Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 14, 2016

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
J. Dong
Huawei Technologies
D. Romascanu
Avaya

May 13, 2016

A YANG Data Model for Entity Management
draft-ietf-netmod-entity-00

Abstract

This document defines a YANG data model for the management of multiple physical entities managed by a single server.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 14, 2016.

Table of Contents

1. Introduction 2
1.1. Terminology 2
1.1.1. Tree Diagrams 2
2. Objectives 3
3. Entity Data Model 3
3.1. The Physical Entry Lists 5
4. Relationship to ENTITY-MIB 5
5. Relationship to ENTITY-SENSOR-MIB 6
6. Relationship to ENTITY-STATE-MIB 6
7. Entity YANG Module 6
8. IANA Considerations 34
9. Security Considerations 35
10. Acknowledgements 35
11. Normative References 35
Authors' Addresses 35

1. Introduction

This document defines a YANG [I-D.ietf-netmod-rfc6020bis] data model for the management of multiple physical entities managed by a single server.

The data model includes configuration data and state data (status information and counters for the collection of statistics).

1.1. Terminology

The keywords "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].

1.1.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.

- Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

This section describes some of the design objectives for the entity model.

- There are many common properties used to identify the entities, which need to be supported in the entity data module.

- There are many important information and states about the entities, which needs to be collected from the devices which support the entity data model.

- The entity data model SHOULD be suitable for new implementations to use as is.

- The entity data model defined in this document can be implemented on a system that also implements ENTITY-MIB, thus the mapping between the entity data model and ENTITY-MIB SHOULD be clear.

3. Entity Data Model

This document defines the YANG module "ietf-entity", which has the following structure:

```
  module: ietf-entity
     +--ro entity-state
     |  +--ro last-change?       yang:date-and-time
     |  +--ro physical-entity* [name]
     |     +--ro name                  string
     |     +--ro class?                identityref
     |     +--ro physical-index?       int32 {entity-mib}?
     |     +--ro description?          string
     |     +--ro contained-in*         -> ../../physical-entity/name
     |     +--ro contains-child*       -> ../../physical-entity/name
     |     +--ro parent-rel-pos?       int32
     |     +--ro hardware-rev?         string
     |     +--ro firmware-rev?         string
     |     +--ro software-rev?         string
     |     +--ro serial-num?           string
```

```
        |     +--ro mfg-name?             string
        |     +--ro model-name?           string
        |     +--ro alias?                string
        |     +--ro asset-id?             string
        |     +--ro is-fru?               boolean
        |     +--ro mfg-date?             yang:date-and-time
        |     +--ro uri*                  inet:uri
        |     +--ro uuid?                 yang:uuid
        |     +--ro state {entity-state}?
        |     |  +--ro state-last-changed?     yang:date-and-time
        |     |  +--ro admin-state?            entity-admin-state
        |     |  +--ro oper-state?             entity-oper-state
        |     |  +--ro usage-state?            entity-usage-state
        |     |  +--ro alarm-status?           entity-alarm-status
        |     |  +--ro standby-status?         entity-standby-status
        |     +--ro sensor-data {entity-sensor}?
        |        +--ro data-type?              entity-sensor-data-type
        |        +--ro data-scale?             entity-sensor-data-scale
        |        +--ro precision?              entity-sensor-precision
        |        +--ro value?                  entity-sensor-value
        |        +--ro oper-status?            entity-sensor-status
        |        +--ro sensor-units-display?   string
        |        +--ro value-timestamp?        yang:date-and-time
        |        +--ro value-update-rate?      uint32
        +--rw entity {entity-config}?
           +--rw physical-entity* [name]
              +--rw name           string
              +--rw serial-num?    string
              +--rw alias?         string
              +--rw asset-id?      string
              +--rw uri*           inet:uri
              +--rw admin-state?   entity-admin-state {entity-state}?

     notifications:
        +---n ent-config-change
        +---n ent-state-oper-enabled {entity-state}?
        |  +--ro name?           -> /entity-state/physical-entity
        |                           /name
        |  +--ro admin-state?    -> /entity-state/physical-entity
        |                           /state/admin-state
        |  +--ro alarm-status?   -> /entity-state/physical-entity
        |                           /state/alarm-status
        +---n ent-state-oper-disabled {entity-state}?
           +--ro name?           -> /entity-state/physical-entity
                                    /name
           +--ro admin-state?    -> /entity-state/physical-entity
                                    /state/admin-state
           +--ro alarm-status?   -> /entity-state/physical-entity
```

/state/alarm-status

3.1. The Physical Entry Lists

The data model for physical entities presented in this document uses a flat list of entities. Each entity in the list is identified by its name. Furthermore, each entity has a mandatory "class" leaf.

