Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

J. Quittek
R. Winter
T. Dietz
NEC Europe Ltd.
October 22, 2012

# Definition of Managed Objects for Battery Monitoring
# draft-ietf-eman-battery-mib-07

## 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 defines managed objects that provide information on the status of batteries in managed devices.

## Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Table of Contents

## 1. Introduction

Today, more and more managed devices contain batteries that supply them with power when disconnected from electrical power distribution grids. Common examples are nomadic and mobile devices, such as notebook computers, netbooks, and smart phones. The status of batteries in such a device, particularly the charging status is typically controlled by automatic functions that act locally on the device and manually by users of the device.

In addition to this, there is a need to monitor battery status of these devices by network management systems. This document defines a portion of the Management Information Base (MIB) that provides a means for monitoring batteries in or attached to managed devices. The Battery MIB module defined in Section 4 meets the requirements for monitoring the status of batteries specified in [I-D.ietf-eman-requirements].

The Battery MIB module provides for monitoring the battery status. According to the framework for energy management [I-D.ietf-eman-framework] it is an Energy Managed Object, and thus, MIB modules such as the Power and Energy Monitoring MIB [I-D.ietf-eman-energy-monitoring-mib] could in principle be implemented for batteries. The Battery MIB extends the more generic aspects of energy management by adding battery-specific information. Amongst other things, the Battery MIB enables the monitoring of:

- the current charge of a battery,
- the age of a battery (charging cycles),
- the state of a battery (e.g. being re-charged),
- last usage of a battery,
- maximum energy provided by a battery (remaining and total capacity).

Further, means are provided for battery-powered devices to send notifications when the current battery charge has dropped below a certain threshold in order to inform the management system of needed replacement. The same applies to the age of a battery.

Many battery-driven devices have existing instrumentation for monitoring the battery status, because this is already needed for local control of the battery by the device. This reduces the effort for implementing the managed objects defined in this document. For many devices only additional software will be needed but no additional hardware instrumentation for battery monitoring.

Since there are a lot of devices in use that contain more than one battery, means for battery monitoring defined in this document

support addressing multiple batteries within a single device. Also, batteries today often come in packages that can include identification and might contain additional hardware and firmware. The former allows to trace a battery and allows continuous monitoring even if the battery is e.g. installed in another device. The firmware version is useful information as the battery behavior might be different for different firmware versions.

Not explicitly in scope of definitions in this document are very small backup batteries, such as for example, batteries used on PC motherboard to run the clock circuit and retain configuration memory while the system is turned off. Other means may be required for reporting on these batteries. However, the MIB module defined in Section 3.1 can be used for this purpose.

A traditional type of managed device containing batteries is an Uninterruptible Power Supply (UPS) system; these supply other devices with electrical energy when the main power supply fails. There is already a MIB module for managing UPS systems defined in RFC 1628 [RFC1628]. This module includes managed objects for monitoring the batteries contained in an UPS system. However, the information provided by these objects is limited and tailored the particular needs of UPS systems.

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].

## 2. The Internet-Standard 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 MIB modules that are compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58,RFC 2580 [RFC2580].

# 3. Design of the Battery MIB Module

## 3.1. MIB Module Structure

The Battery MIB module defined in this document defines objects for reporting information about batteries. All managed objects providing information of the status of a battery are contained in a single table called batteryTable. The batteryTable contains one conceptual row per battery.

If there is an implementation of the Entity MIB module [RFC4133] that identifies the batteries to be reported on by individual values for managed object entPhysicalIndex, then it is REQUIRED that these values are used as index values for the batteryTable.

The kind of entity in the entPhysicalTable of the Entity MIB module is indicated by the value of enumeration object entPhysicalClass. Since there is no value called 'battery' defined for this object, it is RECOMMENDED that for batteries the value of this object is chosen to be powerSupply(6).

