Network Working Group   A. Bierman
INTERNET-DRAFT   Yumaworks
Category: Standards Track   D. Romascanu
Expires: April 10, 2013   AVAYA
                                                            J. Quittek
                                                         NEC Europe Ltd.
                                                       Mouli Chandramouli
                                                       Cisco Systems, Inc.
                                                         October 10, 2012

Entity MIB (Version 4)
draft-ietf-eman-rfc4133bis-03

Abstract

This memo defines a portion of the Management Information Base (MIB) for
use with network management protocols in the Internet community. In
particular, it describes managed objects used for managing multiple
logical and physical entities managed by a single SNMP agent. This
document specifies version of the Entity MIB, which obsoletes version 3
[RFC4133].

Conventions used in this document

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 RFC 2119 [RFC2119].

Table of Contents

1. The SNMP Management Framework

For a detailed overview of the documents that describe the current
Internet-Standard Management Framework, please refer to section 7 of
RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed
the Management Information Base or MIB. MIB objects are generally
accessed through the Simple Network Management Protocol (SNMP).
Objects in the MIB are defined using the mechanisms defined in the
Structure of Management Information (SMI). This memo specifies a MIB
module that is compliant to the SMIv2, which is described in STD 58,
RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580
[RFC2580].

2. Overview

There is a need for a standardized way of representing a single
agent, which supports multiple instances of one MIB. This is
presently true for at least 3 standard MIBs, and is likely to become
true for more and more MIBs as time passes. For example:

- multiple instances of a bridge supported within a single device
  that has a single agent;

- multiple repeaters supported by a single agent;

- multiple OSPF backbone areas, each operating as part of its own
  Autonomous System, and each identified by the same area-id (e.g.,
  0.0.0.0), supported inside a single router with one agent.

The single agent present in each of these cases implies a
relationship binds these entities. Effectively, there is some
"overall" physical entity which houses the sum of the things managed
by that one agent, i.e., there are multiple "logical" entities within
a single physical entity. Sometimes, the overall physical entity
contains multiple (smaller) physical entities, and each logical
entity is associated with a particular physical entity. Sometimes,
the overall physical entity is a "compound" of multiple physical
entities (e.g., a stack of stackable hubs).

What is needed is a way to determine exactly which logical entities
are managed by the agent (with some version of SNMP) in order to
communicate with the agent about a particular logical entity. When
different logical entities are associated with different physical
entities within the overall physical entity, it is also useful to be
able to use this information to distinguish between logical entities.

In these situations, there is no need for varbinds for multiple
logical entities to be referenced in the same SNMP message (although
that might be useful in the future). Rather, it is sufficient, and
in some situations preferable, to have the context/community in the
message identify the logical entity to which the varbinds apply.

Version 2 of this MIB addresses new requirements, which have emerged
since the publication of the first Entity MIB (RFC 2037 [RFC2037]).
There is a need for a standardized way of providing non-volatile,
administratively-assigned identifiers for physical components
represented with the Entity MIB. There is also a need to align the
Entity MIB with the SNMPv3 administrative framework (STD 62, RFC 3411
[RFC3411]). Implementation experience has shown that additional
physical component attributes are also desirable.

Version 3 of this MIB addresses new requirements, which have emerged
since the publication of the second Entity MIB (RFC 2737 [RFC2737]).
There is a need to identify physical entities that are central
processing units (CPUs) and a need to provide a textual convention
that identifies an entPhysicalIndex value or zero, where the value
zero has application-specific semantics. Two new objects have been
added to the entPhysicalTable to identify the manufacturing date and
provide additional URIs for a particular physical entity.

2.1. Terms

Some new terms are used throughout this document:

- Naming Scope
  A "naming scope" represents the set of information that may be
  potentially accessed through a single SNMP operation. All
  instances within the naming scope share the same unique identifier
  space. For SNMPv1, a naming scope is identified by the value of the
  associated 'entLogicalCommunity' instance. For SNMPv3, the term
  'context' is used instead of 'naming scope'. The complete
  definition of an SNMP context can be found in section 3.3.1 of RFC
  3411 [RFC3411].

- Multi-Scoped Object
  A MIB object, for which identical instance values identify
  different managed information in different naming scopes, is called
  a "multi-scoped" MIB object.

- Single-Scoped Object
  A MIB object, for which identical instance values identify the same
  managed information in different naming scopes, is called a
  "single-scoped" MIB object.

   - Logical Entity
     A managed system contains one or more logical entities, each
     represented by at most one instantiation of each of a particular
     set of MIB objects. A set of management functions is associated
     with each logical entity. Examples of logical entities include
     routers, bridges, print-servers, etc.

   - Physical Entity
     A "physical entity" or "physical component" represents an
     identifiable physical resource within a managed system. Zero or
     more logical entities may utilize a physical resource at any given
     time. Determining which physical components are represented by an
     agent in the EntPhysicalTable is an implementation-specific matter.
      Typically, physical resources (e.g., communications ports,
     backplanes, sensors, daughter-cards, power supplies, the overall
     chassis), which can be managed via functions associated with one or
     more logical entities, are included in the MIB.

   - Containment Tree
     Each physical component may be modeled as 'contained' within
     another physical component. A "containment-tree" is the conceptual
     sequence of entPhysicalIndex values that uniquely specifies the
     exact physical location of a physical component within the managed
     system. It is generated by 'following and recording' each
     'entPhysicalContainedIn' instance 'up the tree towards the root',
     until a value of zero indicating no further containment is found.

2.2. Relationship to Community Strings

   For community-based SNMP, differentiating logical entities is one
   (but not the only) purpose of the community string (RFC 1157
   [RFC1157]). This is accommodated by representing each community
   string as a logical entity.

   Note that different logical entities may share the same naming scope
   and, therefore, the same values of entLogicalCommunity. This is
   possible, providing they have no need for the same instance of a MIB
   object to represent different managed information.

2.3. Relationship to SNMP Contexts

   Version 2 of the Entity MIB contains support for associating SNMPv3
   contexts with logical entities. Two new MIB objects, defining an
   SnmpEngineID and ContextName pair, are used together to identify an
   SNMP context associated with a logical entity. This context can be
   used (in conjunction with the entLogicalTAddress and
   entLogicalTDomain MIB objects) to send SNMPv3 messages on behalf of a
   particular logical entity.

2.4. Relationship to Proxy Mechanisms

   The Entity MIB is designed to allow functional component discovery.
   The administrative relationships between different logical entities
   are not visible in any Entity MIB tables. A Network Management
   System (NMS) cannot determine whether MIB instances in different
   naming scopes are realized locally or remotely (e.g., via some proxy
   mechanism) by examining any particular Entity MIB objects.

   The management of administrative framework functions is not an
   explicit goal of the Entity MIB WG at this time. This new area of
   functionality may be revisited after some operational experience with
   the Entity MIB is gained.

   Note that for community-based versions of SNMP, a network
   administrator will likely be able to associate community strings with
   naming scopes that have proprietary mechanisms, as a matter of
   configuration. There are no mechanisms for managing naming scopes
   defined in this MIB.

2.5. Relationship to a Chassis MIB

   Some readers may recall that a previous IETF working group attempted
   to define a Chassis MIB. No consensus was reached by that working
   group, possibly because its scope was too broad. As such, it is not
   the purpose of this MIB to be a "Chassis MIB replacement", nor is it
   within the scope of this MIB to contain all the information which
   might be necessary to manage a "chassis". On the other hand, the
   entities represented by an implementation of this MIB might well be
   contained in a chassis.

2.6. Relationship to the Interfaces MIB

   The Entity MIB contains a mapping table identifying physical
   components that have 'external values' (e.g., ifIndex) associated
   with them within a given naming scope. This table can be used to
   identify the physical location of each interface in the ifTable (RFC
   2863 [RFC2863]). Because ifIndex values in different contexts are
   not related to one another, the interface to physical component
   associations are relative to the same logical entity within the
   agent.

   The Entity MIB also contains 'entPhysicalName' and 'entPhysicalAlias'
   objects, which approximate the semantics of the 'ifName' and
   'ifAlias' objects (respectively) from the Interfaces MIB [RFC2863],
   for all types of physical components.

2.7. Relationship to the Other MIBs

   The Entity MIB contains a mapping table identifying physical
   components that have identifiers from other standard MIBs associated
   with them. For example, this table can be used along with the
   physical mapping table to identify the physical location of each
   repeater port in the rptrPortTable, or each interface in the ifTable.

2.8. Relationship to Naming Scopes

   There is some question as to which MIB objects may be returned within
   a given naming scope. MIB objects which are not multi-scoped within
   a managed system are likely to ignore context information in
   implementation. In such a case, it is likely such objects will be
   returned in all naming scopes (e.g., not just the 'default' naming
   scope or the SNMPv3 default context).

   For example, a community string used to access the management
   information for logical device 'bridge2' may allow access to all the
   non-bridge related objects in the 'default' naming scope, as well as
   a second instance of the Bridge MIB (RFC 1493 [RFC1493]).

   The isolation of single-scoped MIB objects by the agent is an
   implementation-specific matter. An agent may wish to limit the
   objects returned in a particular naming scope to only the multi-
   scoped objects in that naming scope (e.g., system group and the
   Bridge MIB). In this case, all single-scoped management information
   would belong to a common naming scope (e.g., 'default'), which itself
   may contain some multi-scoped objects (e.g., system group).

2.9. Multiple Instances of the Entity MIB

   It is possible that more than one agent may exist in a managed
   system. In such cases, multiple instances of the Entity MIB
   (representing the same managed objects) may be available to an NMS.

   In order to reduce complexity for agent implementation, multiple
   instances of the Entity MIB are not required to be equivalent or even
   consistent. An NMS may be able to 'align' instances returned by
   different agents by examining the columns of each table, but vendor-
   specific identifiers and (especially) index values are likely to be
   different. Each agent may be managing different subsets of the
   entire chassis as well.

   When all of a physically-modular device is represented by a single
   agent, the entry (for which entPhysicalContainedIn has the value
   zero) would likely have 'chassis' as the value of its
   entPhysicalClass. Alternatively, for an agent on a module where the

agent represents only the physical entities on that module (not those
on other modules), the entry (for which entPhysicalContainedIn has
the value zero) would likely have 'module' as the value of its
entPhysicalClass.

An agent implementation of the entLogicalTable is not required to
contain information about logical entities managed primarily by other
agents. That is, the entLogicalTAddress and entLogicalTDomain
objects in the entLogicalTable are provided to support an historical
multiplexing mechanism, not to identify other SNMP agents.

Note that the Entity MIB is a single-scoped MIB, in the event an
agent represents the MIB in different naming scopes.

2.10. Re-Configuration of Entities

Most of the MIB objects defined in this MIB have, at most, a read-
only MAX-ACCESS clause. This is a conscious decision by the working
group to limit this MIB's scope. The second version of the Entity
MIB allows a network administrator to configure some common
attributes of physical components.

2.11. Textual Convention Change

Version 1 of the Entity MIB contains three MIB objects defined with
the (now obsolete) DisplayString textual convention. In version 2 of
the Entity MIB, the syntax for these objects has been updated to use
the (now preferred) SnmpAdminString textual convention.

The entmib working group (which was in charge with the document at
that point) realized that this change is not strictly supported by
SMIv2. In their judgment, the alternative of deprecating the old
objects and defining new objects would have had a more adverse impact
on backward compatibility and interoperability, given the particular
semantics of these objects.

2.12. MIB Structure

The Entity MIB contains five groups of MIB objects:

- entityPhysical group
  Describes the physical entities managed by a single agent.

- entityLogical group
  Describes the logical entities managed by a single agent.

- entityMapping group
  Describes the associations between the physical entities, logical
  entities, interfaces, and non-interface ports managed by a single
  agent.

- entityGeneral group
  Describes general system attributes shared by potentially all types
  of entities managed by a single agent.

- entityNotifications group
  Contains status indication notifications.

2.12.1. entityPhysical Group

   This group contains a single table to identify physical system
   components, called the entPhysicalTable.

   The entPhysicalTable contains one row per physical entity, and must
   always contain at least one row for an "overall" physical entity,
   which should have an entPhysicalClass value of 'stack(11)',
   'chassis(3)' or 'module(9)'.

   Each row is indexed by an arbitrary, small integer, and contains a
   description and type of the physical entity. It also optionally
   contains the index number of another entPhysicalEntry, indicating a
   containment relationship between the two.

   Version 2 of the Entity MIB provides additional MIB objects for each
   physical entity. Some common read-only attributes have been added,
   as well as three writable string objects.

   - entPhysicalAlias
     This string can be used by an NMS as a non-volatile identifier for
     the physical component. Maintaining a non-volatile string for
     every physical component represented in the entPhysicalTable can be
     costly and unnecessary. An agent may algorithmically generate
     'entPhysicalAlias' strings for particular entries (e.g., based on
     the entPhysicalClass value).

   - entPhysicalAssetID
     This string is provided to store a user-specific asset identifier
     for removable physical components. In order to reduce the non-
     volatile storage needed by a particular agent, a network
     administrator should only assign asset identifiers to physical
     entities that are field-replaceable (i.e., not permanently
     contained within another physical entity).

       - entPhysicalSerialNum
         This string is provided to store a vendor-specific serial number
         string for physical components. This writable object is used when
         an agent cannot identify the serial numbers of all installed
         physical entities, and a network administrator wishes to configure
         the non-volatile serial number strings manually (via an NMS
         application).

    Version 3 of the Entity MIB provides two additional MIB objects for
    each physical entity:

    - entPhysicalMfgDate
      This object contains the date of manufacturing of the managed
      entity. If the manufacturing date is unknown or not supported the
      object is not instantiated. The special value '0000000000000000'H
      may also be returned in this case.

    - entPhysicalUris
      This object provides additional identification information about
      the physical entity.

      This object contains one or more Uniform Resource Identifiers
      (URIs) and, therefore, the syntax of this object must conform to
      RFC 3986 [RFC3986] section 2. Uniform Resource Names (URNs), RFC
      3406 [RFC3406], are resource identifiers with the specific
      requirements for enabling location independent identification of a
      resource, as well as longevity of reference. URNs are part of the
      larger URI family with the specific goal of providing persistent
      naming of resources. URI schemes and URN name spaces are
      registered by IANA (see http://www.iana.org/assignments/uri-schemes
      and http://www.iana.org/assignments/urn-namespaces).