The "iana-entity" module defines YANG identities for the hardware types in the IANA-maintained "IANA-ENTITY-MIB" registry.

The "class" leaf is a YANG identity that describes the type of the hardware. Vendors are encouraged to either directly use one of the common IANA-defined identities, or derive a more specific identity from one of them.

There is one optional list of configured physical entities ("/entity/physical-entity"), and a separate list for the operational state of all physical entities ("/entity-state/physical-entity").

4. Relationship to ENTITY-MIB

If the device implements the ENTITY-MIB [RFC6933], each entry in the /entity-state/physical-entity list is mapped to one EntPhysicalEntry. Objects that are writable in the MIB are mapped to nodes in the /entity/physical-entity list.

The "physical-index" leaf MUST contain the value of the corresponding entPhysicalEntry's entPhysicalIndex.

The "class" leaf is mapped to both entPhysicalClass and entPhysicalVendorType. If the value of the "class" leaf is an identity that is either derived from or is one of the identities in the "iana-entity" module, then entPhysicalClass contains the corresponding IANAPhysicalClass enumeration value. Otherwise, entPhysicalClass contains the IANAPhysicalClass value "other(1)". Vendors are encouraged to define an identity (derived from an identity in "iana-entity" if possible) for each enterprise-specific registration identifier used for entPhysicalVendorType, and use that identity for the "class" leaf.

The following tables list the YANG data nodes with corresponding objects in the ENTITY-MIB.

-- YANG data nodes and related ENTITY-MIB objects

YANG data node in /entity-state /physical-entity	ENTITY-MIB object
name	entPhysicalName
class	entPhysicalClass
	entPhysicalVendorType
physical-index	entPhysicalIndex
description	entPhysicalDescr
contained-in	entPhysicalContainedIn
contains-child	entPhysicalChildIndex
parent-rel-pos	entPhysicalParentRelPos
hardware-rev	entPhysicalHardwareRev
firmware-rev	entPhysicalFirmwareRev
software-rev	entPhysicalSoftwareRev
serial-num	entPhysicalSerialNum
mfg-name	entPhysicalMfgName
model-name	entPhysicalModelName
alias	entPhysicalAlias
asset-id	entPhysicalAssetID
is-fru	entPhysicalIsFRU
mfg-date	entPhysicalMfgDate
uri	entPhysicalUris
uuid	entPhysicalUUID

5. Relationship to ENTITY-SENSOR-MIB

TBD relationship to [RFC3433].

6. Relationship to ENTITY-STATE-MIB

TBD relationship to [RFC4268].