The batteryTable contains three groups of objects. The first group (OIDs ending with 2-11) provides information on static properties of the battery. The second group of objects (OIDs ending with 12-19) provides information on the current battery state, if it is charging or discharging, how much it is charged, its remaining capacity, the number of experienced charging cycles, etc.

```
batteryTable(1)
+--batteryEntry(1) [batteryIndex]
   +-- --- Integer32       batteryIndex(1)
   +-- r-n SnmpAdminString batteryIdentifier(2)
   +-- r-n SnmpAdminString batteryFirmwareVersion(3)
   +-- r-n Enumeration     batteryType(4)
   +-- r-n Unsigned32      batteryTechnology(5)
   +-- r-n Unsigned32      batteryDesignVoltage(6)
   +-- r-n Unsigned32      batteryNumberOfCells(7)
   +-- r-n Unsigned32      batteryDesignCapacity(8)
   +-- r-n Unsigned32      batteryMaxChargingCurrent(9)
   +-- r-n Unsigned32      batteryTrickleChargingCurrent(10)
   +-- r-n Unsigned32      batteryActualCapacity(11)
   +-- r-n Unsigned32      batteryChargingCycleCount(12)
   +-- r-n DateAndTime     batteryLastChargingCycleTime(13)
   +-- r-n Enumeration     batteryChargingOperState(14)
   +-- rwn Enumeration     batteryChargingAdminState(15)
   +-- r-n Unsigned32      batteryActualCharge(16)
   +-- r-n Unsigned32      batteryActualVoltage(17)
   +-- r-n Integer32       batteryActualCurrent(18)
   +-- r-n Integer32       batteryTemperature(19)
   +-- rwn Unsigned32      batteryAlarmLowCharge(20)
   +-- rwn Unsigned32      batteryAlarmLowVoltage(21)
   +-- rwn Unsigned32      batteryAlarmLowCapacity(22)
   +-- rwn Unsigned32      batteryAlarmHighCycleCount(23)
   +-- rwn Integer32       batteryAlarmHighTemperature(24)
   +-- rwn Integer32       batteryAlarmLowTemperature(25)
```

The third group of objects in this table (OIDs ending with 20-25) indicates thresholds which can be used to raise an alarm if a property of the battery exceeds one of them. Raising an alarm may include sending a notification.

The Battery MIB defines four notifications. One indicating a low battery charging state, one indicating an aged battery that may need to be replaced and two dealing with battery temperature. The temperature-related notifications are either indicating the battery temperature to have risen above or fallen below a predefined value.

## 3.2. Battery Technologies

Static information in the batteryTable includes battery type and technology. The battery type distinguishes primary (not rechargeable) batteries from rechargeable (secondary) batteries and capacitors. The battery technology describes the actual technology of a battery, which typically is a chemical technology.

Since battery technologies are subject of intensive research and

widely used technologies are often replaced by successor technologies within an few years, the list of battery technologies was not chosen as a fixed list. Instead, IANA has created a registry for battery technologies at http://www.iana.org/assignments/eman where numbers are assigned to battery technologies (TBD).

The table below shows battery technologies known today that are in commercial use with the numbers assigned to them by IANA. New entries can be added to the IANA registry if new technologies are developed or if missing technologies are identified. Note that there exists a huge number of battery types that are not listed in the IANA registry. Many of them are experimental or cannot be used in an economically useful way. New entries should be added to the IANA registry only if the respective technologies are in commercial use and relevant to standardized battery monitoring over the Internet.

| battery technology | assigned number |
|---|---|
| Unknown | 1 |
| Other | 2 |
| Zinc-carbon | 3 |
| Zinc chloride | 4 |
| Nickel oxyhydroxide | 5 |
| Lithium-copper oxide | 6 |
| Lithium-iron disulfide | 7 |
| Lithium-manganese dioxide | 8 |
| Zinc-air | 9 |
| Silver oxide | 10 |
| Alkaline | 11 |
| Lead acid | 12 |
| Nickel-cadmium | 13 |
| Nickel-metal hybride | 14 |
| Nickel-zinc | 15 |
| Lithium-ion | 16 |
| Lithium polymer | 17 |
| Double layer capacitor | 18 |

## 3.3. Charging Cycles

The lifetime of a battery can be approximated using the measure of charging cycles. A commonly used definition of a charging cycle is the amount of discharge equal to the design (or nominal) capacity of the battery [SBS]. This means that a single charging cycle may include several steps of partial charging and discharging until the amount of discharging has reached the design capacity of the battery.

After that the next charging cycle immediately starts.