      For example, the entPhysicalUris object may be used to encode a URI
      containing a Common Language Equipment Identifier (CLEI) URN for
      the managed physical entity. The URN name space for CLEIs is
      defined in [RFC4152], and the CLEI format is defined in
      [T1.213][T1.213a]. For example, an entPhysicalUris instance may
      have the value of

          URN:CLEI:D4CE18B7AA

      [RFC3986] and [RFC4152] identify this as a URI in the CLEI URN name
      space. The specific CLEI code, D4CE18B7AA, is based on the example
      provided in [T1.213a].

      Multiple URIs may be present and are separated by white space
      characters. Leading and trailing white space characters are
      ignored.

If no additional identification information is known about the
physical entity or supported, the object is not instantiated.

2.12.2. entityLogical Group

   This group contains a single table to identify logical entities,
   called the entLogicalTable.

   The entLogicalTable contains one row per logical entity. Each row is
   indexed by an arbitrary, small integer and contains a name,
   description, and type of the logical entity. It also contains
   information to allow access to the MIB information for the logical
   entity. This includes SNMP versions that use a community name (with
   some form of implied context representation) and SNMP versions that
   use the SNMP ARCH [RFC3411] method of context identification.

   If an agent represents multiple logical entities with this MIB, then
   this group must be implemented for all logical entities known to the
   agent.

   If an agent represents a single logical entity, or multiple logical
   entities within a single naming scope, then implementation of this
   group may be omitted by the agent.

2.12.3. entityMapping Group

   This group contains three tables to identify associations between
   different system components.

   - entLPMappingTable
     This table contains mappings between entLogicalIndex values
     (logical entities) and entPhysicalIndex values (the physical
     components supporting that entity). A logical entity can map to
     more than one physical component, and more than one logical entity
     can map to (share) the same physical component. If an agent
     represents a single logical entity, or multiple logical entities
     within a single naming scope, then implementation of this table may
     be omitted by the agent.

   - entAliasMappingTable
     This table contains mappings between entLogicalIndex,
     entPhysicalIndex pairs, and 'alias' object identifier values. This
     allows resources managed with other MIBs (e.g., repeater ports,
     bridge ports, physical and logical interfaces) to be identified in
     the physical entity hierarchy. Note that each alias identifier is
     only relevant in a particular naming scope. If an agent represents
     a single logical entity, or multiple logical entities within a

single naming scope, then implementation of this table may be
omitted by the agent.

- entPhysicalContainsTable
  This table contains simple mappings between
  'entPhysicalContainedIn' values for each container/'containee'
  relationship in the managed system. The indexing of this table
  allows an NMS to quickly discover the 'entPhysicalIndex' values for
  all children of a given physical entity.

2.12.4. entityGeneral Group

This group contains general information relating to the other object
groups.

At this time, the entGeneral group contains a single scalar object
(entLastChangeTime), which represents the value of sysUptime when any
part of the Entity MIB configuration last changed.

2.12.5. entityNotifications Group

This group contains notification definitions relating to the overall
status of the Entity MIB instantiation.

2.13. Multiple Agents

Even though a primary motivation for this MIB is to represent the
multiple logical entities supported by a single agent, another
motivation is to represent multiple logical entities supported by
multiple agents (in the same "overall" physical entity). Indeed, it
is implicit in the SNMP architecture that the number of agents is
transparent to a network management station.

However, there is no agreement at this time as to the degree of
cooperation that should be expected for agent implementations.
Therefore, multiple agents within the same managed system are free to
implement the Entity MIB independently. (For more information, refer
to Section 2.9, "Multiple Instances of the Entity MIB".)

2.14. Changes Since RFC 2037

2.14.1. Textual Conventions

The PhysicalClass TC text has been clarified, and a new enumeration
to support 'stackable' components has been added. The
SnmpEngineIdOrNone TC has been added to support SNMPv3.

2.14.2. New entPhysicalTable Objects

   The entPhysicalHardwareRev, entPhysicalFirmwareRev, and
   entPhysicalSoftwareRev objects have been added for revision
   identification.

   The entPhysicalSerialNum, entPhysicalMfgName, entPhysicalModelName,
   and entPhysicalIsFru objects have been added for better vendor
   identification for physical components. In the event the agent
   cannot identify this information, the entPhysicalSerialNum object can
   be set by a management station.

   The entPhysicalAlias and entPhysicalAssetID objects have been added
   for better user component identification. These objects are intended
   to be set by a management station and preserved by the agent across
   restarts.

2.14.3. New entLogicalTable Objects

   The entLogicalContextEngineID and entLogicalContextName objects have
   been added to provide an SNMP context for SNMPv3 access on behalf of
   a logical entity.

2.14.4. Bug Fixes

   A bug was fixed in the entLogicalCommunity object. The subrange was
   incorrect (1..255) and is now (0..255). The description clause has
   also been clarified. This object is now deprecated.

   The entLastChangeTime object description has been changed to
   generalize the events that cause an update to the last change
   timestamp.

   The syntax was changed from DisplayString to SnmpAdminString for the
   entPhysicalDescr, entPhysicalName, and entLogicalDescr objects.

2.15. Changes Since RFC 2737

2.15.1. Textual Conventions

   The PhysicalIndexOrZero TC has been added to allow objects to
   reference an entPhysicalIndex value or zero. The PhysicalClass TC
   has been extended to support a new enumeration for central processing
   units.

2.15.2. New Objects

   The entPhysicalMfgDate object has been added to the entPhysicalTable
   to provide the date of manufacturing of the managed entity.

The entPhysicalUris object has been added to the entPhysicalTable to
provide additional identification information about the physical
entity, such as a Common Language Equipment Identifier (CLEI) URN.

2.15.3. Bug Fixes

The syntax was changed from INTEGER to Integer32 for the
entPhysicalParentRelPos, entLogicalIndex, and
entAliasLogicalIndexOrZero objects, and from INTEGER to
PhysicalIndexOrZero for the entPhysicalContainedIn object.

2.16. Changes Since RFC 4133

2.16.1. MIB module addition Creation of a new MIB module IANA-ENTITY-
  MIB which makes the PhysicalIndex TC an IANA-maintained Textual
  Convention. Over time, there is the need to add new enumerated values
  for PhysicalClass. If the syntax of IANAPhysicalClass were defined
  in this MIB module then a new version of this MIB would have to be
  re-issued in order to define new values.

2.16.2. Modification to some of the MIB objects

Addition of a new MIB object to the entPhysicalTable -
entPhysicalUUID. In comparison to entPhysicalUris the new object is
read-only and restricted to a fixed size to allow only for RFC 4122
[RFC4122] compliant values.

Creation of a new MODULE-COMPLIANCE module entity4CRCompliance for
devices with constrained resources like batteries, which might
require a limited number of objects to be supported
(entPhysicalClass, entPhysicalName, entPhysicalUUID)

2.16.3. New TC for Universal Unique Identifier

Two new Textual Conventions (TC) UUID and UUIDorZero were created to
represent a Universal Unique Identifier (UUID), with a syntax that
conforms to RFC 4122, section 4.1. Defining them as TCs will allow
for future re-use in other MIB modules that will import the TC. These
Textual Conventions are included in the UUID-TC-MIB module.
3. MIB Definitions

3.1. ENTITY MIB

ENTITY-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, mib-2, NOTIFICATION-TYPE,
    Integer32

```
        FROM SNMPv2-SMI
    TDomain, TAddress, TEXTUAL-CONVENTION,
    AutonomousType, RowPointer, TimeStamp, TruthValue,
    DateAndTime
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    Uri
        FROM URI-TC-MIB
    UUIDorZero
        FROM UUID-TC-MIB
    IANAPhysicalClass
        FROM IANA-ENTITY-MIB;

entityMIB MODULE-IDENTITY
    LAST-UPDATED "201210090000Z"
    ORGANIZATION "IETF Energy Management Working Group"
    CONTACT-INFO
            "          WG E-mail: eman@ietf.org
                    Mailing list subscription info:
                      http://www.ietf.org/mailman/listinfo/eman

                    Andy Bierman
                    Yumaworks
                    Email: andy@yumaworks.com

                    Dan Romascanu
                    AVAYA
                    Park Atidim, Bldg. #3
                    Tel Aviv, 61581
                    Israel
                    Phone: +972-3-6458414
                    Email: dromasca@avaya.com

                    Juergen Quittek
                    NEC Europe Ltd.
                    Network Research Division
                    Kurfuersten-Anlage 36
                    Heidelberg  69115
                    DE
                    Phone: +49 6221 4342-115
                    Email: quittek@neclab.eu

                    Mouli Chandramouli
                    Cisco Systems, Inc.
                    Sarjapur Outer Ring Road
```

                        Bangalore 560103
                        IN
                        Phone: +91 80 4429 2409
                        Email: moulchan@cisco.com"

        DESCRIPTION
                "The MIB module for representing multiple logical
                entities supported by a single SNMP agent.

                Copyright (C) The Internet Society (2012). This
                version of this MIB module is part of RFC xxxx; see
                the RFC itself for full legal notices."

REVISION   "201210090000Z"
        DESCRIPTION
                "Entity MIB (Version 4).
                 This revision obsoletes RFC 4133.
                 Additions:

                 Changes:
                   - according to comments made on draft-eman-rfc4133bis-02
                 The UUIDorZero TC is now imported from UUID-TC-MIB.
                 This version published as RFC xxxx."

    REVISION   "201210040000Z"
     DESCRIPTION
                "Entity MIB (Version 4).
                 This revision obsoletes RFC 4133.
                 Additions:

                 Changes:
                   - according to comments made on draft-eman-rfc4133bis-01
                 split the PhysicalUUID TC into two TCs
                 This version published as RFC xxxx."

    REVISION   "201210030000Z"
     DESCRIPTION
                "Entity MIB (Version 4).
                 This revision obsoletes RFC 4133.
                 Additions:
                   - PhysicalUUID TC
                 Changes:
                   - according to comments made on draft-eman-rfc4133bis-00

```
             This version published as RFC xxxx."

     REVISION         "201209100000Z"
     DESCRIPTION
             "Entity MIB (Version 4).
              This revision obsoletes RFC 4133.
              Additions:
                 -
              Changes:
                 - according to comments made on draft-chandramouli-01
              This version published as RFC xxxx."

     REVISION         "201206100000Z"
     DESCRIPTION
             "Initial Version of Entity MIB (Version 4).
              This revision obsoletes RFC 4133.
              Additions:
                 -
              Changes:
                 -
              This version published as RFC xxxx."


     REVISION         "200508100000Z"
     DESCRIPTION
             "Initial Version of Entity MIB (Version 3).
              This revision obsoletes RFC 2737.
              Additions:
                 - cpu(12) enumeration added to IANAPhysicalClass TC
                 - DISPLAY-HINT clause to PhysicalIndex TC
                 - PhysicalIndexOrZero TC
                 - entPhysicalMfgDate object
                 - entPhysicalUris object
              Changes:
                 - entPhysicalContainedIn SYNTAX changed from
                   INTEGER to PhysicalIndexOrZero

              This version published as RFC 4133."

     REVISION         "199912070000Z"
     DESCRIPTION
             "Initial Version of Entity MIB (Version 2).
              This revision obsoletes RFC 2037.
              This version published as RFC 2737."

     REVISION         "199610310000Z"
     DESCRIPTION
             "Initial version (version 1), published as
```

                    RFC 2037."
      ::= { mib-2 47 }

entityMIBObjects OBJECT IDENTIFIER ::= { entityMIB 1 }

-- MIB contains four groups
entityPhysical OBJECT IDENTIFIER ::= { entityMIBObjects 1 }
entityLogical OBJECT IDENTIFIER ::= { entityMIBObjects 2 }
entityMapping OBJECT IDENTIFIER ::= { entityMIBObjects 3 }
entityGeneral OBJECT IDENTIFIER ::= { entityMIBObjects 4 }

-- Textual Conventions
PhysicalIndex ::= TEXTUAL-CONVENTION
    DISPLAY-HINT   "d"
    STATUS   current
    DESCRIPTION
            "An arbitrary value that uniquely identifies the physical
            entity. The value should be a small, positive integer.
            Index values for different physical entities are not
            necessarily contiguous."
    SYNTAX Integer32 (1..2147483647)

PhysicalIndexOrZero ::= TEXTUAL-CONVENTION
    DISPLAY-HINT   "d"
    STATUS   current
    DESCRIPTION
            "This textual convention is an extension of the
            PhysicalIndex convention, which defines a greater than zero
            value used to identify a physical entity. This extension
            permits the additional value of zero. The semantics of the
            value zero are object-specific and must, therefore, be
            defined as part of the description of any object that uses
            this syntax. Examples of the usage of this extension are
            situations where none or all physical entities need to be
            referenced."
    SYNTAX Integer32 (0..2147483647)

SnmpEngineIdOrNone ::= TEXTUAL-CONVENTION
    STATUS   current
    DESCRIPTION
            "A specially formatted SnmpEngineID string for use with the
            Entity MIB.

            If an instance of an object of SYNTAX SnmpEngineIdOrNone has
            a non-zero length, then the object encoding and semantics
            are defined by the SnmpEngineID textual convention (see STD
            62, RFC 3411 [RFC3411]).

            If an instance of an object of SYNTAX SnmpEngineIdOrNone
            contains a zero-length string, then no appropriate
            SnmpEngineID is associated with the logical entity (i.e.,
            SNMPv3 is not supported)."
    SYNTAX OCTET STRING (SIZE(0..32)) -- empty string or SnmpEngineID

--   The Physical Entity Table
entPhysicalTable OBJECT-TYPE
    SYNTAX   SEQUENCE OF EntPhysicalEntry
    MAX-ACCESS   not-accessible
    STATUS   current
    DESCRIPTION
            "This table contains one row per physical entity. There is
            always at least one row for an 'overall' physical entity."
    ::= { entityPhysical 1 }

entPhysicalEntry   OBJECT-TYPE
    SYNTAX   EntPhysicalEntry
    MAX-ACCESS   not-accessible

    STATUS   current
    DESCRIPTION
            "Information about a particular physical entity.