7. Entity YANG Module

<CODE BEGINS> file "ietf-entity@2016-05-13.yang"

```
module ietf-entity {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-entity";
  prefix ent;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
```

```
    }

    organization
      "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

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

       WG Chair: Lou Berger
                 <mailto:lberger@labn.net>

       WG Chair: Juergen Schoenwaelder
                 <mailto:j.schoenwaelder@jacobs-university.de>

       WG Chair: Kent Watsen
                 <mailto:kwatsen@juniper.net>

       Editor:   Andy Bierman
                 <mailto:andy@yumaworks.com>

       Editor:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Editor:   Jie Dong
                 <mailto:jie.dong@huawei.com>

       Editor:   Dan Romascanu
                 <mailto:dromasca@avaya.com>";

    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.

    description
      "This module contains a collection of YANG definitions for
       managing physical entities.

       Copyright (c) 2016 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
```

```
       the RFC itself for full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2016-05-13 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Entity Management";
    }

    /*
     * Features
     */

    feature entity-mib {
      description
        "This feature indicates that the device implements
         the ENTITY-MIB.";
      reference "RFC 6933: Entity MIB (Version 4)";
    }

    feature entity-config {
      description
        "Indicates that the server supports configuration of
         physical entities.";
    }

    feature entity-state {
      description
        "Indicates the ENTITY-STATE-MIB objects are supported";
      reference "RFC 4268: Entity State MIB";
    }

    feature entity-sensor {
      description
        "Indicates the ENTITY-SENSOR-MIB objects are supported";
      reference "RFC 3433: Entity Sensor MIB";
    }

    /*
     * Typedefs
     */

    typedef entity-admin-state {
      type enumeration {
        enum unknown {
          value 1;
```

```
          description
            "The resource is unable to report administrative state.";
        }
        enum locked {
          value 2;
          description
            "The resource is administratively prohibited from use.";
        }
        enum shutting-down {
          value 3;
          description
            "The resource usage is administratively limited to current
             instances of use.";
        }
        enum unlocked {
          value 4;
          description
            "The resource is not administratively prohibited from use.";
        }
      }
      description
        "Represents the various possible administrative states.";
      reference "RFC 4268: EntityAdminState";
    }

    typedef entity-oper-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report operational state.";
        }
        enum disabled {
          value 2;
          description
            "The resource is totally inoperable.";
        }
        enum enabled {
          value 3;
          description
            "The resource is partially or fully operable.";
        }
        enum testing {
          value 4;
          description
            "The resource is currently being tested and cannot
             therefore report whether it is operational or not.";
        }
```

```
      }
      description
        "Represents the possible values of operational states.";
      reference "RFC 4268: EntityOperState";
    }

    typedef entity-usage-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report usage state.";
        }
        enum idle {
          value 2;
          description
            "The resource is servicing no users.";
        }
        enum active {
          value 3;
          description
            "The resource is currently in use and it has sufficient
             spare capacity to provide for additional users.";
        }
        enum busy {
          value 4;
          description
            "The resource is currently in use, but it currently has
             no spare capacity to provide for additional users.";
        }
      }
      description
        "Represents the possible values of usage states.";
      reference "RFC 4268, EntityUsageState";
    }

    typedef entity-alarm-status {
      type bits {
        bit unknown {
          position 0;
          description
            "The resource is unable to report alarm state.";
        }
        bit under-repair {
          position 1;
          description
            "The resource is currently being repaired, which, depending
             on the implementation, may make the other values in this
```

```
            bit string not meaningful.";
        }
        bit critical {
          position 2;
          description
            "One or more critical alarms are active against the
             resource.";
        }
        bit major {
          position 3;
          description
            "One or more major alarms are active against the resource.";
        }
        bit minor {
          position 4;
          description
            "One or more minor alarms are active against the resource.";
        }
        bit warning {
          position 5;
          description
            "One or more warning alarms are active against the resource.
             This alarm status is not defined in X.733.";
        }
        bit indeterminate {
          position 6;
          description
            "One or more alarms of whose perceived severity cannot be
             determined are active against this resource.
             This alarm status is not defined in X.733.";
        }
      }
      description
        "Represents the possible values of alarm status.
         An Alarm [RFC3877] is a persistent indication of an error or
         warning condition.

         When no bits of this attribute are set, then no active
         alarms are known against this entity and it is not under
         repair.";
      reference "RFC 4268: EntityAlarmStatus";
    }

    typedef entity-standby-status {
      type enumeration {
        enum unknown {
          value 1;
          description
```

```
            "The resource is unable to report standby state.";
        }
        enum hot-standby {
          value 2;
          description
            "The resource is not providing service, but it will be
             immediately able to take over the role of the resource
             to be backed up, without the need for initialization
             activity, and will contain the same information as the
             resource to be backed up.";
        }
        enum cold-standby {
          value 3;
          description
            "The resource is to back up another resource, but will not
             be immediately able to take over the role of a resource
             to be backed up, and will require some initialization
             activity.";
        }
        enum providing-service {
          value 4;
          description
            "The resource is providing service.";