## 4. Definitions

```
BATTERY-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    mib-2, Integer32, Unsigned32
        FROM SNMPv2-SMI                                 -- RFC2578
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB                         -- RFC3411
    DateAndTime
        FROM SNMPv2-TC                                  -- RFC2579
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF;                               -- RFC2580

batteryMIB MODULE-IDENTITY
    LAST-UPDATED "201106261200Z"         -- 26 june 2010
    ORGANIZATION "IETF EMAN Working Group"
    CONTACT-INFO
        "General Discussion: eman@ietf.org
        To Subscribe: http://www.ietf.org/mailman/listinfo/eman
        Archive: http://www.ietf.org/mail-archive/web/eman

        Editor:
          Juergen Quittek
          NEC Europe Ltd.
          NEC Laboratories Europe
          Kurfuersten-Anlage 36
          69115 Heidelberg
          Germany
          Tel: +49 6221 4342-115
          Email: quittek@neclab.eu"

    DESCRIPTION
        "This MIB module defines a set of objects for monitoring
        batteries of networked devices and of their components.

        Copyright (c) 2010 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
```

```
            (http://trustee.ietf.org/license-info).

            This version of this MIB module is part of RFC yyyy; see
            the RFC itself for full legal notices."
    -- replace yyyy with actual RFC number & remove this notice

    --  Revision history

        REVISION "201106261200Z"         -- 26 June 2010
        DESCRIPTION
            "Initial version, published as RFC yyyy."
    -- replace yyyy with actual RFC number & remove this notice

        ::= { mib-2 zzz }
    -- zzz to be assigned by IANA.

    --*****************************************************************
    -- Top Level Structure of the MIB module
    --*****************************************************************

    batteryNotifications OBJECT IDENTIFIER ::= { batteryMIB 0 }
    batteryObjects       OBJECT IDENTIFIER ::= { batteryMIB 1 }
    batteryConformance   OBJECT IDENTIFIER ::= { batteryMIB 2 }

    --=================================================================
    -- 1. Object Definitions
    --=================================================================

    -----------------------------------------------------------------
    -- 1.1. Battery Table
    -----------------------------------------------------------------
    batteryTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF BatteryEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table provides information on batteries.
            It contains one conceptual row per battery."
        ::= { batteryObjects 1 }

    batteryEntry OBJECT-TYPE
        SYNTAX      BatteryEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An entry providing information on a battery."
        INDEX  { batteryIndex }
        ::= { batteryTable 1 }
```

```
   BatteryEntry ::=
       SEQUENCE {
          batteryIndex                    Integer32,
          batteryIdentifier               SnmpAdminString,
          batteryFirmwareVersion          SnmpAdminString,
          batteryType                     INTEGER,
          batteryTechnology               Unsigned32,
          batteryDesignVoltage            Unsigned32,
          batteryNumberOfCells            Unsigned32,
          batteryDesignCapacity           Unsigned32,
          batteryMaxChargingCurrent       Unsigned32,
          batteryTrickleChargingCurrent   Unsigned32,
          batteryActualCapacity           Unsigned32,
          batteryChargingCycleCount       Unsigned32,
          batteryLastChargingCycleTime    DateAndTime,
          batteryChargingOperState        INTEGER,
          batteryChargingAdminState       INTEGER,
          batteryActualCharge             Unsigned64,
          batteryActualVoltage            Unsigned32,
          batteryActualCurrent            Integer32,
          batteryTemperature              Integer32,
          batteryAlarmLowCharge           Unsigned32,
          batteryAlarmLowVoltage          Unsigned32,
          batteryAlarmLowCapacity         Unsigned32,
          batteryAlarmHighCycleCount      Unsigned32,
          batteryAlarmHighTemperature     Integer32,
          batteryAlarmLowTemperature      Integer32
       }

   batteryIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This object identifies a battery for which status is
           reported. Index values MUST be locally unique.

           If there is an instance of the entPhysicalTable (defined in
           the ENTITY-MIB module, see RFC 4133) with an individual
           entry for each battery, then it is REQUIRED that values of
           batteryIndex match the corresponding values of
           entPhysicalIndex for the batteries. Otherwise, index values
           may be chosen arbitrarily."
       ::= { batteryEntry 1 }

   batteryIdentifier OBJECT-TYPE
       SYNTAX      SnmpAdminString
       MAX-ACCESS  read-only
```

```
        STATUS      current
        DESCRIPTION
            "This object contains an identifier for the battery.

            Many manufacturers deliver not pure batteries but battery
            packages including additional hardware and firmware.
            Typically, these modules include an identifier that can be
            retrieved by a device at which a battery has been installed.
            The identifier is useful when batteries are removed and
            re-installed at the same or other devices. Then the device
            or the network management system can trace batteries and
            achieve continuity of battery monitoring.

            If the battery identifier cannot be represented using the
            ISO/IEC IS 10646-1 character set, then a hexadecimal
            encoding of a binary representation of the battery
            identifier must be used.

            The value of this object must be an empty string if there
            is no battery identifier or if the battery idenitfier is
            unknown."
        ::= { batteryEntry 2 }

    batteryFirmwareVersion OBJECT-TYPE
        SYNTAX      SnmpAdminString
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the version number of the firmware
            that is included in a battery module.

            Many manufacturers deliver not pure batteries but battery
            packages including additional hardware and firmware.

            Since the bahavior of the battery may change with the
            firmware, it may be useful to retrieve the firmware version
            number.

            The value of this object must be an empty string if there
            is no firmware or if the version number of the firware is
            unknown."
        ::= { batteryEntry 3 }

    batteryType OBJECT-TYPE
        SYNTAX      INTEGER {
                        unknown(1),
                        other(2),
                        primary(3),
```

```
                    rechargeable(4),
                    capacitor(5)
                }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the type of battery.
            It distinguishes between primary (not rechargeable)
            batteries, rechargeable (secondary) batteries and capacitors
            which are not really batteries but often used in the same
            way as a battery.

            The value other(2) can be used if the battery type is known
            but none of the ones above.  Value unknown(1) is to be used
            if the type of battery cannot be determined."
