        entity (e.g., component).";
}
leaf mfg-name {
    type string;
    config false;
    description
        "The name of the manufacturer of this entity
        (e.g., component).";
}
leaf mfg-date {
    type yang:date-and-time;
    config false;
    description
        "The date of manufacturing of the entity (e.g., component).";
}
leaf serial-number {
    type string;
    config false;
    description
        "The vendor-specific serial number of the the entity
        (e.g., component) instance. It is expected that vendors
        assign unique serial numbers to different network element
        instances within the scope of the product name.";
}
leaf product-name {
    type string;
    config false;
    description
        "The vendor-specific and human-interpretable string
        describing the entity (e.g., component) type. It is expected
        that vendors assign unique product names to different entity
        (e.g., component) types within the scope of the vendor.";
}
}
```

```
/*
 * Data Nodes
 */

container network-inventory {
  description
    "Top-level container for network inventory.";
  container network-elements {
    description
      "The top-level container for the list of network elements
       within the network.";
    list network-element {
      key "ne-id";
      description
        "The list of network elements within the network.";
      leaf ne-id {
        type string;
        description
          "An identifier that uniquely identifies the NE in
           a network.";
      }
      leaf ne-type {
        type identityref {
          base nwi:ne-type;
        }
        default "nwi:ne-physical";
        config false;
        description
          "The NE type.";
      }
    }
    uses ne-component-common-entity-attributes;
    container components {
      description
        "The top-level container for the list of components
         within a network element.";
      list component {
        key "component-id";
        description
          "The list of components within a network element.";
        uses component-attributes;
        container child-component-ref {
          config false;
          description
            "A placeholder for adding the reference to child
             component(s): to further discuss whether to define
             a child leaf-list as RFC 8348 or a list of
             sub-components as openconfig-platform.";
        }
      }
    }
  }
}
```

```
leaf parent-rel-pos {
  type int32 {
    range "0 .. 2147483647";
  }
  config false;
  description
    "The relative position with respect to the parent
    component among all the sibling components.";
  reference
    "RFC 6933: Entity MIB (Version 4) -
    entPhysicalParentRelPos";
}
leaf parent {
  type leafref {
    path "../../component/component-id";
    require-instance false;
  }
  config false;
  description
    "The identifier of the component that physically
    contains this component.

    If this leaf is not instantiated, it indicates that
    this component is not contained in any other
    component.

    In the event that a physical component is contained
    by more than one physical component
    (e.g., double-wide modules), this node contains the
    identifier of one of these components.
    An implementation MUST use the same name every time
    this node is instantiated.";
  reference
    "RFC 6933: Entity MIB (Version 4) -
    entPhysicalContainedIn";
}
leaf is-main {
  when "derived-from-or-self(..nwi:class, "
    + "'ianahw:chassis')";
  type boolean;
  config false;
  description
    "This node indicates whether the chassis is taking or
    not the 'main' role.

    This node is applicable only to scenarios where
    there are chassis components which can take the
    'main' role (e.g., multi-chassis network elements),
```