            Each entry provides objects (entPhysicalDescr,
            entPhysicalVendorType, and entIANAPhysicalClass) to help an NMS
            identify and characterize the entry, and objects
            (entPhysicalContainedIn and entPhysicalParentRelPos) to help
            an NMS relate the particular entry to other entries in this
            table."
    INDEX   { entPhysicalIndex }
    ::= { entPhysicalTable 1 }

EntPhysicalEntry ::= SEQUENCE {
     entPhysicalIndex   PhysicalIndex,
     entPhysicalDescr   SnmpAdminString,
     entPhysicalVendorType   AutonomousType,
     entPhysicalContainedIn   PhysicalIndexOrZero,

         entPhysicalClass   IANAPhysicalClass,
         entPhysicalParentRelPos   Integer32,
         entPhysicalName   SnmpAdminString,
         entPhysicalHardwareRev   SnmpAdminString,
         entPhysicalFirmwareRev   SnmpAdminString,
         entPhysicalSoftwareRev   SnmpAdminString,
         entPhysicalSerialNum   SnmpAdminString,
         entPhysicalMfgName   SnmpAdminString,
         entPhysicalModelName   SnmpAdminString,
         entPhysicalAlias   SnmpAdminString,
         entPhysicalAssetID   SnmpAdminString,
         entPhysicalIsFRU   TruthValue,
         entPhysicalMfgDate   DateAndTime,
         entPhysicalUris   Uri,
         entPhysicalUUID   UUIDorZero

    }

    entPhysicalIndex   OBJECT-TYPE
        SYNTAX   PhysicalIndex
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
                "The index for this entry."
        ::= { entPhysicalEntry 1 }

    entPhysicalDescr OBJECT-TYPE
        SYNTAX   SnmpAdminString
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "A textual description of physical entity. This object
                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."
        ::= { entPhysicalEntry 2 }

    entPhysicalVendorType OBJECT-TYPE
        SYNTAX   AutonomousType
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "An indication of the vendor-specific hardware type of the
                physical entity. Note that this is different from the
                definition of MIB-II's sysObjectID.

                An agent should set this object to an enterprise-specific

                registration identifier value indicating the specific
                equipment type in detail. The associated instance of
                entIANAPhysicalClass is used to indicate the general type of
                hardware device.

                If no vendor-specific registration identifier exists for
                this physical entity, or the value is unknown by this agent,
                then the value { 0 0 } is returned."
        ::= { entPhysicalEntry 3 }

entPhysicalContainedIn OBJECT-TYPE
        SYNTAX   PhysicalIndexOrZero
        MAX-ACCESS   read-only
        STATUS   current
        DESCRIPTION
                "The value of entPhysicalIndex for the physical entity which
                'contains' this physical entity. A value of zero indicates
                this physical entity is not contained in any other physical
                entity. Note that the set of 'containment' relationships
                define a strict hierarchy; that is, recursion is not
                allowed.

                In the event that a physical entity is contained by more
                than one physical entity (e.g., double-wide modules), this
                object should identify the containing entity with the lowest
                value of entPhysicalIndex."
        ::= { entPhysicalEntry 4 }

entPhysicalClass OBJECT-TYPE
        SYNTAX   IANAPhysicalClass
        MAX-ACCESS   read-only

        STATUS   current
        DESCRIPTION
                "An indication of the general hardware type of the physical
                entity.

                An agent should set this object to the standard enumeration
                value that most accurately indicates the general class of
                the physical entity, or the primary class if there is more
                than one entity.

                If no appropriate standard registration identifier exists
                for this physical entity, then the value 'other(1)' is
                returned. If the value is unknown by this agent, then the
                value 'unknown(2)' is returned."

     ::= { entPhysicalEntry 5 }

entPhysicalParentRelPos OBJECT-TYPE
     SYNTAX   Integer32 (-1..2147483647)
     MAX-ACCESS   read-only
     STATUS   current
     DESCRIPTION
             "An indication of the relative position of this 'child'
             component among all its 'sibling' components. Sibling
             components are defined as entPhysicalEntries that share the
             same instance values of each of the entPhysicalContainedIn
             and entPhysicalClass objects.

             An NMS can use this object to identify the relative ordering
             for all sibling components of a particular parent
             (identified by the entPhysicalContainedIn instance in each
             sibling entry).

             If possible, this value should match any external labeling
             of the physical component. For example, for a container
             (e.g., card slot) labeled as 'slot #3',
             entPhysicalParentRelPos should have the value '3'. Note
             that the entPhysicalEntry for the module plugged in slot 3
             should have an entPhysicalParentRelPos value of '1'.

             If the physical position of this component does not match
             any external numbering or clearly visible ordering, then
             user documentation or other external reference material
             should be used to determine the parent-relative position.
             If this is not possible, then the agent should assign a
             consistent (but possibly arbitrary) ordering to a given set
             of 'sibling' components, perhaps based on internal
             representation of the components.

             If the agent cannot determine the parent-relative position
             for some reason, or if the associated value of
             entPhysicalContainedIn is '0', then the value '-1' is
             returned. Otherwise, a non-negative integer is returned,
             indicating the parent-relative position of this physical
             entity.

             Parent-relative ordering normally starts from '1' and
             continues to 'N', where 'N' represents the highest
             positioned child entity. However, if the physical entities
             (e.g., slots) are labeled from a starting position of zero,
             then the first sibling should be associated with an
             entPhysicalParentRelPos value of '0'. Note that this
             ordering may be sparse or dense, depending on agent

            implementation.

            The actual values returned are not globally meaningful, as
            each 'parent' component may use different numbering
            algorithms. The ordering is only meaningful among siblings
            of the same parent component.

            The agent should retain parent-relative position values
            across reboots, either through algorithmic assignment or use
            of non-volatile storage."
     ::= { entPhysicalEntry 6 }

entPhysicalName OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "The textual name of the physical entity. The value of this
            object should be the name of the component as assigned by
            the local device and should be suitable for use in commands
            entered at the device's 'console'. This might be a text
            name (e.g., 'console') or a simple component number (e.g.,
            port or module number, such as '1'), depending on the
            physical component naming syntax of the device.

            If there is no local name, or if this object is otherwise
            not applicable, then this object contains a zero-length
            string.

            Note that the value of entPhysicalName for two physical
            entities will be the same in the event that the console
            interface does not distinguish between them, e.g., slot-1
            and the card in slot-1."
     ::= { entPhysicalEntry 7 }

entPhysicalHardwareRev   OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS read-only
    STATUS   current
    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).

            Note that if revision information is stored internally in a

            non-printable (e.g., binary) format, then the agent must
            convert such information to a printable format, in an
            implementation-specific manner.

            If no specific hardware revision string is associated with
            the physical component, or if this information is unknown to
            the agent, then this object will contain a zero-length
            string."
    ::= { entPhysicalEntry 8 }

entPhysicalFirmwareRev   OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS   read-only
    STATUS   current
    DESCRIPTION
            "The vendor-specific firmware revision string for the
            physical entity.

            Note that if revision information is stored internally in a
            non-printable (e.g., binary) format, then the agent must
            convert such information to a printable format, in an
            implementation-specific manner.

            If no specific firmware programs are associated with the
            physical component, or if this information is unknown to the
            agent, then this object will contain a zero-length string."
    ::= { entPhysicalEntry 9 }

entPhysicalSoftwareRev   OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS   read-only
    STATUS   current
    DESCRIPTION
            "The vendor-specific software revision string for the
            physical entity.

            Note that if revision information is stored internally in a
            non-printable (e.g., binary) format, then the agent must
            convert such information to a printable format, in an
            implementation-specific manner.

            If no specific software programs are associated with the
            physical component, or if this information is unknown to the
            agent, then this object will contain a zero-length string."
    ::= { entPhysicalEntry 10 }

entPhysicalSerialNum   OBJECT-TYPE
    SYNTAX   SnmpAdminString (SIZE (0..32))

          MAX-ACCESS read-write
          STATUS   current
          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).

                  On the first instantiation of an physical entity, the value
                  of entPhysicalSerialNum associated with that entity is set
                  to the correct vendor-assigned serial number, if this
                  information is available to the agent. If a serial number
                  is unknown or non-existent, the entPhysicalSerialNum will be
                  set to a zero-length string instead.

                  Note that implementations that can correctly identify the
                  serial numbers of all installed physical entities do not
                  need to provide write access to the entPhysicalSerialNum
                  object. Agents which cannot provide non-volatile storage
                  for the entPhysicalSerialNum strings are not required to
                  implement write access for this object.

                  Not every physical component will have a serial number, or
                  even need one. Physical entities for which the associated
                  value of the entPhysicalIsFRU object is equal to 'false(2)'
                  (e.g., the repeater ports within a repeater module), do not
                  need their own unique serial number. An agent does not have
                  to provide write access for such entities, and may return a
                  zero-length string.

                  If write access is implemented for an instance of
                  entPhysicalSerialNum, and a value is written into the
                  instance, the agent must retain the supplied value in the
                  entPhysicalSerialNum instance (associated with the same
                  physical entity) for as long as that entity remains
                  instantiated. This includes instantiations across all
                  re-initializations/reboots of the network management system,
                  including those resulting in a change of the physical

                  entity's entPhysicalIndex value."
          ::= { entPhysicalEntry 11 }

  entPhysicalMfgName   OBJECT-TYPE
          SYNTAX   SnmpAdminString
          MAX-ACCESS read-only
          STATUS   current
          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
                entPhysicalModelName, entPhysicalFirmwareRev,
                entPhysicalSoftwareRev, and the entPhysicalSerialNum
                objects, are only meaningful amongst entPhysicalEntries with
                the same value of entPhysicalMfgName.

                If the manufacturer name string associated with the physical
                component is unknown to the agent, then this object will
                contain a zero-length string."
        ::= { entPhysicalEntry 12 }

entPhysicalModelName   OBJECT-TYPE
        SYNTAX   SnmpAdminString
        MAX-ACCESS read-only
        STATUS   current
        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 agent, then this object will
                contain a zero-length string."
        ::= { entPhysicalEntry 13 }

entPhysicalAlias   OBJECT-TYPE
        SYNTAX   SnmpAdminString (SIZE (0..32))
        MAX-ACCESS read-write
        STATUS   current
        DESCRIPTION
                "This object is an 'alias' name for the physical entity, as
                specified by a network manager, and provides a non-volatile
                'handle' for the physical entity.

                On the first instantiation of a physical entity, the value
                of entPhysicalAlias associated with that entity is set to
                the zero-length string. However, the agent may set the
                value to a locally unique default value, instead of a
                zero-length string.

                If write access is implemented for an instance of
                entPhysicalAlias, and a value is written into the instance,
                the agent must retain the supplied value in the

            entPhysicalAlias instance (associated with the same physical
            entity) for as long as that entity remains instantiated.
            This includes instantiations across all
            re-initializations/reboots of the network management system,
            including those resulting in a change of the physical
            entity's entPhysicalIndex value."
      ::= { entPhysicalEntry 14 }

entPhysicalAssetID OBJECT-TYPE
      SYNTAX   SnmpAdminString (SIZE (0..32))
      MAX-ACCESS   read-write
      STATUS   current
      DESCRIPTION
            "This object 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.

            On the first instantiation of a physical entity, the value
            of entPhysicalAssetID associated with that entity is set to
            the zero-length string.

            Not every physical component will have an asset tracking
            identifier, or even need one. Physical entities for which
            the associated value of the entPhysicalIsFRU object is equal
            to 'false(2)' (e.g., the repeater ports within a repeater
            module), do not need their own unique asset tracking
            identifier. An agent does not have to provide write access
            for such entities, and may instead return a zero-length
            string.

            If write access is implemented for an instance of
            entPhysicalAssetID, and a value is written into the
            instance, the agent must retain the supplied value in the
            entPhysicalAssetID instance (associated with the same
            physical entity) for as long as that entity remains
            instantiated. This includes instantiations across all
            re-initializations/reboots of the network management system,
            including those resulting in a change of the physical
            entity's entPhysicalIndex value,

            If no asset tracking information is associated with the
            physical component, then this object will contain a
            zero-length string."
      ::= { entPhysicalEntry 15 }

entPhysicalIsFRU OBJECT-TYPE
      SYNTAX   TruthValue
      MAX-ACCESS   read-only

```
    STATUS       current
    DESCRIPTION
            "This object indicates whether or not this physical entity
            is considered a 'field replaceable unit' by the vendor.  If
            this object contains the value 'true(1)' then this
            entPhysicalEntry identifies a field replaceable unit.  For
            all entPhysicalEntries that represent components
            permanently contained within a field replaceable unit, the
            value 'false(2)' should be returned for this object."
    ::= { entPhysicalEntry 16 }

entPhysicalMfgDate  OBJECT-TYPE
    SYNTAX       DateAndTime
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
            "This object contains the date of manufacturing of the
            managed entity.  If the manufacturing date is unknown or not
            supported, the object is not instantiated.  The special
            value '0000000000000000'H may also be returned in this
            case."
    ::= { entPhysicalEntry 17 }

entPhysicalUris OBJECT-TYPE
    SYNTAX       Uri
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
            "This object contains additional identification information
            about the physical entity.  The object contains URIs and,
            therefore, the syntax of this object must conform to RFC
            3986, section 2.

            Multiple URIs may be present and are separated by white
            space characters.  Leading and trailing white space
            characters are ignored.

            If no additional identification information is known
            about the physical entity or supported, the object is not
            instantiated.  A zero length octet string may also be
            returned in this case."
    REFERENCE
            "RFC 3986, Uniform Resource Identifiers (URI): Generic
            Syntax, section 2, August 1998."

    ::= { entPhysicalEntry 18 }

entPhysicalUUID OBJECT-TYPE
```

        SYNTAX   UUIDorZero
        MAX-ACCESS   read-only
        STATUS   current
        DESCRIPTION
                "This object contains additional identification information
                about the physical entity. The object contains a Universal
                Unique Identifier, the syntax of this object must conform to
                RFC 4122, section 4.1.

                A zero length octet string is returned if no UUID
                 information is known."
        REFERENCE
                "RFC 4122, A Universally Unique IDentifier (UUID) URN
                Namespace, section 4.1, July 2005."