        ::= { batteryEntry 4 }

    batteryTechnology OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the technology used by the battery.
            Numbers identifying battery types are registered at IANA.
            A current list of assignments can be found at
            <http://www.iana.org/assignments/eman>.

            Value 0 (unknown) MUST be used if the type of battery
            cannot be determined.

            Value 1 (other) can be used if the battery type is known
            but not one of the types already registered at IANA."
        ::= { batteryEntry 5 }

    batteryDesignVoltage OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "millivolt"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the design (or nominal) voltage of the
            battery in units of millivolt (mV).

            Note that the design voltage is a constant value and
            typically different from the actual voltage of the battery.

            A value of 0 indicates that the design voltage is unknown."
        ::= { batteryEntry 6 }
```

```
   batteryNumberOfCells OBJECT-TYPE
       SYNTAX      Unsigned32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object indicates the number of cells contained in the
           battery.

           A value of 0 indicates that the number of cells is unknown."
       ::= { batteryEntry 7 }

   batteryDesignCapacity OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "milliampere hours"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object provides the design (or nominal) capacity of the
           battery in units of milliampere hours (mAh).

           Note that the design capacity is a constant value and
           typically different from the actual capacity of the battery.
           Usually, this is a value provided by the manufacturer of the
           battery.

           A value of 0 indicates that the design capacity is
           unknown."
       ::= { batteryEntry 8 }

   batteryMaxChargingCurrent OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "milliampere"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object provides the maximal current to be used for
           charging the battery in units of milliampere (mA).

           Note that the maximal charging current may not lead to
           optimal charge of the battery and that some batteries can
           only be charged with the maximal current for a limited
           amount of time.

           A value of 0 indicates that the maximal charging current is
           unknown."
       ::= { batteryEntry 9 }

   batteryTrickleChargingCurrent OBJECT-TYPE
```

```
        SYNTAX      Unsigned32
        UNITS       "milliampere"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the recommended current to be used for
            trickle charging the battery in units of milliampere (mA).

            Typically, this is a value recommended by the manufacturer
            of the battery or by the manufacturer of the charging
            circuit.

            A value of 0 indicates that the recommended trickle charging
            current is unknown."
        ::= { batteryEntry 10 }

    batteryActualCapacity OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "milliampere hours"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the actual capacity of the
            battery in units of milliampere hours (mAh).

            Typically, the actual capacity of a battery decreases
            with time and with usage of the battery. It is usually
            lower than the design capacity

            Note that the actual capacity needs to be measured and is
            typically an estimate based on observed discharging and
            charging cycles of the battery.

            A value of 'ffffffff'H indicates that the actual capacity
            cannot be determined."
        ::= { batteryEntry 11 }

    batteryChargingCycleCount OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the number of completed charging
            cycles that the battery underwent. In line with the
            Smart Battery Data Specification Revision 1.1, a charging
            cycle is defined as the process of discharging the battery
            by a total amount equal to the battery design capacity as
            given by object batteryDesignCapacity. A charging cycle
```

```
            may include several steps of charging and discharging the
            battery until the discharging amount given by
            batteryDesignCapacity has been reached. As soon as a
            charging cycle has been completed the next one starts
            immediately independent of the battery's current charge at
            the end of the cycle.


            For batteries of type primary(1) the value of this object is
            always 0.

            A value of 'ffffffff'H indicates that the number of charging
            cycles cannot be determined."
        ::= { batteryEntry 12 }

    batteryLastChargingCycleTime OBJECT-TYPE
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The date and time of the last charging cycle.  The value
            '0000000000000000'H is returned if the battery has not been
            charged yet or if the last charging time cannot be
            determined.

            For batteries of type primary(1) the value of this object is
            always '0000000000000000'H."
        ::= { batteryEntry 13 }

    batteryChargingOperState OBJECT-TYPE
        SYNTAX      INTEGER {
                        unknown(1),
                        charging(2),
                        fastCharging(3),
                        maintainingCharge(4),
                        noCharging(5),
                        discharging(6)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the current charging state of the
            battery.

            Value unknown(1) indicates that the charging state of the
            battery cannot be determined.

            Value charging(2) indicates that the battery is being
```

```
            charged in a way that the charge of the battery increases.

            Value fastCharging(3) indicated that the battery is being
            charged rapidly, i.e. faster than in the charging(2) state.
            If multiple fast charging states exist, all of these
            states are indicated by fastCharging(3).

            Value maintainingCharge(4) indicates that the battery is
            being charged with a low current that compensates
            self-discharging. This includes trickle charging, float
            charging and other methods for maintaining the current
            charge of a battery.

            Value noCharging(5) indicates that the battery is not being
            charged or discharged by electric current between the
            battery and electric circuits external to the battery.
            Note that the battery may still be subject to
            self-discharging.

            Value discharging(6) indicates that the battery is being
            discharged and that the charge of the battery decreases."
        ::= { batteryEntry 14 }

    batteryChargingAdminState OBJECT-TYPE
        SYNTAX      INTEGER {
                        charging(2),
                        fastCharging(3),
                        maintainingCharge(4),
                        noCharging(5),
                        discharging(6),
                        notSet(7)
                    }
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The value of this object indicates the desired status of