```

        otherwise it is omitted.";
    }
    container breakout-channels {
        when "derived-from-or-self(..nwi:class, "
            + "'nwi:transceiver-module')";
        presence
            "When present, it indicates that port breakout is
                supported.";
        config false;
        description
            "Top level container for the list of breakout
                channels supported by the transceivers module.";
        list breakout-channel {
            key "channel-id";
            description
                "The list of breakout channels supported by the
                    transceivers module.";
            leaf channel-id {
                type uint8;
                description
                    "An identifier that uniquely identifies the
                        breakout channel within the transceiver
                            module.";
            }
        }
    }
}
}
}
}
}
}
}
<CODE ENDS>
```

Figure 7: Network inventory YANG module

6. Manageability Considerations

<Add any manageability considerations>

7. Security Considerations

<Add any security considerations>

8. IANA Considerations

<Add any IANA considerations>

9. References

9.1. Normative References

- [IANA_ENTITY_MIB]
IANA, "IANA-ENTITY-MIB", n.d.,
<<https://www.iana.org/assignments/ianaentity-mib/ianaentity-mib.xhtml>>.
- [IANA_HW_YANG]
IANA, "iana-hardware YANG Module", n.d.,
<<https://www.iana.org/assignments/iana-hardware/iana-hardware.xhtml>>.
- [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/rfc/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/rfc/rfc6241>>.
- [RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <<https://www.rfc-editor.org/rfc/rfc6933>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/rfc/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/rfc/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/rfc/rfc8174>>.
- [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/rfc/rfc8342>>.

- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/rfc/rfc8343>>.
- [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/rfc/rfc8348>>.
- [TMF_SD2-20]
TM Forum, "SD2-20_Equipment Model", TMF MTOSI 4.0, Network Resource Fulfilment (NRF), SD2-20 , May 2008, <<https://www.tmforum.org/resources/suite/mtosi-4-0/>>.

9.2. Informative References

- [I-D.ietf-ccamp-optical-impairment-topology-yang]
Beller, D., Le Rouzic, E., Belotti, S., Galimberti, G., and I. Busi, "A YANG Data Model for Optical Impairment-aware Topology", Work in Progress, Internet-Draft, draft-ietf-ccamp-optical-impairment-topology-yang-19, 7 July 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-ccamp-optical-impairment-topology-yang-19>>.
- [I-D.ietf-ivy-network-inventory-location]
Wu, B., Belotti, S., Bouquier, J., Peruzzini, F., and P. Bedard, "A YANG Data Model for Network Inventory Location", Work in Progress, Internet-Draft, draft-ietf-ivy-network-inventory-location-03, 7 July 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-ivy-network-inventory-location-03>>.
- [I-D.ietf-ivy-network-inventory-software]
Wu, B., Zhou, C., Wu, Q., and M. Boucadair, "A YANG Network Data Model of Network Inventory Software Extensions", Work in Progress, Internet-Draft, draft-ietf-ivy-network-inventory-software-01, 10 June 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-ivy-network-inventory-software-01>>.