        ::= { entPhysicalEntry 19 }

--   The Logical Entity Table
entLogicalTable OBJECT-TYPE
        SYNTAX   SEQUENCE OF EntLogicalEntry
        MAX-ACCESS   not-accessible
        STATUS   current
        DESCRIPTION
                "This table contains one row per logical entity. For agents
                that implement more than one naming scope, at least one
                entry must exist. Agents which instantiate all MIB objects
                within a single naming scope are not required to implement
                this table."
        ::= { entityLogical 1 }

entLogicalEntry   OBJECT-TYPE
        SYNTAX   EntLogicalEntry
        MAX-ACCESS   not-accessible
        STATUS   current
        DESCRIPTION
                "Information about a particular logical entity. Entities
                may be managed by this agent or other SNMP agents (possibly)
                in the same chassis."
        INDEX   { entLogicalIndex }
        ::= { entLogicalTable 1 }

EntLogicalEntry ::= SEQUENCE {
        entLogicalIndex   Integer32,
        entLogicalDescr   SnmpAdminString,
        entLogicalType   AutonomousType,
        entLogicalCommunity   OCTET STRING,

        entLogicalTAddress   TAddress,
        entLogicalTDomain   TDomain,
        entLogicalContextEngineID   SnmpEngineIdOrNone,
        entLogicalContextName   SnmpAdminString
}

entLogicalIndex OBJECT-TYPE
    SYNTAX   Integer32 (1..2147483647)
    MAX-ACCESS not-accessible
    STATUS   current
    DESCRIPTION

            "The value of this object uniquely identifies the logical
            entity. The value should be a small positive integer; index
            values for different logical entities are not necessarily
            contiguous."
    ::= { entLogicalEntry 1 }

entLogicalDescr OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "A textual description of the logical entity. This object
            should contain a string that identifies the manufacturer's
            name for the logical entity, and should be set to a distinct
            value for each version of the logical entity."
    ::= { entLogicalEntry 2 }

entLogicalType OBJECT-TYPE
    SYNTAX   AutonomousType
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "An indication of the type of logical entity. This will
            typically be the OBJECT IDENTIFIER name of the node in the
            SMI's naming hierarchy which represents the major MIB
            module, or the majority of the MIB modules, supported by the
            logical entity. For example:
                a logical entity of a regular host/router -> mib-2
                a logical entity of a 802.1d bridge -> dot1dBridge
                a logical entity of a 802.3 repeater -> snmpDot3RptrMgmt
            If an appropriate node in the SMI's naming hierarchy cannot
            be identified, the value 'mib-2' should be used."
    ::= { entLogicalEntry 3 }

entLogicalCommunity OBJECT-TYPE
    SYNTAX   OCTET STRING (SIZE (0..255))
    MAX-ACCESS read-only
    STATUS   deprecated
    DESCRIPTION
            "An SNMPv1 or SNMPv2C community-string, which can be used to
            access detailed management information for this logical
            entity. The agent should allow read access with this
            community string (to an appropriate subset of all managed
            objects) and may also return a community string based on the
            privileges of the request used to read this object. Note
            that an agent may return a community string with read-only
            privileges, even if this object is accessed with a
            read-write community string. However, the agent must take

            care not to return a community string that allows more
            privileges than the community string used to access this
            object.

            A compliant SNMP agent may wish to conserve naming scopes by
            representing multiple logical entities in a single 'default'
            naming scope. This is possible when the logical entities,
            represented by the same value of entLogicalCommunity, have
            no object instances in common. For example, 'bridge1' and
            'repeater1' may be part of the main naming scope, but at
            least one additional community string is needed to represent
            'bridge2' and 'repeater2'.

            Logical entities 'bridge1' and 'repeater1' would be
            represented by sysOREntries associated with the 'default'
            naming scope.

            For agents not accessible via SNMPv1 or SNMPv2C, the value
            of this object is the empty string. This object may also
            contain an empty string if a community string has not yet
            been assigned by the agent, or if no community string with
            suitable access rights can be returned for a particular SNMP
            request.

            Note that this object is deprecated. Agents which implement
            SNMPv3 access should use the entLogicalContextEngineID and
            entLogicalContextName objects to identify the context
            associated with each logical entity. SNMPv3 agents may
            return a zero-length string for this object, or may continue
            to return a community string (e.g., tri-lingual agent
            support)."

    ::= { entLogicalEntry 4 }

entLogicalTAddress OBJECT-TYPE
    SYNTAX   TAddress
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "The transport service address by which the logical entity
            receives network management traffic, formatted according to
            the corresponding value of entLogicalTDomain.

            For snmpUDPDomain, a TAddress is 6 octets long: the initial
            4 octets contain the IP-address in network-byte order and
            the last 2 contain the UDP port in network-byte order.
            Consult 'Transport Mappings for the Simple Network
            Management Protocol' (STD 62, RFC 3417 [RFC3417]) for
            further information on snmpUDPDomain."

    ::= { entLogicalEntry 5 }

entLogicalTDomain OBJECT-TYPE
    SYNTAX   TDomain
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "Indicates the kind of transport service by which the
            logical entity receives network management traffic.
            Possible values for this object are presently found in the
            Transport Mappings for Simple Network Management Protocol'
            (STD 62, RFC 3417 [RFC3417])."
    ::= { entLogicalEntry 6 }

entLogicalContextEngineID   OBJECT-TYPE
    SYNTAX   SnmpEngineIdOrNone
    MAX-ACCESS read-only
    STATUS   current
    DESCRIPTION
            "The authoritative contextEngineID that can be used to send
            an SNMP message concerning information held by this logical
            entity, to the address specified by the associated
            'entLogicalTAddress/entLogicalTDomain' pair.

            This object, together with the associated
            entLogicalContextName object, defines the context associated
            with a particular logical entity, and allows access to SNMP
            engines identified by a contextEngineId and contextName

          pair.

          If no value has been configured by the agent, a zero-length
          string is returned, or the agent may choose not to
          instantiate this object at all."
    ::= { entLogicalEntry 7 }

entLogicalContextName   OBJECT-TYPE
    SYNTAX   SnmpAdminString
    MAX-ACCESS   read-only
    STATUS   current
    DESCRIPTION
          "The contextName that can be used to send an SNMP message
          concerning information held by this logical entity, to the
          address specified by the associated
          'entLogicalTAddress/entLogicalTDomain' pair.

          This object, together with the associated
          entLogicalContextEngineID object, defines the context
          associated with a particular logical entity, and allows

          access to SNMP engines identified by a contextEngineId and
          contextName pair.

          If no value has been configured by the agent, a zero-length
          string is returned, or the agent may choose not to
          instantiate this object at all."
    ::= { entLogicalEntry 8 }

entLPMappingTable OBJECT-TYPE
    SYNTAX   SEQUENCE OF EntLPMappingEntry
    MAX-ACCESS   not-accessible
    STATUS   current
    DESCRIPTION
          "This table contains zero or more rows of logical entity to
          physical equipment associations. For each logical entity
          known by this agent, there are zero or more mappings to the
          physical resources, which are used to realize that logical
          entity.

          An agent should limit the number and nature of entries in
          this table such that only meaningful and non-redundant
          information is returned. For example, in a system that
          contains a single power supply, mappings between logical
          entities and the power supply are not useful and should not
          be included.

            Also, only the most appropriate physical component, which is
            closest to the root of a particular containment tree, should
            be identified in an entLPMapping entry.

            For example, suppose a bridge is realized on a particular
            module, and all ports on that module are ports on this
            bridge. A mapping between the bridge and the module would
            be useful, but additional mappings between the bridge and
            each of the ports on that module would be redundant (because
            the entPhysicalContainedIn hierarchy can provide the same
            information). On the other hand, if more than one bridge
            were utilizing ports on this module, then mappings between
            each bridge and the ports it used would be appropriate.

            Also, in the case of a single backplane repeater, a mapping
            for the backplane to the single repeater entity is not
            necessary."
    ::= { entityMapping 1 }

entLPMappingEntry   OBJECT-TYPE
    SYNTAX   EntLPMappingEntry
    MAX-ACCESS   not-accessible

    STATUS   current
    DESCRIPTION
            "Information about a particular logical entity to physical
            equipment association. Note that the nature of the
            association is not specifically identified in this entry.
            It is expected that sufficient information exists in the
            MIBs used to manage a particular logical entity to infer how
            physical component information is utilized."
    INDEX   { entLogicalIndex, entLPPhysicalIndex }
    ::= { entLPMappingTable 1 }

EntLPMappingEntry ::= SEQUENCE {
     entLPPhysicalIndex   PhysicalIndex
}

entLPPhysicalIndex OBJECT-TYPE
    SYNTAX   PhysicalIndex
    MAX-ACCESS   read-only
    STATUS   current
    DESCRIPTION
            "The value of this object identifies the index value of a
            particular entPhysicalEntry associated with the indicated
            entLogicalEntity."

    ::= { entLPMappingEntry 1 }

-- logical entity/component to alias table
entAliasMappingTable OBJECT-TYPE
    SYNTAX   SEQUENCE OF EntAliasMappingEntry
    MAX-ACCESS   not-accessible
    STATUS   current
    DESCRIPTION
            "This table contains zero or more rows, representing
            mappings of logical entity and physical component to
            external MIB identifiers. Each physical port in the system
            may be associated with a mapping to an external identifier,
            which itself is associated with a particular logical
            entity's naming scope. A 'wildcard' mechanism is provided
            to indicate that an identifier is associated with more than
            one logical entity."
    ::= { entityMapping 2 }

entAliasMappingEntry   OBJECT-TYPE
    SYNTAX   EntAliasMappingEntry
    MAX-ACCESS   not-accessible
    STATUS   current
    DESCRIPTION
            "Information about a particular physical equipment, logical

            entity to external identifier binding. Each logical
            entity/physical component pair may be associated with one
            alias mapping. The logical entity index may also be used as
            a 'wildcard' (refer to the entAliasLogicalIndexOrZero object
            DESCRIPTION clause for details.)

            Note that only entPhysicalIndex values that represent
            physical ports (i.e., associated entPhysicalClass value is
            'port(10)') are permitted to exist in this table."
    INDEX { entPhysicalIndex, entAliasLogicalIndexOrZero }
    ::= { entAliasMappingTable 1 }

EntAliasMappingEntry ::= SEQUENCE {
      entAliasLogicalIndexOrZero   Integer32,
      entAliasMappingIdentifier   RowPointer
}

entAliasLogicalIndexOrZero OBJECT-TYPE
    SYNTAX   Integer32 (0..2147483647)
    MAX-ACCESS   not-accessible

```
    STATUS       current
    DESCRIPTION
            "The value of this object identifies the logical entity
            that defines the naming scope for the associated instance
            of the 'entAliasMappingIdentifier' object.

            If this object has a non-zero value, then it identifies the
            logical entity named by the same value of entLogicalIndex.

            If this object has a value of zero, then the mapping between
            the physical component and the alias identifier for this
            entAliasMapping entry is associated with all unspecified
            logical entities.  That is, a value of zero (the default
            mapping) identifies any logical entity that does not have
            an explicit entry in this table for a particular
            entPhysicalIndex/entAliasMappingIdentifier pair.

            For example, to indicate that a particular interface (e.g.,
            physical component 33) is identified by the same value of
            ifIndex for all logical entities, the following instance
            might exist:

                    entAliasMappingIdentifier.33.0 = ifIndex.5

            In the event an entPhysicalEntry is associated differently
            for some logical entities, additional entAliasMapping
            entries may exist, e.g.:



                    entAliasMappingIdentifier.33.0 = ifIndex.6
                    entAliasMappingIdentifier.33.4 =  ifIndex.1
                    entAliasMappingIdentifier.33.5 =  ifIndex.1
                    entAliasMappingIdentifier.33.10 = ifIndex.12

           Note that entries with non-zero entAliasLogicalIndexOrZero
           index values have precedence over zero-indexed entries.  In
           this example, all logical entities except 4, 5, and 10,
           associate physical entity 33 with ifIndex.6."
    ::= { entAliasMappingEntry 1 }

entAliasMappingIdentifier OBJECT-TYPE
    SYNTAX       RowPointer
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
            "The value of this object identifies a particular conceptual
```

                row associated with the indicated entPhysicalIndex and
                entLogicalIndex pair.

                Because only physical ports are modeled in this table, only
                entries that represent interfaces or ports are allowed. If
                an ifEntry exists on behalf of a particular physical port,
                then this object should identify the associated 'ifEntry'.
                For repeater ports, the appropriate row in the
                'rptrPortGroupTable' should be identified instead.

                For example, suppose a physical port was represented by
                entPhysicalEntry.3, entLogicalEntry.15 existed for a
                repeater, and entLogicalEntry.22 existed for a bridge. Then
                there might be two related instances of
                entAliasMappingIdentifier:
                    entAliasMappingIdentifier.3.15 == rptrPortGroupIndex.5.2
                    entAliasMappingIdentifier.3.22 == ifIndex.17
                It is possible that other mappings (besides interfaces and
                repeater ports) may be defined in the future, as required.

                Bridge ports are identified by examining the Bridge MIB and
                appropriate ifEntries associated with each 'dot1dBasePort',
                and are thus not represented in this table."
        ::= { entAliasMappingEntry 2 }

-- physical mapping table
entPhysicalContainsTable OBJECT-TYPE
    SYNTAX   SEQUENCE OF EntPhysicalContainsEntry
    MAX-ACCESS not-accessible
    STATUS   current

    DESCRIPTION
            "A table that exposes the container/'containee'
            relationships between physical entities. This table
            provides all the information found by constructing the
            virtual containment tree for a given entPhysicalTable, but
            in a more direct format.

            In the event a physical entity is contained by more than one
            other physical entity (e.g., double-wide modules), this
            table should include these additional mappings, which cannot
            be represented in the entPhysicalTable virtual containment
            tree."
        ::= { entityMapping 3 }

```
entPhysicalContainsEntry OBJECT-TYPE
    SYNTAX       EntPhysicalContainsEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
            "A single container/'containee' relationship."
    INDEX        { entPhysicalIndex, entPhysicalChildIndex }
    ::= { entPhysicalContainsTable 1 }

EntPhysicalContainsEntry ::= SEQUENCE {
      entPhysicalChildIndex     PhysicalIndex
}

entPhysicalChildIndex OBJECT-TYPE
    SYNTAX       PhysicalIndex
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
            "The value of entPhysicalIndex for the contained physical
            entity."
    ::= { entPhysicalContainsEntry 1 }

-- last change time stamp for the whole MIB
entLastChangeTime OBJECT-TYPE
    SYNTAX       TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
            "The value of sysUpTime at the time a conceptual row is
            created, modified, or deleted in any of these tables:
                    - entPhysicalTable
                    - entLogicalTable
                    - entLPMappingTable
                    - entAliasMappingTable



                    - entPhysicalContainsTable
            "
    ::= { entityGeneral 1 }


-- Entity MIB Trap Definitions
entityMIBTraps      OBJECT IDENTIFIER ::= { entityMIB 2 }
entityMIBTrapPrefix OBJECT IDENTIFIER ::= { entityMIBTraps 0 }

entConfigChange NOTIFICATION-TYPE
    STATUS             current
```

        DESCRIPTION
                "An entConfigChange notification is generated when the value
                of entLastChangeTime changes. It can be utilized by an NMS
                to trigger logical/physical entity table maintenance polls.