            the charging state of the battery. The real state is
            indicated by object batteryChargingOperState. See the
            definition of object batteryChargingOperState for a
            description of the values.

            When this object is initialized by an implementation of the
            BATTERY-MIB module, its value is set to notSet(7).

            However, a SET request can only set this object to either
            charging(2), fastCharging(3), maintainingCharge(4),
            noCharging(5), or discharging(6). Attempts to set this
            object to notSet(7) will always fail with an
```

```
            'inconsistentValue' error. In case multiple fast charging
            states exist, the battery logic can choose an appropriate
            fast charging state - preferably the fastest.

            When the batteryChargingAdminState object is set, then the
            BATTERY-MIB implementation must try to set the battery
            to the indicated state. The result will be indicated by
            object batteryChargingOperState.

            Due to operational conditions and limitations of the
            implementation of the BATTERY-MIB module, changing the
            battery status according to a set value of object
            batteryChargingAdminState may not be possible.

            Setting the value of object batteryChargingAdminState
            may result in not changing the state of the battery
            to this value or even in setting the charging state
            to another value. For example, setting
            batteryChargingAdminState to value fastCharging(3) may
            have no effect when the battery logic is not allowing
            fast charging due to temperature constraints."

        ::= { batteryEntry 15 }

    batteryActualCharge OBJECT-TYPE
        SYNTAX      Unsigned64
        UNITS       "milliampere hours"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the actual charge of the battery
            in units of milliampere hours (mAh).

            Note that the actual charge needs to be measured and is
            typically an estimate based on observed discharging and
            charging cycles of the battery.

            A value of 'ffffffff'H indicates that the actual charge
            cannot be determined."
        ::= { batteryEntry 16 }

    batteryActualVoltage OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "millivolt"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the actual voltage of the battery
```

```
            in units of millivolt (mV).

            A value of 'ffffffff'H indicates that the actual voltage
            cannot be determined."
        ::= { batteryEntry 17 }

    batteryActualCurrent OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "milliampere"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object provides the actual charging or discharging
            current of the battery in units of milliampere (mA).
            Charging current is represented by positive values,
            discharging current is represented by negative values.

            A value of '7fffffff'H indicates that the actual current
            cannot be determined."
        ::= { batteryEntry 18 }

    batteryTemperature OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "deci-degrees Celsius"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The ambient temperature at or near the battery.

            A value of '7fffffff'H indicates that the temperature
            cannot be determined."
        ::= { batteryEntry 19 }

    batteryAlarmLowCharge OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "milliampere hours"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "This object provides the lower threshold value for object
            batteryActualCharge.  If the value of object
            batteryActualCharge falls below this threshold,
            a low battery alarm will be raised.  The alarm procedure may
            include generating a batteryLowNotification.

            A value of 0 indicates that no alarm will be raised for any
            value of object batteryActualCharge."
        ::= { batteryEntry 20 }
```

```
   batteryAlarmLowVoltage OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "millivolt"
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "This object provides the lower threshold value for object
           batteryActualVoltage.  If the value of object
           batteryActualVoltage falls below this threshold,
           a low battery alarm will be raised.  The alarm procedure may
           include generating a batteryLowNotification.

           A value of 0 indicates that no alarm will be raised for any
           value of object batteryActualVoltage."
       ::= { batteryEntry 21 }

   batteryAlarmLowCapacity OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "milliampere hours"
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "This object provides the lower threshold value for object
           batteryActualCapacity.  If the value of object
           batteryActualCapacity falls below this threshold,
           a battery aging alarm will be raised.  The alarm procedure
           may include generating a batteryAgingNotification.

           A value of 0 indicates that no alarm will be raised for any
           value of object batteryActualCapacity."
       ::= { batteryEntry 22 }

   batteryAlarmHighCycleCount OBJECT-TYPE
       SYNTAX      Unsigned32
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "This object provides the upper threshold value for object
           batteryChargingCycleCount.  If the value of object
           batteryChargingCycleCount rises above this threshold,
           a battery aging alarm will be raised.  The alarm procedure
           may include generating a batteryAgingtNotification.

           A value of 0 indicates that no alarm will be raised for any
           value of object batteryChargingCycleCount."
       ::= { batteryEntry 23 }

   batteryAlarmHighTemperature OBJECT-TYPE
```

```
        SYNTAX      Integer32
        UNITS       "deci-degrees Celsius"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "This object provides the upper threshold value for object
            batteryTemperature.  If the value of object
            batteryTemperature rises above this threshold, a battery
            high temperature alarm will be raised.  The alarm procedure
            may include generating a batteryHighTemperatNotification.

            A value of '7fffffff'H indicates that no alarm will be
            raised for any value of object batteryTemperature."
        ::= { batteryEntry 24 }

    batteryAlarmLowTemperature OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "deci-degrees Celsius"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "This object provides the lower threshold value for object
            batteryTemperature.  If the value of object
            batteryTemperature rises above this threshold, a battery
            low temperature alarm will be raised.  The alarm procedure
            may include generating a batteryLowTemperatNotification.

            A value of '7fffffff'H indicates that no alarm will be
            raised for any value of object batteryTemperature."
        ::= { batteryEntry 25 }


    --================================================================
    -- 2. Notifications
    --================================================================

    batteryLowNotification NOTIFICATION-TYPE
        OBJECTS     {
            batteryActualCharge,