- [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-15, 4 July 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-teas-actn-poi-applicability-15>>.

[I-D.ygb-ivy-passive-network-inventory]

Yu, C., Guo, A., Busi, I., Boroon, M., Belotti, S., van caenegem, T., S., S. 1., B., S., Davis, N., Tilocca, M., Peters, B., Yoon, B. Y., LIUYUCONG, Zhao, Y., and A. Sakalabhaktula, "A YANG Data Model for Passive Network Inventory", Work in Progress, Internet-Draft, draft-ygb-ivy-passive-network-inventory-02, 7 July 2025, <<https://datatracker.ietf.org/doc/html/draft-ygb-ivy-passive-network-inventory-02>>.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <<https://www.rfc-editor.org/rfc/rfc1157>>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/rfc/rfc7951>>.

[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/rfc/rfc8340>>.

[RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/rfc/rfc8969>>.

Appendix A. Comparison With Openconfig-platform Data Model

Since more and more devices can be managed by domain controller through OpenConfig, to ensure that our inventory data model can cover these devices' inventory data, we have compared our inventory data model with the "openconfig-platform" model 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	
+-----+-----+-----+			
temperature		performance data	
+-----+-----+-----+			
memory		performance data	
+-----+-----+-----+			
allocated-power		TBD	
+-----+-----+-----+			
used-power		TBD	
+-----+-----+-----+			
pcie		alarm data	
+-----+-----+-----+			
properties		TBD	
+-----+-----+-----+			
subcomponents			
+-----+-----+-----+			
chassis			
+-----+-----+-----+			
port			
+-----+-----+-----+			
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			
+-----+-----+-----+			
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 models

As it mentioned in Section 3.3 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, while some others do not need to be included in this network inventory model like storage and cpu.

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

Appendix B. Terminology of Container

Within this document , with the term "container" we consider an hardware component class capable of containing one or more removable physical entities, e.g. a slot in a chassis is containing a board.

terminology of IVY base model	terminology in other model
container	holder

Table 3: terminology mapping

Appendix C. 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.

Appendix D. Examples of ports, transceivers and port breakouts

This appendix provides some examples of ports, transceivers and port breakouts implementations and how they can be modelled using the "ietf-network-inventory" model defined in Section 5.

Figure 8 shows an example of a single board which contains three type of ports:

1. An integrated port (non pluggable). This port can be of any type (e.g., optical or electrical), single-channel or multi-channel but not supporting breakouts;
2. An empty port;
3. A pluggable port

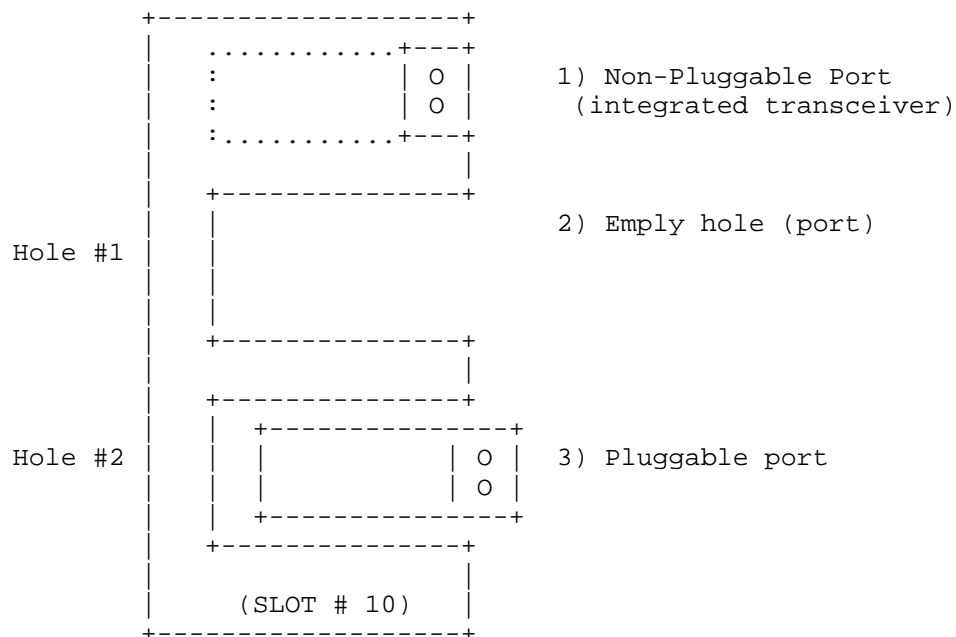


Figure 8: Example of a board with different types of ports

Figure 9 describes an implementation of a single channel optical pluggable trunk port (e.g., a 100G-LR port configured as a single 100GE interface)

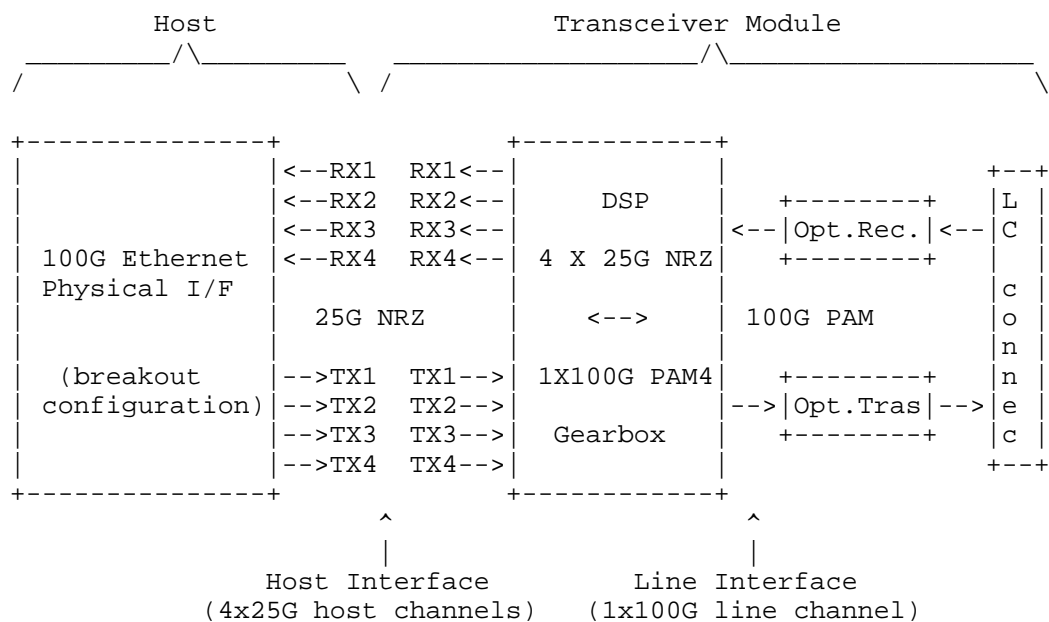


Figure 9: Example of a single channel optical pluggable port