        }
      }
      description
        "Represents the possible values of standby status.";
      reference "RFC 4268: EntityStandbyStatus";
    }

    typedef entity-sensor-data-type {
      type enumeration {
        enum other {
          value 1;
          description
            "A measure other than those listed below.";
        }
        enum unknown {
          value 2;
          description
            "An unknown measurement, or arbitrary, relative numbers";
        }
        enum volts-AC {
          value 3;
          description
            "A measure of electric potential (alternating current).";
        }
        enum volts-DC {
```

```
          value 4;
          description
            "A measure of electric potential (direct current).";
        }
        enum amperes {
          value 5;
          description
            "A measure of electric current.";
        }
        enum watts {
          value 6;
          description
            "A measure of power.";
        }
        enum hertz {
          value 7;
          description
            "A measure of frequency.";
        }
        enum celsius {
          value 8;
          description
            "A measure of temperature.";
        }
        enum percent-RH {
          value 9;
          description
            "A measure of percent relative humidity.";
        }
        enum rpm {
          value 10;
          description
            "A measure of shaft revolutions per minute.";
        }
        enum cmm {
          value 11;
          description
            "A measure of cubic meters per minute (airflow).";
        }
        enum truth-value {
          value 12;
          description
            "Value is one of 1 (true) or 2 (false)";
        }
      }
      description
        "An node using this data type represents the Entity Sensor
         measurement data type associated with a physical sensor
```

```
         value. The actual data units are determined by examining an
         node of this type together with the associated
         entity-sensor-data-scale node.

         An node of this type SHOULD be defined together with nodes
         of type entity-sensor-data-scale and
         entity-sensor-precision. These three types are used to
         identify the semantics of an node of type
         entity-sensor-value.";
       reference "RFC 3433: EntitySensorDataType";
     }

     typedef entity-sensor-data-scale {
       type enumeration {
         enum yocto {
           value 1;
           description
             "Data scaling factor of 10^-24.";
         }
         enum zepto {
           value 2;
           description
             "Data scaling factor of 10^-21.";
         }
         enum atto {
           value 3;
           description
             "Data scaling factor of 10^-18.";
         }
         enum femto {
           value 4;
           description
             "Data scaling factor of 10^-15.";
         }
         enum pico {
           value 5;
           description
             "Data scaling factor of 10^-12.";
         }
         enum nano {
           value 6;
           description
             "Data scaling factor of 10^-9.";
         }
         enum micro {
           value 7;
           description
             "Data scaling factor of 10^-6.";
```

```
        }
        enum milli {
          value 8;
          description
            "Data scaling factor of 10^-3.";
        }
        enum units {
          value 9;
          description
            "Data scaling factor of 10^0.";
        }
        enum kilo {
          value 10;
          description
            "Data scaling factor of 10^3.";
        }
        enum mega {
          value 11;
          description
            "Data scaling factor of 10^6.";
        }
        enum giga {
          value 12;
          description
            "Data scaling factor of 10^9.";
        }
        enum tera {
          value 13;
          description
            "Data scaling factor of 10^12.";
        }
        enum exa {
          value 14;
          description
            "Data scaling factor of 10^15.";
        }
        enum peta {
          value 15;
          description
            "Data scaling factor of 10^18.";
        }
        enum zetta {
          value 16;
          description
            "Data scaling factor of 10^21.";
        }
        enum yotta {
          value 17;
```

```
          description
            "Data scaling factor of 10^24.";
        }
      }
      description
        "An node using this data type represents a data scaling
         factor, represented with an International System of Units (SI)
         prefix.  The actual data units are determined by examining an
         node of this type together with the associated
         entity-sensor-data-type.

         An node of this type SHOULD be defined together with nodes
         of type entity-sensor-data-type and entity-sensor-precision.
         Together, associated nodes of these three types are used to
         identify the semantics of an node of type
         entity-sensor-value.";
      reference "RFC 3433: EntitySensorDataScale";
    }

    typedef entity-sensor-precision {
      type int32 {
        range "-8 .. 9";
      }
      description
        "An node using this data type represents a sensor
         precision range.

         An node of this type SHOULD be defined together with nodes
         of type entity-sensor-data-type and entity-sensor-data-scale.
         Together, associated nodes of these three types are used to
         identify the semantics of an node of type
         entity-sensor-value.

         If an node of this type contains a value in the range 1 to 9,
         it represents the number of decimal places in the fractional
         part of an associated entity-sensor-value fixed- point number.

         If an node of this type contains a value in the range -8 to
         -1, it represents the number of accurate digits in the
         associated entity-sensor-value fixed-point number.

         The value zero indicates the associated entity-sensor-value
         node is not a fixed-point number.

         Server implementors must choose a value for the associated
         entity-sensor-precision node so that the precision and
         accuracy of the associated entity-sensor-value node is
         correctly indicated.
```

```
         For example, a physical entity representing a temperature
         sensor that can measure 0 degrees to 100 degrees C in 0.1
         degree increments, +/- 0.05 degrees, would have an
         entity-sensor-precision value of '1', an
         entity-sensor-data-scale value of 'units', and an
         entity-sensor-value ranging from '0' to '1000'.  The
         entity-sensor-value would be interpreted as 'degrees C * 10'.";
       reference "RFC 3433: EntitySensorPrecision";
     }

     typedef entity-sensor-value {
       type int32 {
         range "-1000000000 .. 1000000000";
       }
       description
        "An node using this data type represents an Entity Sensor
         value.

         An node of this type SHOULD be defined together with nodes