            batteryActualVoltage
        }
        STATUS      current
        DESCRIPTION
            "This notification can be generated when the current charge
            (batteryActualCharge) or the current voltage
            (batteryActualVoltage) of the battery falls below a
            threshold defined by object batteryAlarmLowCharge or object
            batteryAlarmLowVoltage, respectively. The notification can
```

```
            only be sent again when the current voltage or the current
            charge become higher than the respective thresholds
            through charging before falling below the thresholds again
            (to avoid fluctuations through e.g. temperature). The
            notification can also be sent again when a charging process
            is interrupted and either the battery charge
            (batteryActualCharge) or battery voltage
            (batteryActualVoltage) is still below either the value of
            the object batteryAlarmLowCharge or the value of object
            batteryAlarmLowVoltage."
        ::= { batteryNotifications 1 }

    batteryAgingNotification NOTIFICATION-TYPE
        OBJECTS     {
            batteryActualCapacity,
            batteryChargingCycleCount
        }
        STATUS      current
        DESCRIPTION
            "This notification can be generated when the actual
            capacity (batteryActualCapacity) falls below a threshold
            defined by object batteryAlarmLowCapacity
            or when the charging cycle count of the battery
            (batteryChargingCycleCount) exceeds the threshold defined
            by object batteryAlarmHighCycleCount."
        ::= { batteryNotifications 2 }

    batteryHighTemperatNotification NOTIFICATION-TYPE
        OBJECTS     {
            batteryTemperature
        }
        STATUS      current
        DESCRIPTION
            "This notification can be generated when the measured
            temperature (batteryTemperature) rises above a threshold
            defined by object batteryAlarmHighTemperature."
        ::= { batteryNotifications 3 }

    batteryLowTemperatNotification NOTIFICATION-TYPE
        OBJECTS     {
            batteryTemperature
        }
        STATUS      current
        DESCRIPTION
            "This notification can be generated when the measured
            temperature (batteryTemperature) falls below a threshold
            defined by object batteryAlarmLowTemperature."
        ::= { batteryNotifications 4 }
```

```
   --=================================================================
   -- 3. Conformance Information
   --=================================================================

   batteryCompliances OBJECT IDENTIFIER ::= { batteryConformance 1 }
   batteryGroups      OBJECT IDENTIFIER ::= { batteryConformance 2 }

   -----------------------------------------------------------------
   -- 3.1. Compliance Statements
   -----------------------------------------------------------------

   batteryCompliance MODULE-COMPLIANCE
       STATUS      current
       DESCRIPTION
           "The compliance statement for implementations of the
           POWER-STATE-MIB module.

           A compliant implementation MUST implement the objects
           defined in the mandatory groups batteryDescriptionGroup
           and batteryStatusGroup."
       MODULE  -- this module
           MANDATORY-GROUPS {
               batteryDescriptionGroup,
               batteryStatusGroup
           }

           GROUP   batteryAlarmThresholdsGroup
           DESCRIPTION
              "A compliant implementation does not have to implement
               the batteryAlarmThresholdsGroup."

           GROUP   batteryNotificationsGroup
           DESCRIPTION
              "A compliant implementation does not have to implement
               the batteryNotificationsGroup."

           GROUP   batteryAdminGroup
           DESCRIPTION
              "A compliant implementation does not have to implement
               the batteryAdminGroup."

           OBJECT batteryAlarmLowCharge
           MIN-ACCESS  read-only
           DESCRIPTION
               "The agent is not required to support set
               operations to this object."

           OBJECT batteryAlarmLowVoltage
```

```
            MIN-ACCESS  read-only
            DESCRIPTION
                "The agent is not required to support set
                operations to this object."

            OBJECT batteryAlarmLowCapacity
            MIN-ACCESS  read-only
            DESCRIPTION
                "The agent is not required to support set
                operations to this object."

            OBJECT batteryAlarmHighCycleCount
            MIN-ACCESS  read-only
            DESCRIPTION
                "The agent is not required to support set
                operations to this object."

            OBJECT batteryHighTemperatNotification
            MIN-ACCESS  read-only
            DESCRIPTION
                "The agent is not required to support set
                operations to this object."

        ::= { batteryCompliances 1 }

    ----------------------------------------------------------------------
    -- 3.2. MIB Grouping
    ----------------------------------------------------------------------

    batteryDescriptionGroup OBJECT-GROUP
        OBJECTS {
           batteryIdentifier,
           batteryFirmwareVersion,
           batteryType,
           batteryTechnology,
           batteryDesignVoltage,
           batteryNumberOfCells,
           batteryDesignCapacity,
           batteryMaxChargingCurrent,
           batteryTrickleChargingCurrent
        }
        STATUS      current
        DESCRIPTION
           "A compliant implementation MUST implement the objects
           contained in this group."
        ::= { batteryGroups 1 }

    batteryStatusGroup OBJECT-GROUP
```

```
        OBJECTS {
           batteryActualCapacity,
           batteryChargingCycleCount,
           batteryLastChargingCycleTime,
           batteryChargingOperState,
           batteryActualCharge,
           batteryActualVoltage,
           batteryActualCurrent,
           batteryTemperature
        }
        STATUS      current
        DESCRIPTION
           "A compliant implementation MUST implement the objects
           contained in this group."
        ::= { batteryGroups 2 }

    batteryAdminGroup OBJECT-GROUP
        OBJECTS {
           batteryChargingAdminState
        }
        STATUS      current
        DESCRIPTION
           "A compliant implementation does not have to implement the
           object contained in this group."
        ::= { batteryGroups 3 }

    batteryAlarmThresholdsGroup OBJECT-GROUP
        OBJECTS {
           batteryAlarmLowCharge,
           batteryAlarmLowVoltage,
           batteryAlarmLowCapacity,
           batteryAlarmHighCycleCount,
           batteryAlarmHighTemperature,
           batteryAlarmLowTemperature
        }
        STATUS      current
        DESCRIPTION
           "A compliant implementation does not have to implement the
           objects contained in this group."
        ::= { batteryGroups 4 }

    batteryNotificationsGroup NOTIFICATION-GROUP
        NOTIFICATIONS {
           batteryLowNotification,
           batteryAgingNotification,
           batteryHighTemperatNotification,
           batteryLowTemperatNotification
        }
```

```
        STATUS      current
        DESCRIPTION
            "A compliant implementation does not have to implement the
            notifications contained in this group."
        ::= { batteryGroups 5 }
    END
```