Figure 10 describes an implementation of a Wavelength-Division Multiplexing (WDM) based multi-channel optical pluggable trunk port (e.g., a 400G-LR4 port configured as a single 400GE interface).

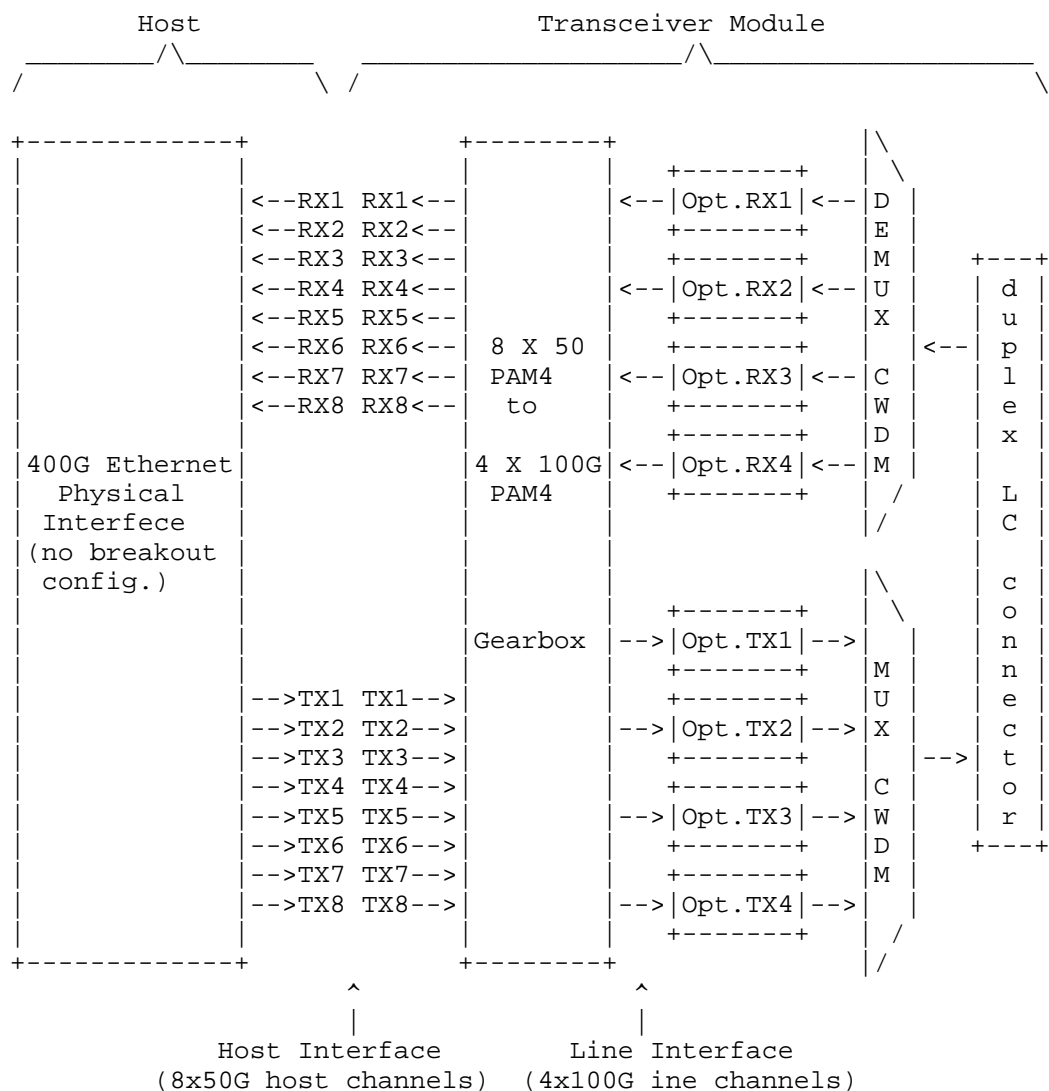


Figure 10: Example of a WDM multi-channel optical pluggable port

In this example, since breakout is not supported, the four WDM channels cannot be modelled as breakout channels and are not relevant from inventory management perspective.

Figure 11 describes an implementation of a Multi-Fiber Push-on (MPO) trunk port (e.g., 400G-DR4 port configured as a single 400GE interface).

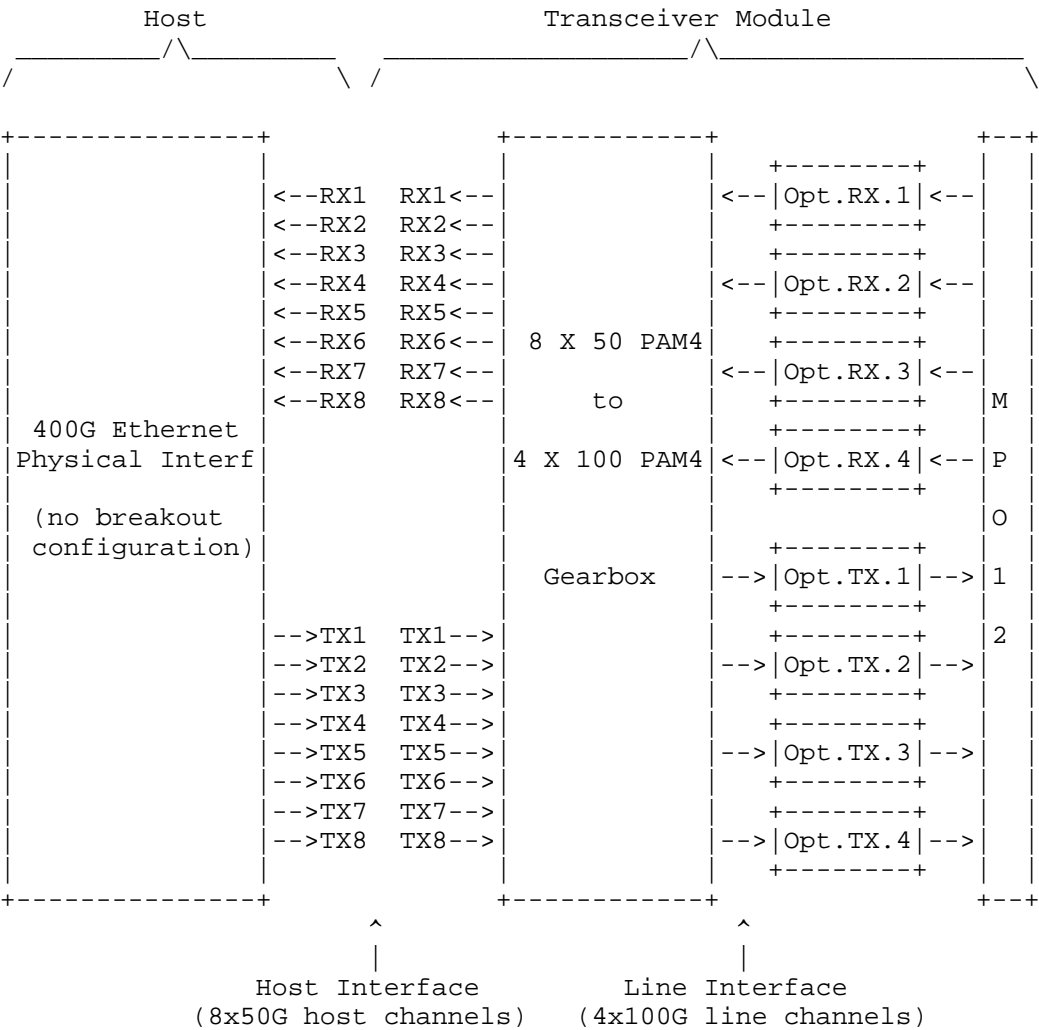


Figure 11: Example of a MPO trunk port

If this MPO port cannot support breakouts, the four line channels cannot be modelled as breakout channels and are not relevant from inventory management perspective. From a network inventory perspective, there is no difference between single-channel ports and MPO trunk ports which do not support port breakouts.

Instead, the MPO port can support breakouts, the four line channels are reported as breakout channels because, as describe in Section 3.3.3, the breakout channels represent the capability of the port to support breakout, independently on how the port is configured (trunk or breakout).

Figure 12 describes an implementation of a MPO breakout port (e.g., 400G-DR4 port configured as 4x100GE interfaces).

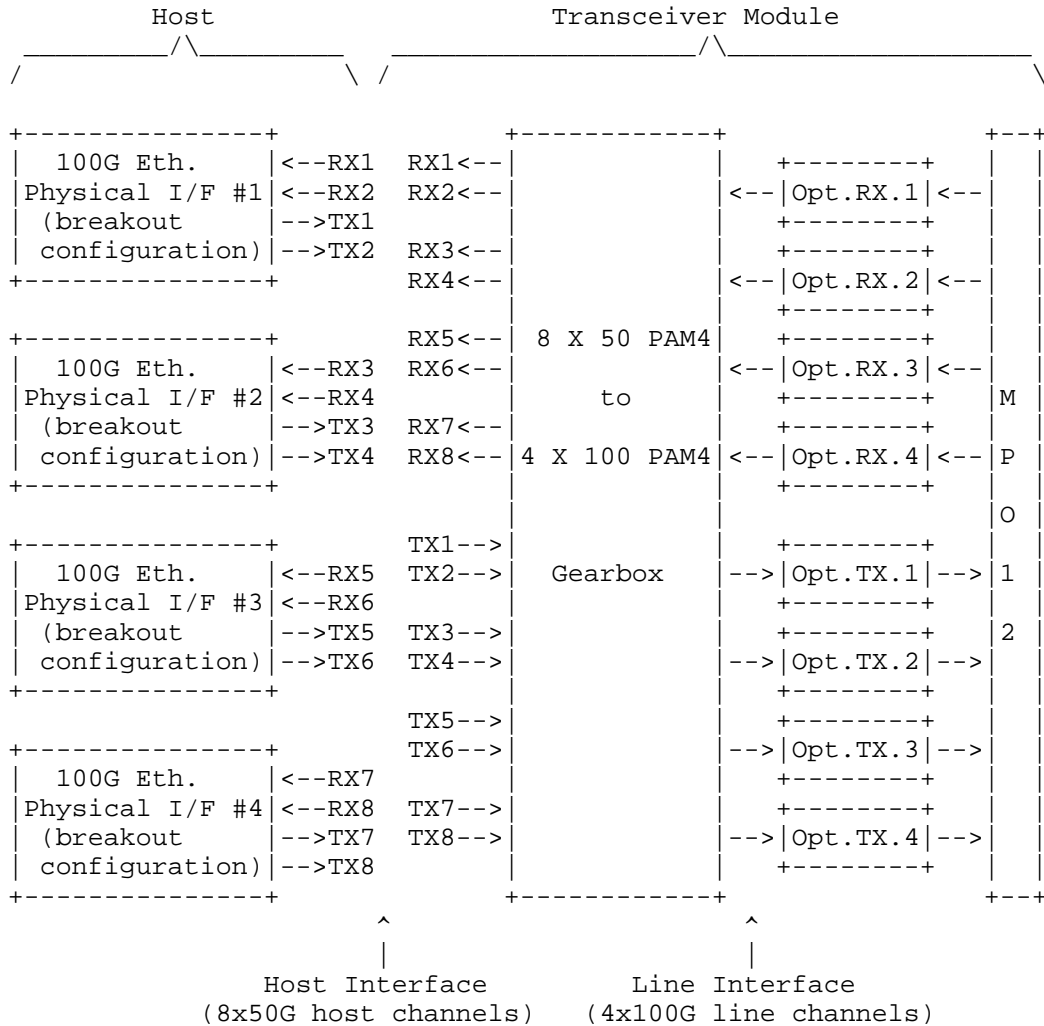


Figure 12: Example of a MPO breakout port

In this example, the four line channels are reported as breakout channels because the port shall support breakout in order to be configured as a breakout port.

D.1. JSON Examples

This appendix contains an example of an instance data tree in JSON encoding [RFC7951], instantiating the "ietf-network-inventory" model to describe a single board, as shown in Figure 8, with seven different types of ports, transceivers and breakouts configurations:

1. An integrated port (non pluggable), as shown in Figure 8;
2. An empty port, as shown in Figure 8;
3. A single channel optical pluggable port, as shown in Figure 8 and Figure 9;
4. A WDM based multi-channel optical pluggable port, as shown in Figure 8 and Figure 10;
5. An MPO trunk port, as shown in Figure 8 and Figure 11, which does not support port breakouts;
6. An MPO trunk port, as shown in Figure 8 and Figure 11, which can not support port breakouts but it has been configured as a trunk port,
7. An MPO breakout port, as shown in Figure 8 and Figure 12.

Note: as described in Section 3.3.3, reporting whether an MPO port is configured as a trunk or as a breakout port, is outside the scope of the base network inventory model.

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-network-inventory:network-inventory": {
    "network-elements": {
      "network-element" : [
        {
          "ne-id": "NE-1",
          "description": "Network element example with ports and \
                           breakouts.",
          "components": {
            "component": [
              {
                "component-id": "board-1",
```

```
"class": "iana-hardware:module",
"description": "Network element example with ports \
                and breakouts."
},
{
  "component-id": "port-1",
  "class": "iana-hardware:port",
  "description": "Example of an integrated (non-\
                pluggable) port.",
  "parent": "board-1",
  "parent-rel-pos": 1
},
{
  "component-id": "transceiver-module-1",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Example of an integrated (non-\