                An agent should not generate more than one entConfigChange
                'notification-event' in a given time interval (five seconds
                is the suggested default). A 'notification-event' is the
                transmission of a single trap or inform PDU to a list of
                notification destinations.

                If additional configuration changes occur within the
                throttling period, then notification-events for these
                changes should be suppressed by the agent until the current
                throttling period expires. At the end of a throttling
                period, one notification-event should be generated if any
                configuration changes occurred since the start of the
                throttling period. In such a case, another throttling
                period is started right away.

                An NMS should periodically check the value of
                entLastChangeTime to detect any missed entConfigChange
                notification-events, e.g., due to throttling or transmission
                loss."
        ::= { entityMIBTrapPrefix 1 }

-- conformance information
entityConformance OBJECT IDENTIFIER ::= { entityMIB 3 }

entityCompliances OBJECT IDENTIFIER ::= { entityConformance 1 }
entityGroups   OBJECT IDENTIFIER ::= { entityConformance 2 }

-- compliance statements
entityCompliance MODULE-COMPLIANCE
    STATUS deprecated

    DESCRIPTION
            "The compliance statement for SNMP entities that implement
            version 1 of the Entity MIB."
    MODULE -- this module
        MANDATORY-GROUPS {
                        entityPhysicalGroup,
                        entityLogicalGroup,
                        entityMappingGroup,

                                 entityGeneralGroup,
                                 entityNotificationsGroup
            }
        ::= { entityCompliances 1 }

    entity2Compliance MODULE-COMPLIANCE
        STATUS deprecated
        DESCRIPTION
                "The compliance statement for SNMP entities that implement
                version 2 of the Entity MIB."
        MODULE -- this module
            MANDATORY-GROUPS {
                                entityPhysicalGroup,
                                entityPhysical2Group,
                                entityGeneralGroup,
                                entityNotificationsGroup
            }
            GROUP entityLogical2Group
            DESCRIPTION
                "Implementation of this group is not mandatory for agents
                that model all MIB object instances within a single naming
                scope."

            GROUP entityMappingGroup
            DESCRIPTION
                "Implementation of the entPhysicalContainsTable is mandatory
                for all agents. Implementation of the entLPMappingTable and
                entAliasMappingTables are not mandatory for agents that
                model all MIB object instances within a single naming scope.

                Note that the entAliasMappingTable may be useful for all
                agents; however, implementation of the entityLogicalGroup or
                entityLogical2Group is required to support this table."

            OBJECT entPhysicalSerialNum
            MIN-ACCESS   not-accessible
            DESCRIPTION
                "Read and write access is not required for agents that
                cannot identify serial number information for physical
                entities, and/or cannot provide non-volatile storage for

                NMS-assigned serial numbers.

                Write access is not required for agents that can identify
                serial number information for physical entities, but cannot
                provide non-volatile storage for NMS-assigned serial

```
            numbers.

            Write access is not required for physical entities for which
            the associated value of the entPhysicalIsFRU object is equal
            to 'false(2)'."

        OBJECT entPhysicalAlias
        MIN-ACCESS   read-only
        DESCRIPTION
            "Write access is required only if the associated
            entPhysicalClass value is equal to 'chassis(3)'."

        OBJECT entPhysicalAssetID
        MIN-ACCESS   not-accessible
        DESCRIPTION
            "Read and write access is not required for agents that
            cannot provide non-volatile storage for NMS-assigned asset
            identifiers.

            Write access is not required for physical entities for which
            the associated value of the entPhysicalIsFRU object is equal
            to 'false(2)'."

        OBJECT entPhysicalClass
        SYNTAX INTEGER {
            other(1),
            unknown(2),
            chassis(3),
            backplane(4),
            container(5),
            powerSupply(6),
            fan(7),
            sensor(8),
            module(9),
            port(10),
            stack(11)
        }
        DESCRIPTION
            "Implementation of the 'cpu(12)' enumeration is not
            required."

    ::= { entityCompliances 2 }



entity3Compliance MODULE-COMPLIANCE
    STATUS   current
```

```
    DESCRIPTION
        "The compliance statement for SNMP entities that implement
        version 3 and 4 (full compliance) of the Entity MIB."
MODULE  -- this module
    MANDATORY-GROUPS {
                       entityPhysicalGroup,
                       entityPhysical2Group,
                       entityGeneralGroup,
                       entityNotificationsGroup
    }
    GROUP entityLogical2Group
    DESCRIPTION
        "Implementation of this group is not mandatory for agents
        that model all MIB object instances within a single naming
        scope."

    GROUP entityMappingGroup
    DESCRIPTION
        "Implementation of the entPhysicalContainsTable is mandatory
        for all agents.  Implementation of the entLPMappingTable and
        entAliasMappingTables are not mandatory for agents that
        model all MIB object instances within a single naming scope.

        Note that the entAliasMappingTable may be useful for all
        agents; however, implementation of the entityLogicalGroup or
        entityLogical2Group is required to support this table."

    OBJECT entPhysicalSerialNum
    MIN-ACCESS   not-accessible
    DESCRIPTION
        "Read and write access is not required for agents that
        cannot identify serial number information for physical
        entities, and/or cannot provide non-volatile storage for
        NMS-assigned serial numbers.

        Write access is not required for agents that can identify
        serial number information for physical entities, but cannot
        provide non-volatile storage for NMS-assigned serial
        numbers.

        Write access is not required for physical entities for
        which the associated value of the entPhysicalIsFRU object
        is equal to 'false(2)'."

    OBJECT entPhysicalAlias
```

```
        MIN-ACCESS   read-only
        DESCRIPTION
            "Write access is required only if the associated
            entPhysicalClass value is equal to 'chassis(3)'."

        OBJECT entPhysicalAssetID
        MIN-ACCESS   not-accessible
        DESCRIPTION
            "Read and write access is not required for agents that
            cannot provide non-volatile storage for NMS-assigned asset
            identifiers.

            Write access is not required for physical entities for which
            the associated value of entPhysicalIsFRU is equal to
            'false(2)'."
    ::= { entityCompliances 3 }


entity4CRCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for SNMP entities that implement
            version 4 of the Entity MIB on devices with constrained
            resources."
    MODULE  -- this module
        MANDATORY-GROUPS {
                            entityPhysicalCRGroup
        }


    ::= { entityCompliances 4 }



-- MIB groupings
entityPhysicalGroup     OBJECT-GROUP
    OBJECTS {
                entPhysicalDescr,
                entPhysicalVendorType,
                entPhysicalContainedIn,
                entPhysicalClass,
                entPhysicalParentRelPos,
                entPhysicalName
            }
    STATUS  current
    DESCRIPTION
            "The collection of objects used to represent physical
            system components, for which a single agent provides
```

```
            management information."
    ::= { entityGroups 1 }

entityLogicalGroup    OBJECT-GROUP
    OBJECTS {
            entLogicalDescr,
            entLogicalType,
            entLogicalCommunity,
            entLogicalTAddress,
            entLogicalTDomain
            }
    STATUS  deprecated
    DESCRIPTION
            "The collection of objects used to represent the list of
            logical entities, for which a single agent provides
            management information."



    ::= { entityGroups 2 }

entityMappingGroup    OBJECT-GROUP
    OBJECTS {
            entLPPhysicalIndex,
            entAliasMappingIdentifier,
            entPhysicalChildIndex
            }
    STATUS  current
    DESCRIPTION
            "The collection of objects used to represent the
            associations between multiple logical entities, physical
            components, interfaces, and port identifiers, for which a
            single agent provides management information."
    ::= { entityGroups 3 }

entityGeneralGroup    OBJECT-GROUP
    OBJECTS {
            entLastChangeTime
            }
    STATUS  current
    DESCRIPTION
            "The collection of objects used to represent general entity
            information, for which a single agent provides management
            information."
    ::= { entityGroups 4 }

entityNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { entConfigChange }
```

```
    STATUS          current
    DESCRIPTION
            "The collection of notifications used to indicate Entity MIB
            data consistency and general status information."
    ::= { entityGroups 5 }

entityPhysical2Group    OBJECT-GROUP
    OBJECTS {
            entPhysicalHardwareRev,
            entPhysicalFirmwareRev,
            entPhysicalSoftwareRev,
            entPhysicalSerialNum,
            entPhysicalMfgName,
            entPhysicalModelName,
            entPhysicalAlias,
            entPhysicalAssetID,
            entPhysicalIsFRU,
            entPhysicalMfgDate,
            entPhysicalUris
            }
    STATUS  current



    DESCRIPTION
            "The collection of objects used to represent physical
            system components, for which a single agent provides
            management information.  This group augments the objects
            contained in the entityPhysicalGroup."
    ::= { entityGroups 6 }

entityLogical2Group    OBJECT-GROUP
    OBJECTS {
            entLogicalDescr,
            entLogicalType,
            entLogicalTAddress,
            entLogicalTDomain,
            entLogicalContextEngineID,
            entLogicalContextName
            }
    STATUS  current
    DESCRIPTION
            "The collection of objects used to represent the
            list of logical entities, for which a single SNMP entity
            provides management information."
    ::= { entityGroups 7 }

entityPhysicalCRGroup    OBJECT-GROUP
```

```
    OBJECTS {
            entPhysicalClass,
            entPhysicalName,
            entPhysicalUUID
          }
    STATUS   current
    DESCRIPTION
            "The collection of objects used to represent physical
            system components for constrained resourced devices,
            for which a single agent provides
            management information."
    ::= { entityGroups 8 }


END

3.2.  IANA-ENTITY-MIB

   IANA-ENTITY-MIB DEFINITIONS ::= BEGIN

       IMPORTS
         MODULE-IDENTITY, mib-2
           FROM SNMPv2-SMI
         TEXTUAL-CONVENTION
           FROM SNMPv2-TC
         ;

       ianaEntityMIB MODULE-IDENTITY
         LAST-UPDATED "201206100000Z"  -- June 10, 2011
         ORGANIZATION "IANA"
         CONTACT-INFO "        Internet Assigned Numbers Authority

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

                         Tel: +1-310-823-9358
                       EMail: iana&iana.org"

         DESCRIPTION
           "This MIB module

           Copyright (C) The IETF Trust (2012).
           The initial version of this MIB module was published in
           RFC yyyy; for full legal notices see the RFC itself.
           Supplementary information may be available at:
           http://www.ietf.org/copyrights/ianamib.html"
```

```
        REVISION      "201206100000Z"  -- June 10, 2012
        DESCRIPTION   "Initial version of this MIB as published in
                       RFC yyyy."

        ::= { mib-2 xxx }

    -- RFC Editor, please replace xxx with the IANA allocation for this
    -- MIB module and yyyy with the number of the approved RFC

    -- Textual Conventions


   IANAPhysicalClass ::= TEXTUAL-CONVENTION
       STATUS            current
       DESCRIPTION
```

"An enumerated value which provides an indication of the
general hardware type of a particular physical entity.
There are no restrictions as to the number of
entPhysicalEntries of each entPhysicalClass, which must be
instantiated by an agent.

The enumeration 'other' is applicable if the physical entity
class is known, but does not match any of the supported
values.

The enumeration 'unknown' is applicable if the physical
entity class is unknown to the agent.

The enumeration 'chassis' 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; and a chassis may only
be contained within a stack.

The enumeration 'backplane' 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 agent may model a
backplane as a single physical entity, which is actually
implemented as multiple discrete physical components (within
a chassis or stack).

The enumeration 'container' 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.

                    The enumeration 'powerSupply' is applicable if the physical
                    entity class is a power-supplying component.

                    The enumeration 'fan' is applicable if the physical entity
                    class is a fan or other heat-reduction component.

                    The enumeration 'sensor' is applicable if the physical
                    entity class is some sort of sensor, such as a temperature
                    sensor within a router chassis.

                    The enumeration 'module' is applicable if the physical
                    entity class is some sort of self-contained sub-system. If
                    the enumeration 'module' 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).

                    The enumeration 'port' is applicable if the physical entity
                    class is some sort of networking port, capable of receiving
                    and/or transmitting networking traffic.

                    The enumeration 'stack' 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 may in
                    fact be comprised of 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.

                    The enumeration 'cpu' is applicable if the physical entity
                    class is some sort of central processing unit.

                    The enumeration ?energyObject? is applicable if the
                    physical entity is some sort of a 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.

                    The enumeration ?battery? is applicable of the physical
                    entity class is some sort of an energy battery device. "

            SYNTAX   INTEGER {

```
          other(1),
          unknown(2),
          chassis(3),
          backplane(4),
          container(5),      -- e.g., chassis slot or daughter-card holder
          powerSupply(6),
          fan(7),
          sensor(8),
          module(9),         -- e.g., plug-in card or daughter-card
          port(10),
          stack(11),         -- e.g., stack of multiple chassis entities
          cpu(12),
          energyObject(13),
          battery (14)
       }


   END

3.3. UUID-TC-MIB

   UUID-TC-MIB DEFINITIONS ::= BEGIN

       IMPORTS
         MODULE-IDENTITY, mib-2
           FROM SNMPv2-SMI
         TEXTUAL-CONVENTION
           FROM SNMPv2-TC ;

       uuidTCMIB MODULE-IDENTITY
         LAST-UPDATED "201210090000Z"  -- October 9, 2011
         ORGANIZATION "IETF Energy Management Working Group"
         CONTACT-INFO "
                       WG E-mail: eman@ietf.org
                       Mailing list subscription info:
                         http://www.ietf.org/mailman/listinfo/eman

                       Dan Romascanu
                       AVAYA
                       Park Atidim, Bldg. #3
                       Tel Aviv, 61581
                       Israel
                       Phone: +972-3-6458414
                       Email: dromasca@avaya.com

                       Juergen Quittek
                       NEC Europe Ltd.
                       Network Research Division
```

                          Kurfuersten-Anlage 36
                          Heidelberg 69115
                          DE
                          Phone: +49 6221 4342-115
                          Email: quittek@neclab.eu

                          Mouli Chandramouli
                          Cisco Systems, Inc.
                          Sarjapur Outer Ring Road
                          Bangalore 560103
                          IN
                          Phone: +91 80 4429 2409
                          Email: moulchan@cisco.com"

          DESCRIPTION
            "This MIB module

            Copyright (C) The IETF Trust (2012).
            The initial version of this MIB module was published in
            RFC yyyy; for full legal notices see the RFC itself.
            Supplementary information may be available at:
            http://www.ietf.org/copyrights/ianamib.html"

          REVISION   "201210090000Z" -- October 9, 2012
          DESCRIPTION "Initial version of this MIB as published in
                        RFC yyyy."