## 5. Security Considerations

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write. 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. These are the tables and objects and their sensitivity/vulnerability:

- batteryChargingAdminState
  Setting the battery charging state can be beneficial for an operator for various reasons such as charging batteries when the price of electricity is low. However, setting the charging state can e.g. be used by an attacker to discharge batteries of devices and thereby swiching these devices off if they are powered solely by batteries. In particular, if the batteryAlarmLowCharge and batteryAlarmLowVoltage can also be set, this attack will go unnoticed (i.e. no notifications are sent).

- batteryAlarmLowCharge and batteryAlarmLowVoltage
  These objects set the threshold for an alarm to be raised when the battery charge or voltage falls below the corresponding one of them. An attacker setting one of these alarm values can switch off the alarm by setting it to the 'off' value 0 or modify the alarm behavior by setting it to any other value. The result may be loss of data if the battery runs empty without warning to a receipient expecting such a notification.

- batteryAlarmLowCapacity and batteryAlarmHighCycleCount
  These objects set the threshold for an alarm to be raised when the battery becomes older and less performant than required for stable operation. An attacker setting this alarm value can switch off the alarm by setting it to the 'off' value 0 or modify the alarm behavior by setting it to any other value. This may either lead to a costly replacement of a working battery or too old or too weak batteries are used. The consequence of the latter could e.g. be that a battery cannot provide power long enough between two scheduled charging actions causing the powered device to shut down and potentially loose data.