         of type entity-sensor-data-type, entity-sensor-data-scale, and
         entity-sensor-precision.  Together, associated nodes of those
         three types are used to identify the semantics of an node of
         this data type.

         The semantics of an node using this data type are determined
         by the value of the associated entity-sensor-data-type node.

         If the associated entity-sensor-data-type node is equal to
         'voltsAC', 'voltsDC', 'amperes', 'watts', 'hertz',
         'celsius', or 'cmm', then an node of this type MUST
         contain a fixed point number ranging from -999,999,999 to
         +999,999,999.  The value -1000000000 indicates an underflow
         error. The value +1000000000 indicates an overflow error.  The
         entity-sensor-precision indicates how many fractional digits
         are represented in the associated entity-sensor-value node.

         If the associated entity-sensor-data-type node is equal to
         'percentRH', then an node of this type MUST contain a
         number ranging from 0 to 100.

         If the associated entity-sensor-data-type node is equal to
         'rpm', then an node of this type MUST contain a number
         ranging from -999,999,999 to +999,999,999.

         If the associated entity-sensor-data-type node is equal to
         'truthvalue', then an node of this type MUST contain
         either the value 1 (true) or the value 2 (false)'.
```

```
        If the associated entity-sensor-data-type node is equal to
        'other' or unknown', then an node of this type MUST
        contain a number ranging from -1000000000 to 1000000000.";
      reference "RFC 3433: EntitySensorValue";
    }

    typedef entity-sensor-status {
      type enumeration {
        enum ok {
          value 1;
          description
            "Indicates that the server can obtain the sensor value.";
        }
        enum unavailable {
          value 2;
          description
            "Indicates that the server presently cannot obtain the
             sensor value.";
        }
        enum nonoperational {
          value 3;
          description
            "Indicates that the server believes the sensor is broken.
             The sensor could have a hard failure (disconnected wire),
             or a soft failure such as out-of-range, jittery, or wildly
             fluctuating readings.";
        }
      }
      description
        "An node using this data type represents the operational
         status of a physical sensor.";
      reference "RFC 3433: EntitySensorStatus";
    }

    /*
     * Identities
     */

    identity entity-physical-class {
      description
        "This identity is the base for all physical entity class
         identifiers.";
    }


    /*
     * Operational state data nodes
     */
```

```
    container entity-state {
      config false;
      description
        "Data nodes for the operational state of physical entities.";

      leaf last-change {
        type yang:date-and-time;
        description
          "The time the '/entity-state/physical-entity' list changed.";
      }

      list physical-entity {
        key name;
        description
          "List of physical entities";
        reference "RFC 6933: entPhysicalEntry";

        leaf name {
          type string;
          description
            "Administrative name assigned to this physical entity.
             No restrictions apply. Not required to be the same as
             entPhysicalName.";
        }

        leaf class {
          type identityref {
            base entity-physical-class;
          }
          mandatory true;
          description
            "An indication of the general hardware type
             of the physical entity.";
          reference "RFC 6933: entPhysicalClass";
        }

        leaf physical-index {
          if-feature entity-mib;
          type int32 {
            range "1..2147483647";
          }
          description
            "The entPhysicalIndex for the entPhysicalEntry represented
             by this list entry.";
          reference "RFC 6933: entPhysicalIndex";
        }

        leaf description {
```

```
          type string;
          description
            "A textual description of physical entity.  This node
             should contain a string that identifies the manufacturer's
             name for the physical entity and should be set to a
             distinct value for each version or model of the physical
             entity.";
          reference "RFC 6933: entPhysicalDescr";
        }

        leaf-list contained-in {
          type leafref {
            path "../../physical-entity/name";
          }
          description
            "The name of the physical entity that 'contains'
             this physical entity.";
          reference "RFC 6933: entPhysicalContainedIn";
        }

        leaf-list contains-child {
          type leafref {
            path "../../physical-entity/name";
          }
          description
            "The name of the contained physical entity.";
          reference "RFC 6933: entPhysicalChildIndex";
        }

        leaf parent-rel-pos {
          type int32 {
            range "0 .. 2147483647";
          }
          description
            "An indication of the relative position of this child
             component among all its sibling components.  Sibling
             components are defined as physical entities that share the
             same instance values of each of the contained-in
             and class elements.";
          reference "RFC 6933: entPhysicalParentRelPos";
        }

        leaf hardware-rev {
          type string;
          description
            "The vendor-specific hardware revision string for the
             physical entity.  The preferred value is the hardware
             revision identifier actually printed on the component
```

```
           itself (if present).";
        reference "RFC 6933: entPhysicalHardwareRev";
      }

      leaf firmware-rev {
        type string;
        description
          "The vendor-specific firmware revision string for the
           physical entity.";
        reference "RFC 6933: entPhysicalFirmwareRev";
      }

      leaf software-rev {
        type string;
        description
          "The vendor-specific software revision string for the
           physical entity.";
        reference "RFC 6933: entPhysicalSoftwareRev";
      }

      leaf serial-num {
        type string;
        description
          "The vendor-specific serial number string for the physical
           entity.  The preferred value is the serial number string
           actually printed on the component itself (if present).

           If a serial number has been configured for this entity in
           /entity/physical-entity/serial-num, this node contains the
           configured value.";
        reference "RFC 6933: 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 the serial-num nodes are
           only meaningful amongst physical entities 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 will contain a zero-length string.";
```

```