          ::= { mib-2 xxx }

       -- RFC Editor, please replace xxx with the IANA allocation for this
       -- MIB module and yyyy with the number of the approved RFC

       -- Textual Conventions

    UUID ::= TEXTUAL-CONVENTION
        DISPLAY-HINT   "4x-2x-2x-1x1x-6x"
        STATUS   current
        DESCRIPTION
                "Universal Unique Identifier information. The syntax must
                conform to RFC 4122, section 4.1."

           SYNTAX   OCTET STRING (SIZE (16))

    UUIDorZero ::= TEXTUAL-CONVENTION
        DISPLAY-HINT   "4x-2x-2x-1x1x-6x"
        STATUS   current
        DESCRIPTION
                " Universal Unique Identifier information. The syntax

                  must conform to RFC 4122, section 4.1.

                  The semantics of the value zero-length OCTET STRING are
                  object-specific and must therefore be defined as part of
                  the description of any object that uses this syntax."

              SYNTAX   OCTET STRING (SIZE (0|16))

       END

4. Usage Examples

The following sections iterate the instance values for two example
networking devices. These examples are kept simple to make them
more understandable. Auxiliary components such as fans, sensors,
empty slots, and sub-modules are not shown, but might be modeled in
real implementations.

4.1. Router/Bridge

   The first example is a router containing two slots. Each slot contains
   a 3 port router/bridge module. Each port is represented in the ifTable.
   There are two logical instances of OSPF running and two logical
   bridges:

    Physical entities -- entPhysicalTable:
       1 Field-replaceable physical chassis:
         entPhysicalDescr.1 ==   'Acme Chassis Model 100'
         entPhysicalVendorType.1 ==   acmeProducts.chassisTypes.1
         entPhysicalContainedIn.1 ==   0
         entPhysicalClass.1 ==   chassis(3)
         entPhysicalParentRelPos.1 ==   0
         entPhysicalName.1 ==   '100-A'
         entPhysicalHardwareRev.1 ==   'A(1.00.02)'
         entPhysicalSoftwareRev.1 ==   ''
         entPhysicalFirmwareRev.1 ==   ''
         entPhysicalSerialNum.1 ==   'C100076544'
         entPhysicalMfgName.1 ==   'Acme'
         entPhysicalModelName.1 ==   '100'
         entPhysicalAlias.1 ==   'cl-SJ17-3-006:rack1:rtr-U3'
         entPhysicalAssetID.1 ==   '0007372293'
         entPhysicalIsFRU.1 ==   true(1)
         entPhysicalMfgDate.1 ==   '2002-5-26,13:30:30.0,-4:0'
         entPhysicalUris.1 ==   'URN:CLEI:CNME120ARA'
       2 slots within the chassis:
         entPhysicalDescr.2 ==   'Acme Chassis Slot Type AA'
         entPhysicalVendorType.2 ==   acmeProducts.slotTypes.1
         entPhysicalContainedIn.2 ==   1

```
        entPhysicalClass.2 ==             container(5)
        entPhysicalParentRelPos.2 ==      1
        entPhysicalName.2 ==              'S1'
        entPhysicalHardwareRev.2 ==       'B(1.00.01)'
        entPhysicalSoftwareRev.2 ==       ''
        entPhysicalFirmwareRev.2 ==       ''
        entPhysicalSerialNum.2 ==         ''
        entPhysicalMfgName.2 ==           'Acme'
        entPhysicalModelName.2 ==         'AA'
        entPhysicalAlias.2 ==             ''
        entPhysicalAssetID.2 ==           ''
        entPhysicalIsFRU.2 ==             false(2)
        entPhysicalMfgDate.2 ==           '2002-7-26,12:22:12.0,-4:0'
        entPhysicalUris.2 ==              'URN:CLEI:CNME123ARA'

        entPhysicalDescr.3 ==             'Acme Chassis Slot Type AA'
        entPhysicalVendorType.3 =         acmeProducts.slotTypes.1
        entPhysicalContainedIn.3 ==       1
        entPhysicalClass.3 ==             container(5)
        entPhysicalParentRelPos.3 ==      2
        entPhysicalName.3 ==              'S2'
        entPhysicalHardwareRev.3 ==       '1.00.07'
        entPhysicalSoftwareRev.3 ==       ''
        entPhysicalFirmwareRev.3 ==       ''
        entPhysicalSerialNum.3 ==         ''
        entPhysicalMfgName.3 ==           'Acme'
        entPhysicalModelName.3 ==         'AA'
        entPhysicalAlias.3 ==             ''
        entPhysicalAssetID.3 ==           ''
        entPhysicalIsFRU.3 ==             false(2)
        entPhysicalMfgDate.3 ==           '2002-7-26,12:12:12.0,-4:0'
        entPhysicalUris.3 ==              'URN:CLEI:CNME123ARA'

     2 Field-replaceable modules:
     Slot 1 contains a module with 3 ports:
        entPhysicalDescr.4 ==             'Acme Router-100'
        entPhysicalVendorType.4  ==       acmeProducts.moduleTypes.14
        entPhysicalContainedIn.4 ==       2
        entPhysicalClass.4 ==             module(9)
        entPhysicalParentRelPos.4 ==      1
        entPhysicalName.4 ==              'M1'
        entPhysicalHardwareRev.4 ==       '1.00.07'
        entPhysicalSoftwareRev.4 ==       '1.4.1'
        entPhysicalFirmwareRev.4 ==       'A(1.1)'
        entPhysicalSerialNum.4 ==         'C100087363'
        entPhysicalMfgName.4 ==           'Acme'
        entPhysicalModelName.4 ==         'R100-FE'
        entPhysicalAlias.4 ==             'rtr-U3:m1:SJ17-3-eng'
```

```
        entPhysicalAssetID.4 ==           '0007372462'
        entPhysicalIsFRU.4 ==             true(1)
        entPhysicalMfgDate.4 ==           '2003-7-18,13:30:30.0,-4:0'
        entPhysicalUris.4 ==              'URN:CLEI:CNRU123CAA'

        entPhysicalDescr.5 ==             'Acme Ethernet-100 Port'
        entPhysicalVendorType.5  ==       acmeProducts.portTypes.2
        entPhysicalContainedIn.5 ==       4
        entPhysicalClass.5 ==             port(10)
        entPhysicalParentRelPos.5 ==      1
        entPhysicalName.5 ==              'P1'
        entPhysicalHardwareRev.5 ==       'G(1.02)'
        entPhysicalSoftwareRev.5 ==       ''
        entPhysicalFirmwareRev.5 ==       '1.1'
        entPhysicalSerialNum.5 ==         ''
        entPhysicalMfgName.5 ==           'Acme'
        entPhysicalModelName.5 ==         'FE-100'
        entPhysicalAlias.5 ==             ''
        entPhysicalAssetID.5 ==           ''
        entPhysicalIsFRU.5 ==             false(2)
        entPhysicalMfgDate.5 ==           '2003-7-18,14:20:22.0,-4:0'
        entPhysicalUris.5 ==              'URN:CLEI:CNMES23ARA'

        entPhysicalDescr.6 ==             'Acme Ethernet-100 Port'
        entPhysicalVendorType.6  ==       acmeProducts.portTypes.2
        entPhysicalContainedIn.6 ==       4
        entPhysicalClass.6 ==             port(10)
        entPhysicalParentRelPos.6 ==      2
        entPhysicalName.6 ==              'P2'
        entPhysicalHardwareRev.6 ==       'G(1.02)'
        entPhysicalSoftwareRev.6 ==       ''
        entPhysicalFirmwareRev.6 ==       '1.1'
        entPhysicalSerialNum.6 ==         ''
        entPhysicalMfgName.6 ==           'Acme'
        entPhysicalModelName.6 ==         'FE-100'
        entPhysicalAlias.6 ==             ''
        entPhysicalAssetID.6 ==           ''
        entPhysicalIsFRU.6 ==             false(2)
        entPhysicalMfgDate.6 ==           '2003-7-19,10:15:15.0,-4:0'
        entPhysicalUris.6 ==              'URN:CLEI:CNMES23ARA'

        entPhysicalDescr.7 ==             'Acme Router-100 FDDI-Port'
        entPhysicalVendorType.7  ==       acmeProducts.portTypes.3
        entPhysicalContainedIn.7 ==       4
        entPhysicalClass.7 ==             port(10)
        entPhysicalParentRelPos.7 ==      3
        entPhysicalName.7 ==              'P3'
        entPhysicalHardwareRev.7 ==       'B(1.03)'
```

```
        entPhysicalSoftwareRev.7 ==         '2.5.1'
        entPhysicalFirmwareRev.7 ==         '2.5F'
        entPhysicalSerialNum.7 ==           ''
        entPhysicalMfgName.7 ==             'Acme'
        entPhysicalModelName.7 ==           'FDDI-100'
        entPhysicalAlias.7 ==               ''
        entPhysicalAssetID.7 ==             ''
        entPhysicalIsFRU.7 ==               false(2)


    Slot 2 contains another 3-port module:
        entPhysicalDescr.8 ==               'Acme Router-100 Comm Module'
        entPhysicalVendorType.8  ==         acmeProducts.moduleTypes.15
        entPhysicalContainedIn.8 ==         3
        entPhysicalClass.8 ==               module(9)
        entPhysicalParentRelPos.8 ==        1
        entPhysicalName.8 ==                'M2'
        entPhysicalHardwareRev.8 ==         '2.01.00'
        entPhysicalSoftwareRev.8 ==         '3.0.7'
        entPhysicalFirmwareRev.8 ==         'A(1.2)'
        entPhysicalSerialNum.8 ==           'C100098732'
        entPhysicalMfgName.8 ==             'Acme'
        entPhysicalModelName.8 ==           'C100'
        entPhysicalAlias.8 ==               'rtr-U3:m2:SJ17-2-eng'
        entPhysicalAssetID.8 ==             '0007373982'
        entPhysicalIsFRU.8 ==               true(1)
        entPhysicalMfgDate.8 ==             '2002-5-26,13:30:15.0,-4:0'
        entPhysicalUris.8 ==                'URN:CLEI:CNRT321MAA'


        entPhysicalDescr.9 ==               'Acme Fddi-100 Port'
        entPhysicalVendorType.9 ==          acmeProducts.portTypes.5
        entPhysicalContainedIn.9 ==         8
        entPhysicalClass.9 ==               port(10)
        entPhysicalParentRelPos.9 ==        1
        entPhysicalName.9 ==                'FDDI Primary'
        entPhysicalHardwareRev.9 ==         'CC(1.07)'
        entPhysicalSoftwareRev.9 ==         '2.0.34'
        entPhysicalFirmwareRev.9 ==         '1.1'
        entPhysicalSerialNum.9 ==           ''
        entPhysicalMfgName.9 ==             'Acme'
        entPhysicalModelName.9 ==           'FDDI-100'
        entPhysicalAlias.9 ==               ''
        entPhysicalAssetID.9 ==             ''
        entPhysicalIsFRU.9 ==               false(2)


        entPhysicalDescr.10 ==              'Acme Ethernet-100 Port'
        entPhysicalVendorType.10 ==         acmeProducts.portTypes.2
        entPhysicalContainedIn.10 ==        8
        entPhysicalClass.10 ==              port(10)
```

```
        entPhysicalParentRelPos.10 ==   2
        entPhysicalName.10 ==           'Ethernet A'
        entPhysicalHardwareRev.10 ==    'G(1.04)'
        entPhysicalSoftwareRev.10 ==    ''
        entPhysicalFirmwareRev.10 ==    '1.3'
        entPhysicalSerialNum.10 ==      ''
        entPhysicalMfgName.10 ==        'Acme'
        entPhysicalModelName.10 ==      'FE-100'
        entPhysicalAlias.10 ==          ''
        entPhysicalAssetID.10 ==        ''
        entPhysicalIsFRU.10 ==          false(2)
        entPhysicalMfgDate.10 ==        '2002-7-26,13:30:15.0,-4:0'
        entPhysicalUris.10 ==           'URN:CLEI:CNMES23ARA'

        entPhysicalDescr.11 ==          'Acme Ethernet-100 Port'
        entPhysicalVendorType.11 ==     acmeProducts.portTypes.2
        entPhysicalContainedIn.11 ==    8
        entPhysicalClass.11 ==          port(10)
        entPhysicalParentRelPos.11 ==   3
        entPhysicalName.11 ==           'Ethernet B'
        entPhysicalHardwareRev.11 ==    'G(1.04)'
        entPhysicalSoftwareRev.11 ==    ''
        entPhysicalFirmwareRev.11 ==    '1.3'
        entPhysicalSerialNum.11 ==      ''
        entPhysicalMfgName.11 ==        'Acme'
        entPhysicalModelName.11 ==      'FE-100'
        entPhysicalAlias.11 ==          ''
        entPhysicalAssetID.11 ==        ''
        entPhysicalIsFRU.11 ==          false(2)
        entPhysicalMfgDate.11 ==        '2002-8-16,15:35:15.0,-4:0'
        entPhysicalUris.11 ==           'URN:CLEI:CNMES23ARA'

    Logical entities -- entLogicalTable; no SNMPv3 support
     2 OSPF instances:
        entLogicalDescr.1 ==            'Acme OSPF v1.1'
        entLogicalType.1 ==             ospf
        entLogicalCommunity.1 ==        'public-ospf1'
        entLogicalTAddress.1 ==         192.0.2.1:161
        entLogicalTDomain.1 ==          snmpUDPDomain
        entLogicalContextEngineID.1 ==  ''
        entLogicalContextName.1 ==      ''

        entLogicalDescr.2 ==            'Acme OSPF v1.1'
        entLogicalType.2 ==             ospf
        entLogicalCommunity.2 ==        'public-ospf2'
        entLogicalTAddress.2 ==         192.0.2.1:161
        entLogicalTDomain.2 ==          snmpUDPDomain
        entLogicalContextEngineID.2 ==  ''
```

```
            entLogicalContextName.2 ==         ''

        2 logical bridges:
            entLogicalDescr.3 ==               'Acme Bridge v2.1.1'
            entLogicalType.3  ==               dot1dBridge
            entLogicalCommunity.3 ==           'public-bridge1'
            entLogicalTAddress.3 ==            192.0.2.1:161
            entLogicalTDomain.3 ==             snmpUDPDomain
            entLogicalContextEngineID.3 ==     ''
            entLogicalContextName.3 ==         ''

            entLogicalDescr.4 ==               'Acme Bridge v2.1.1'
            entLogicalType.4 ==                dot1dBridge
            entLogicalCommunity.4 ==           'public-bridge2'
            entLogicalTAddress.4 ==            192.0.2.1:161
            entLogicalTDomain.4 ==             snmpUDPDomain
            entLogicalContextEngineID.4 ==     ''
            entLogicalContextName.4 ==         ''

    Logical to Physical Mappings:
      1st OSPF instance: uses module 1-port 1
            entLPPhysicalIndex.1.5 ==          5

      2nd OSPF instance: uses module 2-port 1
            entLPPhysicalIndex.2.9 ==          9

      1st bridge group: uses module 1, all ports

      [ed. -- Note that these mappings are included in the table because
      another logical entity (1st OSPF) utilizes one of the
      ports.  If this were not the case, then a single mapping
      to the module (e.g., entLPPhysicalIndex.3.4) would be
      present instead.]
            entLPPhysicalIndex.3.5 ==          5
            entLPPhysicalIndex.3.6 ==          6
            entLPPhysicalIndex.3.7 ==          7

      2nd bridge group: uses module 2, all ports
            entLPPhysicalIndex.4.9  ==         9
            entLPPhysicalIndex.4.10 ==         10
            entLPPhysicalIndex.4.11 ==         11

    Physical to Logical to MIB Alias Mappings -- entAliasMappingTable:
      Example 1: ifIndex values are global to all logical entities
            entAliasMappingIdentifier.5.0 ==  ifIndex.1
            entAliasMappingIdentifier.6.0 ==  ifIndex.2
            entAliasMappingIdentifier.7.0 ==  ifIndex.3
            entAliasMappingIdentifier.9.0 ==  ifIndex.4
```

```
        entAliasMappingIdentifier.10.0 == ifIndex.5
        entAliasMappingIdentifier.11.0 == ifIndex.6


   Example 2: ifIndex values are not shared by all logical entities;
        (Bridge-1 uses ifIndex values 101 - 103 and Bridge-2 uses
  ifIndex values 204-206.)
           entAliasMappingIdentifier.5.0 ==  ifIndex.1
           entAliasMappingIdentifier.5.3 ==  ifIndex.101
           entAliasMappingIdentifier.6.0 ==  ifIndex.2
           entAliasMappingIdentifier.6.3 ==  ifIndex.102
           entAliasMappingIdentifier.7.0 ==  ifIndex.3
           entAliasMappingIdentifier.7.3 ==  ifIndex.103
           entAliasMappingIdentifier.9.0 ==  ifIndex.4
           entAliasMappingIdentifier.9.4 ==  ifIndex.204
           entAliasMappingIdentifier.10.0 == ifIndex.5
           entAliasMappingIdentifier.10.4 == ifIndex.205
           entAliasMappingIdentifier.11.0 == ifIndex.6
           entAliasMappingIdentifier.11.4 == ifIndex.206


  Physical Containment Tree -- entPhysicalContainsTable
    chassis has two containers:
        entPhysicalChildIndex.1.2 ==      2
        entPhysicalChildIndex.1.3 ==      3

    container 1 has a module:
        entPhysicalChildIndex.2.4 ==      4

    container 2 has a module:
        entPhysicalChildIndex.3.8 ==      8

    module 1 has 3 ports:
        entPhysicalChildIndex.4.5 ==      5
        entPhysicalChildIndex.4.6 ==      6
        entPhysicalChildIndex.4.7 ==      7

    module 2 has 3 ports:
        entPhysicalChildIndex.8.9 ==      9
        entPhysicalChildIndex.8.10 ==     10
        entPhysicalChildIndex.8.11 ==     11
```