- batteryAlarmHighTemperature and batteryAlarmLowTemperature
  These objects set thresholds for an alarm to be raised when the battery rises above/falls below them. An attacker setting one of these alarm values can switch off these alarms by setting them to the 'off' value '7fffffff'H or modify the alarm behavior by setting them to any other value. The result may e.g. be an unnecessary shutdown of a device if batteryAlarmHighTemperature is set to too low or damage to the device by too high temperatures if switched off or set to too high values or by damage to the battery when it e.g. is being charged. Batteries can also be damaged e.g. in an attempt to charge them at too low temperatures.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

All potentially sensible or vulnerable objects of this MIB modules are in the batteryTable. In general, there are no serious operational vulnerabilities foreseen in case of an unauthorized read access to this table. However, privacy issues need to be considered. It may be a trade secret of the operator

- how many batteries are installed in a managed node (batteryIndex)
- how old these batteries are (batteryActualCapacity and batteryChargingCycleCount)
- when the next replacement cycle for batteries can be expected (batteryAlarmLowCapacity and batteryAlarmHighCycleCount)
- what battery type and make are used with which firmware version (batteryIdentifier, batteryFirmwareVersion, batteryType, and batteryTechnology)

SNMP versions prior to SNMPv3 did not include adequate security. 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 module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator

responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to GET or SET (change/create/delete) them.

## 6. IANA Considerations

### 6.1. SMI Object Identifier Registration

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

| Descriptor | OBJECT IDENTIFIER value |
|---|---|
| batteryMIB | { mib-2 xxx } |

[NOTE for IANA: Please allocate an object identifier at http://www.iana.org/assignments/smi-numbers for object batteryMIB.]

### 6.2. Battery Technology Registration

Object batteryTechnology defined in Section 4 reports battery technologies. Eighteen values for battery technologies have initially been defined. They are listed in a table in Section 3.2.

For ensuring extensibility of this list, IANA has created a registry for battery technologies at http://www.iana.org/assignments/eman and filled it with the initial list given in Section 3.2.

New assignments of numbers for battery technologies will be administered by IANA through Expert Review ([RFC5226]). Experts must check for sufficient relevance of a battery technology to be added.

[NOTE for IANA: Please create a new registry under http://www.iana.org/assignments/eman for battery types. Please fill the registry with values from the table in Section 3.2]

## 7. Open Issues

### 7.1. Entity MIB augmentation

Alignment with Entity MIB version 4 is need as soon as it becomes stable.

### 7.2. Kind of entity

In section 3.1 we recommend to use a value of powerSupply(6) for object entPhysicalClass, if the entity is a battery. This sections needs to be updates once we have values for entPhysicalClass maintained by IANA. We should then register a new value "battery(xy)" at IANA and replace "powerSupply(6) in this section.

### 7.3. Voltage and temperature per cell?

For lithium-ion batteries it is common to measure voltage not just in total but also per cell. Also temperature per cell is sometimes of interest. Shall we support this? It would require a cell table.

### 7.4. Notifications for removable batteries

PCs and other devices offer battery replacement at runtime. We need to specify events for added a batteries and removed batteries (batteryAddedNotification, batteryAddedNotification). The energy management system should get informed about such events, because they either create a new entry in the battery table or they remove one from it.

### 7.5. Notification for battery charging state changes?

Do we need a notification for battery charging state changes?

### 7.6. Support for ACPI critical battery state?

The ACPI has a ’critical’ battery state. This is when the battery is in a state that it cannot be used anymore and must be charged. We already have a batteryLowNotification. Would we also need a batteryCriticalNotification?

## 8. Acknowledgements

We would like to thank Steven Chew and Bill Mielke for their valuable input.

## 9. References

### 9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "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.

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

## 9.2. Informative References

[I-D.ietf-eman-requirements]
Quittek, J., Chandramouli, M., Winter, R., Dietz, T., and B. Claise, "Requirements for Energy Management", draft-ietf-eman-requirements-09 (work in progress), October 2012.

[I-D.ietf-eman-framework]
Claise, B., Parello, J., Silver, L., Quittek, J., and B. Nordman, "Energy Management Framework", draft-ietf-eman-framework-05 (work in progress), July 2012.

[I-D.ietf-eman-energy-monitoring-mib]
Chandramouli, M., Silver, L., Quittek, J., Dietz, T., and B. Claise, "Power and Energy Monitoring MIB", draft-ietf-eman-energy-monitoring-mib-03 (work in progress), July 2012.

[RFC1628] Case, J., "UPS Management Information Base", RFC 1628, May 1994.

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

[SBS] "Smart Battery Data Specification", Revision 1.1, December 1998.

Authors' Addresses

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

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

Rolf Winter
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-121
Email: Rolf.Winter@neclab.eu

Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-128
Email: Thomas.Dietz@neclab.eu