        reference "RFC 6933: entPhysicalMfgName";
      }

      leaf model-name {
        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 will
           contain a zero-length string.";
        reference "RFC 6933: entPhysicalModelName";
      }

      leaf alias {
        type string {
          length "0 .. 32";
        }
        description
          "An 'alias' name for the physical entity, as specified by
           a network manager, and provides a non-volatile 'handle'
           for the physical entity.

           If an alias has been configured for this entity in
           /entity/physical-entity/alias, this node contains the
           configured value.  If no such alias has been configured,
           the server may set the value of this node to a locally
           unique value.";
        reference "RFC 6933: entPhysicalAlias";
      }

      leaf asset-id {
        type string {
          length "0 .. 32";
        }
        description
          "This node is a user-assigned asset tracking identifier
           (as specified by a network manager) for the physical entity
           and provides non-volatile storage of this information.

           If an asset tracking identifier has been configured for
           this entity in /entity/physical-entity/addet-id, this node
           contains the configured value.";
        reference "RFC 6933: entPhysicalAssetID";
      }
```

```
        leaf is-fru {
          type boolean;
          description
            "This node indicates whether or not this physical entity
             is considered a 'field replaceable unit' by the vendor.  If
             this node contains the value 'true', then this
             physical entity identifies a field replaceable unit.  For
             all physical entities that represent components permanently
             contained within a field replaceable unit, the value
             'false' should be returned for this node.";
          reference "RFC 6933: entPhysicalIsFRU";
        }

        leaf mfg-date {
          type yang:date-and-time;
          description
            "The date of manufacturing of the managed entity.";
          reference "RFC 6933: entPhysicalMfgDate";
        }

        leaf-list uri {
          type inet:uri;
          description
            "This node contains identification information about the
             physical entity.

             If uris have been configured for this entity in
             /entity/physical-entity/uri, this node contains the
             configured values.";
          reference "RFC 6933: entPhysicalUris";
        }

        leaf uuid {
          type yang:uuid;
          description
            "A Universally Unique Identifier of the physical entity.";
          reference "RFC 6933: entPhysicalUUID";
        }

        container state {
          if-feature entity-state;
          description
            "State-related nodes";
          reference "RFC 4268: Entity State MIB";

          leaf state-last-changed {
            type yang:date-and-time;
            description
```

```
        "The date and time when the value of any of the
         admin-state, oper-state, usage-state, alarm-status, or
         standby-status changed for this entity.

         If there has been no change since the last
         re-initialization of the local system, this node contains
         the date and time of local system initialization.  If
         there has been no change since the entity was added to
         the local system, this node contains the date and time of
         the insertion.";
      reference "RFC 4268: entStateLastChanged";
    }

    leaf admin-state {
      type entity-admin-state;
      description
        "The administrative state for this entity.

         This node refers to an entities administrative permission
         to service both other entities within its containment
         hierarchy as well other users of its services defined by
         means outside the scope of this module.

         Some physical entities exhibit only a subset of the
         remaining administrative state values.  Some entities
         cannot be locked, and hence this node exhibits only the
         'unlocked' state.  Other entities cannot be shutdown
         gracefully, and hence this node does not exhibit the
         'shutting-down' state.";
      reference "RFC 4268: entStateAdmin";
    }

    leaf oper-state {
      type entity-oper-state;
      description
        "The operational state for this entity.

         Note that this node does not follow the administrative
         state.  An administrative state of down does not predict
         an operational state of disabled.

         Note that some implementations may not be able to
         accurately report oper-state while the admin-state node
         has a value other than 'unlocked'.  In these cases, this
         node MUST have a value of 'unknown'.";
      reference "RFC 4268: entStateOper";
    }
```

```
          leaf usage-state {
            type entity-usage-state;
            description
              "The usage state for this entity.

               This node refers to an entity's ability to service more
               physical entities in a containment hierarchy.

               Some entities will exhibit only a subset of the usage
               state values.  Entities that are unable to ever service
               any entities within a containment hierarchy will always
               have a usage state of 'busy'.  Some entities will only
               ever be able to support one entity within its containment
               hierarchy and will therefore only exhibit values of
               'idle' and 'busy'.";
            reference "RFC 4268, entStateUsage";
          }

          leaf alarm-status {
            type entity-alarm-status;
            description
              "The alarm status for this entity.  It does not include
               the alarms raised on child components within its
               containment hierarchy.";
            reference "RFC 4268: entStateAlarm";
          }

          leaf standby-status {
            type entity-standby-status;
            description
              "The standby status for this entity.

               Some entities will exhibit only a subset of the
               remaining standby state values.  If this entity
               cannot operate in a standby role, the value of this
               node will always be 'providing-service'.";
            reference "RFC 4268: entStateStandby";
          }
        }

        container sensor-data {
          when 'derived-from-or-self(../class,
                                     "iana-entity", "sensor")' {
            description
              "Sensor data nodes present for any entity of type
               'sensor'";
          }
          if-feature entity-sensor;
```

```
          description
            "Sensor-related nodes.";
          reference "RFC 3433: Entity Sensor MIB";

          leaf data-type {
            type entity-sensor-data-type;
            description
              "The type of data units associated with the
               sensor value";