                pluggable) port.",
  "parent": "port-1",
  "is-fru": false
},
{
  "component-id": "port-2",
  "class": "iana-hardware:port",
  "description": "Example of an empty port.",
  "parent": "board-1",
  "parent-rel-pos": 2
},
{
  "component-id": "port-3",
  "class": "iana-hardware:port",
  "description": "Example of a single channel optical \
                pluggable port.",
  "parent": "board-1",
  "parent-rel-pos": 3
},
{
  "component-id": "transceiver-module-3",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Example of a single channel optical \
                pluggable port.",
  "parent": "port-3",
  "is-fru": true
},
{
  "component-id": "port-4",
  "class": "iana-hardware:port",
  "description": "Example of a WDM multi-channel \
                pluggable port.",
```

```
    "parent": "board-1",
    "parent-rel-pos": 4
  },
  {
    "component-id": "transceiver-module-4",
    "class": "ietf-network-inventory:transceiver-module",
    "description": "Example of a WDM multi-channel \
                    pluggable port.",
    "parent": "port-4",
    "is-fru": true
  },
  {
    "component-id": "port-5",
    "class": "iana-hardware:port",
    "description": "Example of an optical MPO pluggable \
                    trunk port (not supporting breakouts).",
    "parent": "board-1",
    "parent-rel-pos": 5
  },
  {
    "component-id": "transceiver-module-5",
    "class": "ietf-network-inventory:transceiver-module",
    "description": "Example of an optical MPO pluggable \
                    trunk port (not supporting breakouts).",
    "parent": "port-5",
    "is-fru": true
  },
  {
    "component-id": "port-6",
    "class": "iana-hardware:port",
    "description": "Example of an optical MPO pluggable \
                    trunk port (supporting breakouts).",
    "parent": "board-1",
    "parent-rel-pos": 6
  },
  {
    "component-id": "transceiver-module-6",
    "class": "ietf-network-inventory:transceiver-module",
    "description": "Example of an optical MPO pluggable \
                    trunk port (supporting breakouts).",
    "parent": "port-6",
    "is-fru": true,
    "transceiver-module-specific-info": {
      "breakout-channels": {
        "breakout-channel": [
          {
            "channel-id": 1
          }
        ]
      }
    }
  },
```

```

        {
            "channel-id": 2
        },
        {
            "channel-id": 3
        },
        {
            "channel-id": 4
        }
    ]
}
},
{
    "component-id": "port-7",
    "class": "iana-hardware:port",
    "description": "Example of an optical MPO pluggable \
                    breakout port.",
    "parent": "board-1",
    "parent-rel-pos": 7
},
{
    "component-id": "transceiver-module-7",
    "class": "ietf-network-inventory:transceiver-module",
    "description": "Example of an optical MPO pluggable \
                    breakout port.",
    "parent": "port-7",
    "is-fru": true,
    "transceiver-module-specific-info": {
        "breakout-channels": {
            "breakout-channel": [
                {
                    "channel-id": 1
                },
                {
                    "channel-id": 2
                },
                {
                    "channel-id": 3
                },
                {
                    "channel-id": 4
                }
            ]
        }
    }
}
]