4.2. Repeaters

   The second example is a 3-slot Hub with 2 backplane ethernet
   segments. Slot three is empty, and the remaining slots contain
   ethernet repeater modules.

   Note that this example assumes an older Repeater MIB implementation,
   (RFC 1516 [RFC1516]) rather than the new Repeater MIB (RFC 2108

   [RFC2108]). The new version contains an object called
   'rptrPortRptrId', which should be used to identify repeater port
   groupings, rather than using community strings or contexts.

   Physical entities -- entPhysicalTable:
      1 Field-replaceable physical chassis:
         entPhysicalDescr.1 ==   'Acme Chassis Model 110'
         entPhysicalVendorType.1 == acmeProducts.chassisTypes.2
         entPhysicalContainedIn.1 == 0
         entPhysicalClass.1 ==   chassis(3)
         entPhysicalParentRelPos.1 ==0
         entPhysicalName.1 ==   '110-B'
         entPhysicalHardwareRev.1 == 'A(1.02.00)'
         entPhysicalSoftwareRev.1 == ''
         entPhysicalFirmwareRev.1 == ''
         entPhysicalSerialNum.1 ==   'C100079294'
         entPhysicalMfgName.1 ==   'Acme'
         entPhysicalModelName.1 ==   '110'
         entPhysicalAlias.1 ==   'bldg09:floor1:rptr18:0067eea0229f'
         entPhysicalAssetID.1 ==   '0007386327'
         entPhysicalIsFRU.1 ==   true(1)

      2 Chassis Ethernet Backplanes:
         entPhysicalDescr.2 ==   'Acme Ethernet Backplane Type A'
         entPhysicalVendorType.2 ==   acmeProducts.backplaneTypes.1
         entPhysicalContainedIn.2 ==   1
         entPhysicalClass.2 ==   backplane(4)
         entPhysicalParentRelPos.2 == 1
         entPhysicalName.2 ==   'B1'
         entPhysicalHardwareRev.2 ==   'A(2.04.01)'
         entPhysicalSoftwareRev.2 ==   ''
         entPhysicalFirmwareRev.2 ==   ''
         entPhysicalSerialNum.2 ==   ''
         entPhysicalMfgName.2 ==   'Acme'
         entPhysicalModelName.2 ==   'BK-A'
         entPhysicalAlias.2 ==   ''
         entPhysicalAssetID.2 ==   ''
         entPhysicalIsFRU.2 ==   false(2)

         entPhysicalDescr.3 ==   'Acme Ethernet Backplane Type A'
         entPhysicalVendorType.3 ==   acmeProducts.backplaneTypes.1
         entPhysicalContainedIn.3 ==   1
         entPhysicalClass.3 ==   backplane(4)
         entPhysicalParentRelPos.3 == 2
         entPhysicalName.3 ==   'B2'
         entPhysicalHardwareRev.3 ==   'A(2.04.01)'
         entPhysicalSoftwareRev.3 ==   ''
         entPhysicalFirmwareRev.3 ==   ''

```
        entPhysicalSerialNum.3 ==       ''
        entPhysicalMfgName.3 ==         'Acme'
        entPhysicalModelName.3 ==       'BK-A'
        entPhysicalAlias.3 ==           ''
        entPhysicalAssetID.3 ==         ''
        entPhysicalIsFRU.3 ==           false(2)


    3 slots within the chassis:
        entPhysicalDescr.4 ==           'Acme Hub Slot Type RB'
        entPhysicalVendorType.4  ==     acmeProducts.slotTypes.5
        entPhysicalContainedIn.4 ==     1
        entPhysicalClass.4 ==           container(5)
        entPhysicalParentRelPos.4 ==    1
        entPhysicalName.4 ==            'Slot 1'
        entPhysicalHardwareRev.4 ==     'B(1.00.03)'
        entPhysicalSoftwareRev.4 ==     ''
        entPhysicalFirmwareRev.4 ==     ''
        entPhysicalSerialNum.4 ==       ''
        entPhysicalMfgName.4 ==         'Acme'
        entPhysicalModelName.4 ==       'RB'
        entPhysicalAlias.4 ==           ''
        entPhysicalAssetID.4 ==         ''
        entPhysicalIsFRU.4 ==           false(2)


        entPhysicalDescr.5 ==           'Acme Hub Slot Type RB'
        entPhysicalVendorType.5  ==     acmeProducts.slotTypes.5
        entPhysicalContainedIn.5 ==     1
        entPhysicalClass.5 ==           container(5)
        entPhysicalParentRelPos.5 ==    2
        entPhysicalName.5 ==            'Slot 2'
        entPhysicalHardwareRev.5 ==     'B(1.00.03)'
        entPhysicalSoftwareRev.5 ==     ''
        entPhysicalFirmwareRev.5 ==     ''
        entPhysicalSerialNum.5 ==       ''
        entPhysicalMfgName.5 ==         'Acme'
        entPhysicalModelName.5 ==       'RB'
        entPhysicalAlias.5 ==           ''
        entPhysicalAssetID.5 ==         ''
        entPhysicalIsFRU.5 ==           false(2)


        entPhysicalDescr.6 ==           'Acme Hub Slot Type RB'
        entPhysicalVendorType.6  ==     acmeProducts.slotTypes.5
        entPhysicalContainedIn.6 ==     1
        entPhysicalClass.6 ==           container(5)
        entPhysicalParentRelPos.6 ==    3
        entPhysicalName.6 ==            'Slot 3'
        entPhysicalHardwareRev.6 ==     'B(1.00.03)'
        entPhysicalSoftwareRev.6 ==     ''
```

```
        entPhysicalFirmwareRev.6 ==     ''
        entPhysicalSerialNum.6 ==       ''
        entPhysicalMfgName.6 ==         'Acme'
        entPhysicalModelName.6 ==       'RB'
        entPhysicalAlias.6 ==           ''
        entPhysicalAssetID.6 ==         ''
        entPhysicalIsFRU.6 ==           false(2)


    Slot 1 contains a plug-in module with 4 10-BaseT ports:
        entPhysicalDescr.7  ==          'Acme 10Base-T Module 114'
        entPhysicalVendorType.7 ==      acmeProducts.moduleTypes.32
        entPhysicalContainedIn.7  ==    4
        entPhysicalClass.7 ==           module(9)
        entPhysicalParentRelPos.7 ==    1
        entPhysicalName.7 ==            'M1'
        entPhysicalHardwareRev.7 ==     'A(1.02.01)'
        entPhysicalSoftwareRev.7 ==     '1.7.2'
        entPhysicalFirmwareRev.7 ==     'A(1.5)'
        entPhysicalSerialNum.7 ==       'C100096244'
        entPhysicalMfgName.7 ==         'Acme'
        entPhysicalModelName.7 =        '114'
        entPhysicalAlias.7 ==           'bldg09:floor1:eng'
        entPhysicalAssetID.7 ==         '0007962951'
        entPhysicalIsFRU.7 ==           true(1)


        entPhysicalDescr.8 ==           'Acme 10Base-T Port RB'
        entPhysicalVendorType.8 ==      acmeProducts.portTypes.10
        entPhysicalContainedIn.8  ==    7
        entPhysicalClass.8 ==           port(10)
        entPhysicalParentRelPos.8 ==    1
        entPhysicalName.8 ==            'Ethernet-A'
        entPhysicalHardwareRev.8 ==     'A(1.04F)'
        entPhysicalSoftwareRev.8 ==     ''
        entPhysicalFirmwareRev.8 ==     '1.4'
        entPhysicalSerialNum.8 ==       ''
        entPhysicalMfgName.8 ==         'Acme'
        entPhysicalModelName.8 ==       'RB'
        entPhysicalAlias.8 ==           ''
        entPhysicalAssetID.8 ==         ''
        entPhysicalIsFRU.8 ==           false(2)


        entPhysicalDescr.9  ==          'Acme 10Base-T Port RB'
        entPhysicalVendorType.9 ==      acmeProducts.portTypes.10
        entPhysicalContainedIn.9 ==     7
        entPhysicalClass.9 ==           port(10)
        entPhysicalParentRelPos.9 ==    2
        entPhysicalName.9 ==            'Ethernet-B'
        entPhysicalHardwareRev.9 ==     'A(1.04F)'
```

```
     entPhysicalSoftwareRev.9 ==     ''
     entPhysicalFirmwareRev.9 ==     '1.4'
     entPhysicalSerialNum.9 ==       ''
     entPhysicalMfgName.9 ==         'Acme'
     entPhysicalModelName.9 =        'RB'
     entPhysicalAlias.9 ==           ''
     entPhysicalAssetID.9 ==         ''
     entPhysicalIsFRU.9 ==           false(2)


     entPhysicalDescr.10 ==          'Acme 10Base-T Port RB'
     entPhysicalVendorType.10 ==     acmeProducts.portTypes.10
     entPhysicalContainedIn.10 ==    7
     entPhysicalClass.10 ==          port(10)
     entPhysicalParentRelPos.10 ==   3
     entPhysicalName.10 ==           'Ethernet-C'
     entPhysicalHardwareRev.10 ==    'B(1.02.07)'
     entPhysicalSoftwareRev.10 ==    ''
     entPhysicalFirmwareRev.10 ==    '1.4'
     entPhysicalSerialNum.10 ==      ''
     entPhysicalMfgName.10 ==        'Acme'
     entPhysicalModelName.10 ==      'RB'
     entPhysicalAlias.10 ==          ''
     entPhysicalAssetID.10 ==        ''
     entPhysicalIsFRU.10 ==          false(2)


     entPhysicalDescr.11 ==          'Acme 10Base-T Port RB'
     entPhysicalVendorType.11  ==    acmeProducts.portTypes.10
     entPhysicalContainedIn.11 ==    7
     entPhysicalClass.11 ==          port(10)
     entPhysicalParentRelPos.11 ==   4
     entPhysicalName.11 ==           'Ethernet-D'
     entPhysicalHardwareRev.11 ==    'B(1.02.07)'
     entPhysicalSoftwareRev.11 ==    ''
     entPhysicalFirmwareRev.11 ==    '1.4'
     entPhysicalSerialNum.11 ==      ''
     entPhysicalMfgName.11 ==        'Acme'
     entPhysicalModelName.11 ==      'RB'
     entPhysicalAlias.11 ==          ''
     entPhysicalAssetID.11 ==        ''
     entPhysicalIsFRU.11 ==          false(2)

   Slot 2 contains another ethernet module with 2 ports.
     entPhysicalDescr.12 ==          'Acme 10Base-T Module Model 4'
     entPhysicalVendorType.12 ==     acmeProducts.moduleTypes.30
     entPhysicalContainedIn.12 =     5
     entPhysicalClass.12 ==          module(9)
     entPhysicalParentRelPos.12 ==   1
     entPhysicalName.12 ==           'M2'
```

```
        entPhysicalHardwareRev.12 ==   'A(1.01.07)'
        entPhysicalSoftwareRev.12 ==   '1.8.4'
        entPhysicalFirmwareRev.12 ==   'A(1.8)'
        entPhysicalSerialNum.12 ==     'C100102384'
        entPhysicalMfgName.12 ==       'Acme'
        entPhysicalModelName.12 ==     '4'
        entPhysicalAlias.12 ==         'bldg09:floor1:devtest'
        entPhysicalAssetID.12 ==       '0007968462'
        entPhysicalIsFRU.12 ==         true(1)

        entPhysicalDescr.13 ==         'Acme 802.3 AUI Port'
        entPhysicalVendorType.13  ==   acmeProducts.portTypes.11
        entPhysicalContainedIn.13 ==   12
        entPhysicalClass.13 ==         port(10)
        entPhysicalParentRelPos.13 ==  1
        entPhysicalName.13 ==          'AUI'
        entPhysicalHardwareRev.13 ==   'A(1.06F)'
        entPhysicalSoftwareRev.13 ==   ''
        entPhysicalFirmwareRev.13 ==   '1.5'
        entPhysicalSerialNum.13 ==     ''
        entPhysicalMfgName.13 ==       'Acme'
        entPhysicalModelName.13 ==     ''
        entPhysicalAlias.13 ==         ''
        entPhysicalAssetID.13 ==       ''
        entPhysicalIsFRU.13 ==         false(2)

        entPhysicalDescr.14 ==         'Acme 10Base-T Port RD'
        entPhysicalVendorType.14  ==   acmeProducts.portTypes.14
        entPhysicalContainedIn.14 ==   12
        entPhysicalClass.14 ==         port(10)
        entPhysicalParentRelPos.14 ==  2
        entPhysicalName.14 ==          'E2'
        entPhysicalHardwareRev.14 ==   'B(1.01.02)'
        entPhysicalSoftwareRev.14 ==   ''
        entPhysicalFirmwareRev.14 ==   '2.1'
        entPhysicalSerialNum.14 ==     ''
        entPhysicalMfgName.14 ==       'Acme'
        entPhysicalModelName.14 ==     ''
        entPhysicalAlias.14 ==         ''
        entPhysicalAssetID.14 ==       ''
        entPhysicalIsFRU.14 ==         false(2)

   Logical entities -- entLogicalTable; with SNMPv3 support
      Repeater 1--comprised of any ports attached to backplane 1
        entLogicalDescr.1 ==           'Acme repeater v3.1'
        entLogicalType.1  ==           snmpDot3RptrMgt
        entLogicalCommunity.1          'public-repeater1'
        entLogicalTAddress.1 ==        192.0.2.1:161
```

```
        entLogicalTDomain.1 ==        snmpUDPDomain
        entLogicalContextEngineID.1 == '80000777017c7d7e7f'H
        entLogicalContextName.1 ==    'repeater1'

   Repeater 2--comprised of any ports attached to backplane 2:
        entLogicalDescr.2 ==          'Acme repeater v3.1'
        entLogicalType.2  ==          snmpDot3RptrMgt
        entLogicalCommunity.2 ==      'public-repeater2'
        entLogicalTAddress.2 ==       192.0.2.1:161
        entLogicalTDomain.2 ==        snmpUDPDomain
        entLogicalContextEngineID.2 == '80000777017c7d7e7f'H
        entLogicalContextName.2 ==    'repeater2'

  Logical to Physical Mappings -- entLPMappingTable:

   repeater1 uses backplane 1, slot 1-ports 1 & 2, slot 2-port 1
   [ed. -- Note that a mapping to the module is not included,
    because this example represents a port-switchable hub.
    Even though all ports on the module could belong to the
    same repeater as a matter of configuration, the LP port
    mappings should not be replaced dynamically with a single
    mapping for the module (e.g., entLPPhysicalIndex.1.7).
    If all ports on the module shared a single backplane connection,
    then a single mapping for the module would be more appropriate.]

       entLPPhysicalIndex.1.2 ==       2
       entLPPhysicalIndex.1.8 ==       8
       entLPPhysicalIndex.1.9 ==       9
       entLPPhysicalIndex.1.13 ==      13

   repeater2 uses backplane 2, slot 1-ports 3 & 4, slot 2-port 2
       entLPPhysicalIndex.2.3 ==       3
       entLPPhysicalIndex.2.10 ==      10
       entLPPhysicalIndex.2.11 ==      11
       entLPPhysicalIndex.2.14 ==      14

  Physical to Logical to MIB Alias Mappings -- entAliasMappingTable:
    Repeater Port Identifier values are shared by both repeaters:
       entAliasMappingIdentifier.8.0 ==     rptrPortGroupIndex.1.1
       entAliasMappingIdentifier.9.0 ==     rptrPortGroupIndex.1.2
       entAliasMappingIdentifier.10.0 ==    rptrPortGroupIndex.1.3
       entAliasMappingIdentifier.11.0 ==    rptrPortGroupIndex.1.4
       entAliasMappingIdentifier.13.0 ==    rptrPortGroupIndex.2.1
       entAliasMappingIdentifier.14.0 ==    rptrPortGroupIndex.2.2

  Physical Containment Tree -- entPhysicalContainsTable
    chassis has two backplanes and three containers:
       entPhysicalChildIndex.1.2 ==    2
```

```
        entPhysicalChildIndex.1.3 ==   3
        entPhysicalChildIndex.1.4 ==   4
        entPhysicalChildIndex.1.5 ==   5
        entPhysicalChildIndex.1.6 ==   6

   container 1 has a module:
        entPhysicalChildIndex.4.7 ==   7

   container 2 has a module
        entPhysicalChildIndex.5.12 ==  12
   [ed. -- in this example, container 3 is empty.]

   module 1 has 4 ports:
        entPhysicalChildIndex.7.8 ==   8
        entPhysicalChildIndex.7.9 ==   9
        entPhysicalChildIndex.7.10 ==  10
        entPhysicalChildIndex.7.11 ==  11

   module 2 has 2 ports:
        entPhysicalChildIndex.12.13 == 13
        entPhysicalChildIndex.12.14 == 14
```