            reference "RFC 3433: entPhySensorType";
          }

          leaf data-scale {
            type entity-sensor-data-scale;
            description
              "The (power of 10) scaling factor associated
               with the sensor value";
            reference "RFC 3433: entPhySensorScale";
          }

          leaf precision {
            type entity-sensor-precision;
            description
              "The number of decimal places of precision
               associated with the sensor value";
            reference "RFC 3433: entPhySensorPrecision";
          }

          leaf value {
            type entity-sensor-value;
            description
              "The most recent measurement obtained by the server
               for this sensor.";
            reference "RFC 3433: entPhySensorValue";
          }

          leaf oper-status {
            type entity-sensor-status;
            description
              "The operational status of the sensor.";
            reference "RFC 3433: entPhySensorOperStatus";
          }

          leaf sensor-units-display {
            type string;
            description
              "A textual description of the data units that should be
               used in the display of the sensor value.";
```

```
            reference "RFC 3433: entPhySensorUnitsDisplay";
          }

          leaf value-timestamp {
            type yang:date-and-time;
            description
              "The time the status and/or value of this sensor was
               last obtained by the server.";
            reference "RFC 3433: entPhySensorValueTimeStamp";
          }

          leaf value-update-rate {
            type uint32;
            units "milliseconds";
            description
              "An indication of the frequency that the server updates
               the associated 'value' node, representing in
               milliseconds.  The value zero indicates:

                - the sensor value is updated on demand (e.g.,
                  when polled by the server for a get-request),
                - the sensor value is updated when the sensor
                  value changes (event-driven),
                - the server does not know the update rate.";
            reference "RFC 3433: entPhySensorValueUpdateRate";
          }
        }
      }
    }

    /*
     * Configuration data nodes
     */

    container entity {
      if-feature entity-config;
      description
        "Configuration parameters for physical entities.";

      list physical-entity {
        key name;
        description
          "List of configuration data for physical entities.";

        leaf name {
          type string;
          description
            "Administrative name assigned to this physical entity.
```

```
            No restrictions apply.";
        }

        leaf serial-num {
          type string;
          description
            "The vendor-specific serial number string for the physical
             entity.  The preferred value is the serial number string
             actually printed on the component itself (if present).

             This node is indented to be used for physical entities
             for which the server cannot determine the serial number.";
          reference "RFC 6933: entPhysicalSerialNum";
        }

        leaf alias {
          type string {
            length "0 .. 32";
          }
          description
            "This node is an 'alias' name for the physical entity, as
             specified by a network manager, and provides a non-volatile
             'handle' for the physical entity.";
          reference "RFC 6933: entPhysicalAlias";
        }

        leaf asset-id {
          type string {
            length "0 .. 32";
          }
          description
            "This node is a user-assigned asset tracking identifier
             (as specified by a network manager) for the physical
             entity";
          reference "RFC 6933: entPhysicalAssetID";
        }

        leaf-list uri {
          type inet:uri;
          description
            "This node contains identification information about the
             physical entity.";
          reference "RFC 6933: entPhysicalUris";
        }

        leaf admin-state {
          if-feature entity-state;
          type entity-admin-state;
```

```
          description
            "The administrative state for this entity.

             This node refers to an entity's administrative
             permission to service both other entities within
             its containment hierarchy as well other users of
             its services defined by means outside the scope
             of this module.

             Some physical entities exhibit only a subset of the
             remaining administrative state values.  Some entities
             cannot be locked, and hence this node exhibits only
             the 'unlocked' state.  Other entities cannot be shutdown
             gracefully, and hence this node does not exhibit the
             'shutting-down' state.";
          reference "RFC 4268, entStateAdmin";
        }
      }
    }

    /*
     * Notifications
     */

    notification ent-config-change {
      description
        "An ent-config-change notification is generated when the value
         of /entity-state/last-change changes.";
      reference "RFC 6933, entConfigChange";
    }

    notification ent-state-oper-enabled {
      if-feature entity-state;
      description
        "An ent-state-oper-enabled notification signifies that
         an entity has transitioned into the 'enabled' state.";

      leaf name {
        type leafref {
          path "/entity-state/physical-entity/name";
        }
        description
          "The name of the entity that has transitioned into the
           'enabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/entity-state/physical-entity/state/admin-state";
```

```
        }
        description
          "The administrative state for the entity.";
      }
      leaf alarm-status {
        type leafref {
          path "/entity-state/physical-entity/state/alarm-status";
        }
        description
          "The alarm status for the entity.";
      }
      reference "RFC 4268, entStateOperEnabled";
    }

    notification ent-state-oper-disabled {
      if-feature entity-state;
      description
        "An ent-state-oper-disabled notification signifies that
         an entity has transitioned into the 'disabled' state.";

      leaf name {
        type leafref {
          path "/entity-state/physical-entity/name";
        }
        description
          "The name of the entity that has transitioned into the
           'disabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/entity-state/physical-entity/state/admin-state";
        }
        description
          "The administrative state for the entity.";
      }
      leaf alarm-status {
        type leafref {
          path "/entity-state/physical-entity/state/alarm-status";
        }
        description
          "The alarm status for the entity.";
      }
      reference "RFC 4268, entStateOperDisabled";
    }

  }

   <CODE ENDS>
```