```

```
    }  
  }  
] }  
}  
}
```

Appendix E. Example of multi-chassis network elements

This appendix provides some examples of multi-chassis network elements and how they can be modelled using the "ietf-network-inventory" model defined in Section 5.

Multi-chassis network elements are network elements composed by two or more chassis interconnected, in principle, with any topology.

Stacked switches are an example of multi-chassis which consist of multiple standalone switches that are interconnected through dedicated stack ports and cables and managed as a single logical unit. Stacked switch: - are connected using a daisy-chain or a ring topology - are managed using a single IP Address - synchronized software-upgrade - use Priority/MAC-Addr(s) decide Master/Members selection and communication.

Figure 13 and Figure 14 describe two examples of stacked switch with three stacked switches (pizza boxes) connected in a daisy-chain or ring topology.

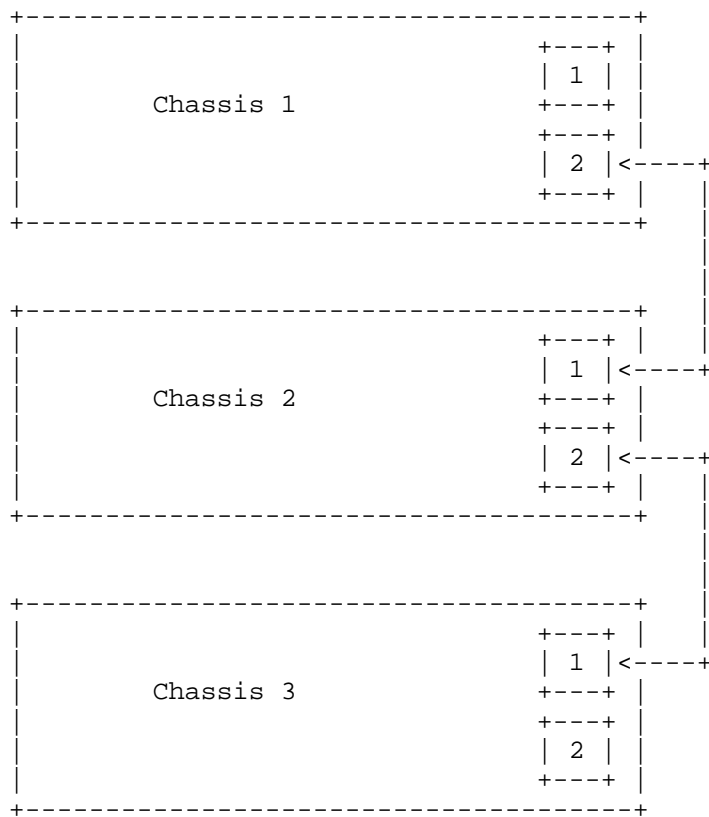


Figure 13: Example of a stacked switch in a daisy chain topology

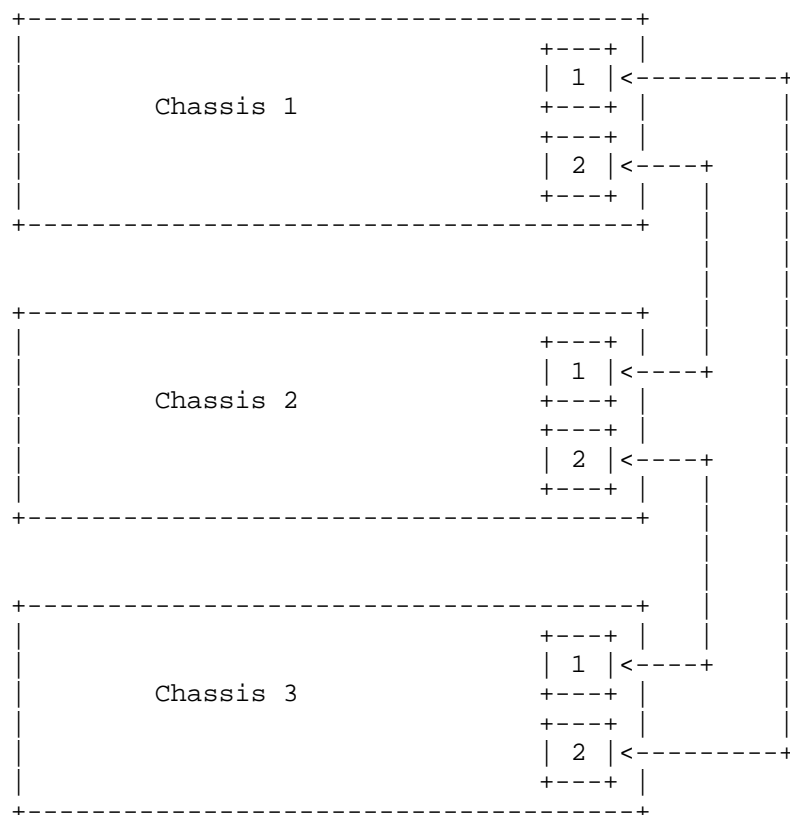


Figure 14: Example of a stacked switch in a ring topology

Using the base network inventory model, each stackable switch can be modelled as a chassis within the same network element, which models the stacked switch. The stack ports are modelled like other ports. The stack cables are not reported using the base network inventory model but can be reported using the passive network inventory model under definition in [I-D.ygb-ivy-passive-network-inventory].

Cascaded switches are another example of multi-chassis which consist of multiple standalone switches that are interconnected and managed as a single logical unit. Cascaded switch: - are usually connected in a tree topology - are managed using a single IP Address - the root of the tree is configured as Master.

Figure 15 describe an example of cascaded switch with three chassis connected in a tree topology.

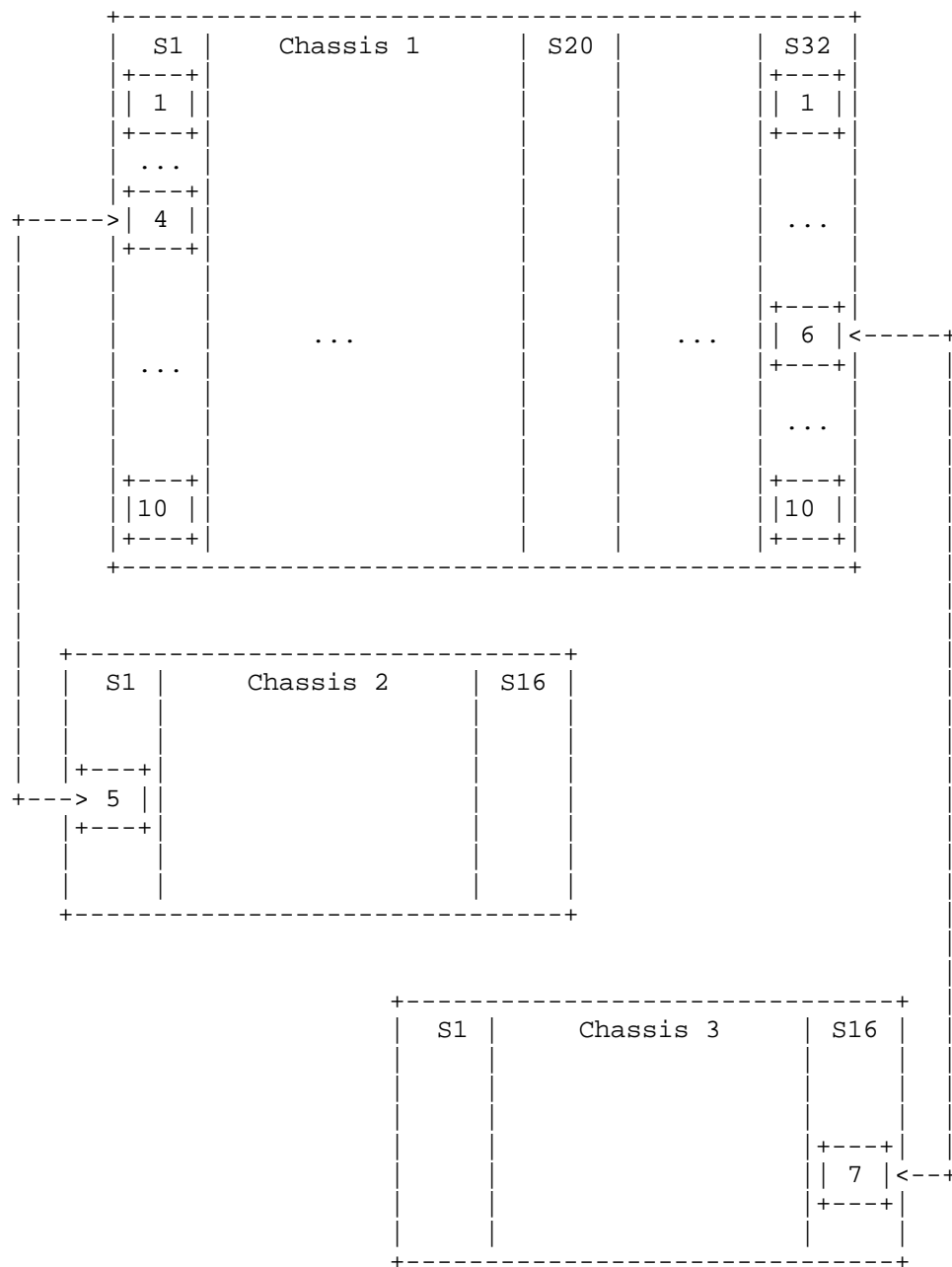


Figure 15: Example of a cascaded switch in a tree topology

Using the base network inventory model each interconnected switch can be modelled as a chassis within the same network element, which models the cascaded switch. The ports used to interconnect the different chassis are normal (traffic) ports and modelled like other ports. The interconnecting cables are not reported using the base network inventory model but can be reported using the passive network inventory model under definition in [I-D.ygb-ivy-passive-network-inventory].