5. Security Considerations

   There are a number of management objects defined in this MIB that
   have a MAX-ACCESS clause of read-write and/or read-create. Such
   objects may be considered sensitive or vulnerable in some network
   environments. The support for SET operations in a non-secure
   environment without proper protection can have a negative effect on
   network operations.

   There are a number of managed objects in this MIB that may contain
   sensitive information. These are:

   entPhysicalDescr
   entPhysicalVendorType
   entPhysicalHardwareRev
   entPhysicalFirmwareRev
   entPhysicalSoftwareRev
   entPhysicalSerialNum
   entPhysicalMfgName
   entPhysicalModelName

      These objects expose information about the physical entities
      within a managed system, which may be used to identify the vendor,
      model, and version information of each system component.

   entPhysicalAssetID

        This object can allow asset identifiers for various system
        components to be exposed, in the event this MIB object is actually
        configured by an NMS application.

    entLogicalDescr
    entLogicalType

        These objects expose the type of logical entities present in the
        managed system.

    entLogicalCommunity

        This object exposes community names associated with particular
        logical entities within the system.

    entLogicalTAddress
    entLogicalTDomain

        These objects expose network addresses that can be used to
        communicate with an SNMP agent on behalf of particular logical
        entities within the system.

    entLogicalContextEngineID
    entLogicalContextName

        These objects identify the authoritative SNMP engine that contains
        information on behalf of particular logical entities within the
        system.

It is thus important to control even GET access to these objects
and possibly to even encrypt the values of these object when
sending them over the network via SNMP. Not all versions of SNMP
provide features for such a secure environment.

SNMPv1 by itself is not a secure environment. Even if the network
itself is secure (for example by using IPSec), even then, there is
no control as to who on the secure network is allowed to access and
GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security
features as provided by the SNMPv3 framework. Specifically, the
use of the User-based Security Model RFC 3414 [RFC3414] and the
View-based Access Control Model RFC 3415 [RFC3415] is recommended.

It is then a customer/user responsibility to ensure that the SNMP
entity giving access to an instance of this MIB, is properly
configured to give access to the objects only to those principals
(users) that have legitimate rights to indeed GET or SET

   (change/create/delete) them.

6. IANA Considerations

   This document defines first version of the IANA-maintained
   IANA-ENTITY-MIB module, which will allow for new physical classes
   to be added to the enumeration in IANAPhysicalClass.
   An Expert Review, as defined in RFC 5226 [RFC5226], is REQUIRED,
   for each modification.

   The MIB module in this document uses the following IANA-assigned
   OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

            Descriptor   OBJECT IDENTIFIER value
            ----------   -----------------------
            entityMIB   { mib-2 47 }

7. Acknowledgements

   The first three versions of RFCs on the ENTITY MIB were authored by
   A. Bierman and K. McCloghrie. The authors would like thank A. Bierman
   and K. McCloghrie for the earlier versions of ENTITY MIB.

   The motivation for the extension to RFC 4133 stems from the
   requirements of the EMAN WG at IETF.

   The authors also thank Juergen Schoenwaelder for his review and
   comments on this draft.

8. References

8.1. Normative References

   [RFC2578]   McCloghrie, K., Perkins, D., and J. Schoenwaelder,
                 "Structure of Management Information Version 2
                 (SMIv2)", STD 58, RFC 2578, April 1999.

   [RFC2579]   McCloghrie, K., Perkins, D., and J. Schoenwaelder,
                 "Textual Conventions for SMIv2", STD 58, RFC 2579,
                 April 1999.

   [RFC2580]   McCloghrie, K., Perkins, D., and J. Schoenwaelder,
                 "Conformance Statements for SMIv2", STD 58, RFC 2580,
                 April 1999.

   [RFC3411]   Harrington, D., Presuhn, R., and B. Wijnen, "An
                 Architecture for Describing Simple Network Management

                        Protocol (SNMP) Management Frameworks", STD 62, RFC
                        3411, December 2002.

   [RFC3417]   Presuhn, R., "Transport Mappings for the Simple Network
                Management Protocol (SNMP)", STD 62, RFC 3417, December
                2002.

   [RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter,
                "Uniform Resource Identifier (URI): Generic Syntax",
                STD 66, RFC 3986, January 2005.

   [RFC2737]   McCloghrie, K. and A. Bierman, "Entity MIB (Version
                2)", RFC 2737, December 1999.

   [RFC4133]   McCloghrie, K. and A. Bierman, "Entity MIB (Version
                3)", RFC 4133, August 2005.
   [RFC4122]   Leach P., Mealling M., and R. Salz, " A Universally
                Unique IDentifier (UUID) URN Namespace", RFC 4122,
                July 2005.
   [RFC5226]   Narten T., and H. Alvestrand, " Guidelines for Writing
                an IANA Considerations Section in RFCs" RFC 5226, May
                2008.

8.2. Informative References

   [RFC1157]   Case, J., Fedor, M., Schoffstall, M., and J. Davin,
                "Simple Network Management Protocol", STD 15, RFC 1157,
                May 1990.

   [RFC1493]   Decker, E., Langille, P., Rijsinghani, A., and K.
                McCloghrie, "Definitions of Managed Objects for
                Bridges", RFC 1493, July 1993.

   [RFC1516]   McMaster, D. and K. McCloghrie, "Definitions of Managed
                Objects for IEEE 802.3 Repeater Devices", RFC 1516,
                September 1993.

   [RFC4122]   Leach, P., Mealling, M., Salz, R., "A Universally
                Unique IDentifier (UUID) URN Namespace", RFC 4122, July
                2005.

   [RFC2108]   de Graaf, K., Romascanu, D., McMaster, D., and K.
                McCloghrie, "Definitions of Managed Objects for IEEE
                802.3 Repeater Devices using SMIv2", RFC 2108, February
                1997.

   [RFC2737]   McCloghrie, K. and A. Bierman, "Entity MIB (Version
                2)", RFC 2737, December 1999.

   [RFC4133]   McCloghrie, K. and A. Bierman, "Entity MIB (Version
                   3)", RFC 4133, August 2005.

   [RFC2863]   McCloghrie, K. and F. Kastenholz, "The Interfaces Group
                   MIB", RFC 2863, June 2000.

   [RFC3406]   Daigle, L., van Gulik, D., Iannella, R., and P.
                   Faltstrom, "Uniform Resource Names (URN) Namespace
                   Definition Mechanisms", BCP 66, RFC 3406, October 2002.

   [RFC3410]   Case, J., Mundy, R., Partain, D., and B. Stewart,
                   "Introduction and Applicability Statements for
                   Internet-Standard Management Framework", RFC 3410,
                   December 2002.

   [RFC3414]   Blumenthal, U. and B. Wijnen, "User-based Security
                   Model (USM) for version 3 of the Simple Network
                   Management Protocol (SNMPv3)", STD 62, RFC 3414,
                   December 2002.

   [RFC3415]   Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based
                   Access Control Model (VACM) for the Simple Network
                   Management Protocol (SNMP)", STD 62, RFC 3415, December
                   2002.

   [RFC4152]   Tesink, K. and R. Fox, "A Uniform Resource Name (URN)
                   Namespace for the CLEI Code", RFC 4152, August 2005.

   [EMAN-FMWK]   Parello, J. Claise, B. Schoening, B. Quittek, J. and
                   Nordman, B. "Energy Management Framework", draft-ietf-
                   eman-framework-05, July 2012.

   [T1.213]   ATIS T1.213-2001, "Coded Identification of Equipment
                   Entities in the North American Telecommunications
                   System for Information Exchange", 2001, www.ansi.org.

   [T1.213a]   ATIS T1.213a, "Supplement to T1.213-2001, Coded
                   Identification of Equipment Entities in the North
                   American Telecommunications System for Information
                   Exchange, to correct the representation of the Basic
                   Code in Figure B.1", 2001, www.ansi.org.

Authors' Addresses

   Andy Bierman
   Yumaworks

   Email: andy@yumaworks.com

   Dan Romascanu
   AVAYA
   Park Atidim, Bldg. #3
   Tel Aviv, 61581
   Israel

   Phone: +972-3-6458414
   Email: dromasca@avaya.com

   Juergen Quittek
   NEC Europe Ltd.
   Network Research Division
   Kurfuersten-Anlage 36
   Heidelberg 69115
   DE

   Phone: +49 6221 4342-115
   Email: quittek@neclab.eu

   Mouli Chandramouli
   Cisco Systems, Inc.
   Sarjapur Outer Ring Road
   Bangalore 560103
   IN

   Phone: +91 80 4429 2409
   Email: moulchan@cisco.com