Move this to a separate document?:

<CODE BEGINS> file "iana-entity@2016-05-13.yang"

```
module iana-entity {
  namespace "urn:ietf:params:xml:ns:yang:iana-entity";
  prefix ianaent;

  import ietf-entity {
    prefix ent;
  }

  organization "IANA";
  contact
    "        Internet Assigned Numbers Authority

     Postal: ICANN
             4676 Admiralty Way, Suite 330
             Marina del Rey, CA 90292

     Tel:    +1 310 823 9358
     <mailto:iana@iana.org>";


  description
    "IANA defined identities for physical class.";
  reference
    "https://www.iana.org/assignments/ianaentity-mib/ianaentity-mib";

  // 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 2016-05-13 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for Entity Management";
  }

  identity unknown {
    base ent:entity-physical-class;
    description
      "This identity is applicable if the physical entity class
       is unknown to the server.";
  }
```

```
    identity chassis {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is an overall container for networking equipment.  Any class of
         physical entity, except a stack, may be contained within a
         chassis; a chassis may only be contained within a stack.";
    }

    identity backplane {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of device for aggregating and forwarding
         networking traffic, such as a shared backplane in a modular
         ethernet switch.  Note that an implementation may model a
         backplane as a single physical entity, which is actually
         implemented as multiple discrete physical components (within a
         chassis or stack).";
    }

    identity container {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is capable of containing one or more removable physical
         entities, possibly of different types.  For example, each
         (empty or full) slot in a chassis will be modeled as a
         container.  Note that all removable physical entities should be
         modeled within a container entity, such as field-replaceable
         modules, fans, or power supplies.  Note that all known
         containers should be modeled by the agent, including empty
         containers.";
    }

    identity power-supply {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is a power-supplying component.";
    }

    identity fan {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is a fan or other heat-reduction component.";
    }
```

```
    identity sensor {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of sensor, such as a temperature sensor within a
         router chassis.";
    }

    identity module {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of self-contained sub-system.  If a 'module'
         entity is removable, then it should be modeled within a
         container entity; otherwise, it should be modeled directly
         within another physical entity (e.g., a chassis or another
         module).";
    }

    identity port {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of networking port, capable of receiving and/or
         transmitting networking traffic.";
    }

    identity stack {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of super-container (possibly virtual) intended to
         group together multiple chassis entities.  A stack may be
         realized by a 'virtual' cable, a real interconnect cable
         attached to multiple chassis, or multiple interconnect cables.
         A stack should not be modeled within any other physical
         entities, but a stack may be contained within another stack.
         Only chassis entities should be contained within a stack.";
    }

    identity cpu {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of central processing unit.";
    }

    identity energy-object {
```

```
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of energy object, i.e., a piece of equipment that
         is part of or attached to a communications network that is
         monitored, controlled, or aids in the management of another
         device for Energy Management.";
    }

    identity battery {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of battery.";
    }

    identity storage-drive {
      base ent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of entity with data storage capability as main
         functionality, e.g., disk drive (HDD), solid state device
         (SSD), hybrid (SSHD), object storage (OSD) or other.";
    }
  }
```

<CODE ENDS>

8. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: TBD

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
  name:         ietf-entity
  namespace:    TBD
  prefix:       ent
  reference:    RFC XXXX
```

9. Security Considerations

TBD

10. Acknowledgements

TBD

11. Normative References

[I-D.ietf-netmod-rfc6020bis]
Bjorklund, M., "The YANG 1.1 Data Modeling Language", draft-ietf-netmod-rfc6020bis-12 (work in progress), April 2016.

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

[RFC3433] Bierman, A., Romascanu, D., and K. Norseth, "Entity Sensor Management Information Base", RFC 3433, DOI 10.17487/RFC3433, December 2002, <http://www.rfc-editor.org/info/rfc3433>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4268] Chisholm, S. and D. Perkins, "Entity State MIB", RFC 4268, DOI 10.17487/RFC4268, November 2005, <http://www.rfc-editor.org/info/rfc4268>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <http://www.rfc-editor.org/info/rfc6933>.

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Dan Romascanu
Avaya

Email: dromasca@avaya.com