E.1. JSON Examples

This appendix contains an example of an instance data tree in JSON encoding [RFC7951], instantiating the "ietf-network-inventory" model to describe the three examples of multi-chassis NEs, as shown in xxx, yyy and zzz.

Note: the base inventory model allows reporting only the chassis and ports configuration. Reporting the link between the chassis of the same NE is outside the scope of the base inventory model. The YANG data model under definition in [I-D.ygb-ivy-passive-network-inventory] as an augmentation of the base inventory model can be used to provide this additional information.

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
{
  "ietf-network-inventory:network-inventory": {
    "network-elements": {
      "network-element" : [
        {
          "ne-id": "NE-1",
          "description": "Stack Switch in a daisy chain topology.",
          "components": {
            "component": [
              {
                "component-id": "chassis-1",
                "class": "iana-hardware:chassis",
                "description": "First switch of the stack.",
                "parent-rel-pos": 1,
                "is-fru": true
              },
              {
                "component-id": "board-1",
                "class": "iana-hardware:module",
                "description": "Single board of the first switch of \
                               the stack.",
                "parent": "chassis-1",
```

```
    "is-fru": false
  },
  {
    "component-id": "port-1-1",
    "class": "iana-hardware:port",
    "description": "First stack port of the first \
                  switch in the stack.",
    "parent": "board-1",
    "parent-rel-pos": 1
  },
  {
    "component-id": "port-1-2",
    "class": "iana-hardware:port",
    "description": "Second stack port of the first \
                  switch in the stack.",
    "parent": "board-1",
    "parent-rel-pos": 2
  },
  {
    "component-id": "transceiver-module-1-2",
    "class": "ietf-network-inventory:transceiver-module",
    "description": "Transceiver module plugged in the \
second stack port of the first switch in the stack.",
    "parent": "port-1-2",
    "is-fru": true
  },
  {
    "component-id": "chassis-2",
    "class": "iana-hardware:chassis",
    "description": "Second switch of the stack.",
    "parent-rel-pos": 1,
    "is-fru": true
  },
  {
    "component-id": "board-2",
    "class": "iana-hardware:module",
    "description": "Single board of the second switch \
                  of the stack.",
    "parent": "chassis-2",
    "is-fru": false
  },
  {
    "component-id": "port-2-1",
    "class": "iana-hardware:port",
    "description": "First stack port of the second \
                  switch in the stack.",
    "parent": "board-2",
    "parent-rel-pos": 1
  }
```

```
    },
    {
      "component-id": "transceiver-module-2-1",
      "class": "ietf-network-inventory:transceiver-module",
      "description": "Transceiver module plugged in the \
        first stack port of the first switch in the stack.",
      "parent": "port-2-1",
      "is-fru": true
    },
    {
      "component-id": "port-2-2",
      "class": "iana-hardware:port",
      "description": "Second stack port of the second \
        switch in the stack.",
      "parent": "board-2",
      "parent-rel-pos": 2
    },
    {
      "component-id": "transceiver-module-2-2",
      "class": "ietf-network-inventory:transceiver-module",
      "description": "Transceiver module plugged in the \
        second stack port of the first switch in the stack.",
      "parent": "port-2-2",
      "is-fru": true
    },
    {
      "component-id": "chassis-3",
      "class": "iana-hardware:chassis",
      "description": "Third switch of the stack.",
      "parent-rel-pos": 1,
      "is-fru": true
    },
    {
      "component-id": "board-3",
      "class": "iana-hardware:module",
      "description": "Single board of the third switch of \
        the stack.",
      "parent": "chassis-3",
      "is-fru": false
    },
    {
      "component-id": "port-3-1",
      "class": "iana-hardware:port",
      "description": "First stack port of the third \
        switch in the stack.",
      "parent": "board-3",
      "parent-rel-pos": 1
    }
  ],
```

```

    {
      "component-id": "transceiver-module-3-1",
      "class": "ietf-network-inventory:transceiver-module",
      "description": "Transceiver module plugged in the \
        first stack port of the third switch in the stack.",
      "parent": "port-3-1",
      "is-fru": true
    },
    {
      "component-id": "port-3-2",
      "class": "iana-hardware:port",
      "description": "Second stack port of the second \
        switch in the stack.",
      "parent": "board-3",
      "parent-rel-pos": 2
    }
  ]
}
},
{
  "ne-id": "NE-2",
  "description": "Stack Switch in a ring topology.",
  "components": {
    "component": [
      {
        "component-id": "chassis-1",
        "class": "iana-hardware:chassis",
        "description": "First switch of the stack.",
        "parent-rel-pos": 1,
        "is-fru": true
      },
      {
        "component-id": "board-1",
        "class": "iana-hardware:module",
        "description": "Single board of the first switch of \
          the stack.",
        "parent": "chassis-1",
        "is-fru": false
      },
      {
        "component-id": "port-1-1",
        "class": "iana-hardware:port",
        "description": "First stack port of the first \
          switch in the stack.",
        "parent": "board-1",
        "parent-rel-pos": 1
      }
    ]
  }
}

```

```
"component-id": "transceiver-module-1-1",
"class": "ietf-network-inventory:transceiver-module",
"description": "Transceiver module plugged in the \
  first stack port of the first switch in the stack.",
"parent": "port-1-1",
"is-fru": true
},
{
  "component-id": "port-1-2",
  "class": "iana-hardware:port",
  "description": "Second stack port of the first \
    switch in the stack.",
  "parent": "board-1",
  "parent-rel-pos": 2
},
{
  "component-id": "transceiver-module-1-2",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
    second stack port of the first switch in the stack.",
  "parent": "port-1-2",
  "is-fru": true
},
{
  "component-id": "chassis-2",
  "class": "iana-hardware:chassis",
  "description": "Second switch of the stack.",
  "parent-rel-pos": 1,
  "is-fru": true
},
{
  "component-id": "board-2",
  "class": "iana-hardware:module",
  "description": "Single board of the second switch \
    of the stack.",
  "parent": "chassis-2",
  "is-fru": false
},
{
  "component-id": "port-2-1",
  "class": "iana-hardware:port",
  "description": "First stack port of the second \
    switch in the stack.",
  "parent": "board-2",
  "parent-rel-pos": 1
},
{
  "component-id": "transceiver-module-2-1",
```

```
"class": "ietf-network-inventory:transceiver-module",
"description": "Transceiver module plugged in the \
  first stack port of the first switch in the stack.",
"parent": "port-2-1",
"is-fru": true
},
{
  "component-id": "port-2-2",
  "class": "iana-hardware:port",
  "description": "Second stack port of the second \
    switch in the stack.",
  "parent": "board-2",
  "parent-rel-pos": 2
},
{
  "component-id": "transceiver-module-2-2",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
    second stack port of the first switch in the stack.",
  "parent": "port-2-2",
  "is-fru": true
},
{
  "component-id": "chassis-3",
  "class": "iana-hardware:chassis",
  "description": "Third switch of the stack.",
  "parent-rel-pos": 1,
  "is-fru": true
},
{
  "component-id": "board-3",
  "class": "iana-hardware:module",
  "description": "Single board of the third switch of \
    the stack.",
  "parent": "chassis-3",
  "is-fru": false
},
{
  "component-id": "port-3-1",
  "class": "iana-hardware:port",
  "description": "First stack port of the third \
    switch in the stack.",
  "parent": "board-3",
  "parent-rel-pos": 1
},
{
  "component-id": "transceiver-module-3-1",
  "class": "ietf-network-inventory:transceiver-module",
```

```

        "description": "Transceiver module plugged in the \
        first stack port of the third switch in the stack.",
        "parent": "port-3-1",
        "is-fru": true
    },
    {
        "component-id": "port-3-2",
        "class": "iana-hardware:port",
        "description": "Second stack port of the second \
                        switch in the stack.",
        "parent": "board-3",
        "parent-rel-pos": 2
    },
    {
        "component-id": "transceiver-module-3-2",
        "class": "ietf-network-inventory:transceiver-module",
        "description": "Transceiver module plugged in the \
        second stack port of the third switch in the stack.",
        "parent": "port-3-2",
        "is-fru": true
    }
]
},
{
    "ne-id": "NE-3",
    "description": "TBA.",
    "components": {
        "component": [
            {
                "component-id": "chassis-1",
                "class": "iana-hardware:chassis",
                "description": "First TBD.",
                "parent-rel-pos": 1,
                "is-fru": true
            },
            {
                "component-id": "board-1",
                "class": "iana-hardware:module",
                "description": "Single board of the first switch of \
                                the stack.",
                "parent": "chassis-1",
                "is-fru": false
            },
            {
                "component-id": "port-1-1",
                "class": "iana-hardware:port",
                "description": "First stack port of the first \

```

```
switch in the stack.",
  "parent": "board-1",
  "parent-rel-pos": 1
},
{
  "component-id": "transceiver-module-1-1",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
    first stack port of the first switch in the stack.",
  "parent": "port-1-1",
  "is-fru": true
},
{
  "component-id": "port-1-2",
  "class": "iana-hardware:port",
  "description": "Second stack port of the first \
    switch in the stack.",
  "parent": "board-1",
  "parent-rel-pos": 2
},
{
  "component-id": "transceiver-module-1-2",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
    second stack port of the first switch in the stack.",
  "parent": "port-1-2",
  "is-fru": true
},
{
  "component-id": "chassis-2",
  "class": "iana-hardware:chassis",
  "description": "Second switch of the stack.",
  "parent-rel-pos": 1,
  "is-fru": true
},
{
  "component-id": "board-2",
  "class": "iana-hardware:module",
  "description": "Single board of the second switch \
    of the stack.",
  "parent": "chassis-2",
  "is-fru": false
},
{
  "component-id": "port-2-1",
  "class": "iana-hardware:port",
  "description": "First stack port of the second \
    switch in the stack.",
```



```
"parent": "board-2",
"parent-rel-pos": 1
},
{
  "component-id": "transceiver-module-2-1",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
first stack port of the first switch in the stack.",
  "parent": "port-2-1",
  "is-fru": true
},
{
  "component-id": "port-2-2",
  "class": "iana-hardware:port",
  "description": "Second stack port of the second \
switch in the stack.",
  "parent": "board-2",
  "parent-rel-pos": 2
},
{
  "component-id": "transceiver-module-2-2",
  "class": "ietf-network-inventory:transceiver-module",
  "description": "Transceiver module plugged in the \
second stack port of the first switch in the stack.",
  "parent": "port-2-2",
  "is-fru": true
},
{
  "component-id": "chassis-3",
  "class": "iana-hardware:chassis",
  "description": "Third switch of the stack.",
  "parent-rel-pos": 1,
  "is-fru": true
},
{
  "component-id": "board-3",
  "class": "iana-hardware:module",
  "description": "Single board of the third switch of \
the stack.",
  "parent": "chassis-3",
  "is-fru": false
},
{
  "component-id": "port-3-1",
  "class": "iana-hardware:port",
  "description": "First stack port of the third \
switch in the stack.",
  "parent": "board-3",
```

```

        "parent-rel-pos": 1
      },
      {
        "component-id": "transceiver-module-3-1",
        "class": "ietf-network-inventory:transceiver-module",
        "description": "Transceiver module plugged in the \
          first stack port of the third switch in the stack.",
        "parent": "port-3-1",
        "is-fru": true
      },
      {
        "component-id": "port-3-2",
        "class": "iana-hardware:port",
        "description": "Second stack port of the second \
          switch in the stack.",
        "parent": "board-3",
        "parent-rel-pos": 2
      },
      {
        "component-id": "transceiver-module-3-2",
        "class": "ietf-network-inventory:transceiver-module",
        "description": "Transceiver module plugged in the \
          second stack port of the third switch in the stack.",
        "parent": "port-3-2",
        "is-fru": true
      }
    ]
  }
}

```

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.

Contributors

Italo Busi
 Huawei Technologies
 Email: italo.busi@huawei.com

Aihua Guo
Futurewei Technologies
Email: aihuaguo.ietf@gmail.com

Victor Lopez
Nokia
Email: victor.lopez@nokia.com

Bo Wu
Huawei Technologies
Email: lana.wubo@huawei.com

Chenfang Zhang
China Unicom
Email: zhangcf80@chinaunicom.cn

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Nigel Davis
Ciena
Email: ndavis@ciena.com

Authors' Addresses

Chaode Yu
Huawei Technologies
Email: yuchaode@huawei.com

Sergio Belotti
Nokia
Email: sergio.belotti@nokia.com

Jean-Francois Bouquier
Vodafone
Email: jeff.bouquier@vodafone.com

Fabio Peruzzini
FiberCop
Email: fabio.peruzzini@fibercop.com

Phil Bedard
Cisco
Email: phbedard@cisco.com