MIB Modules for Energy Management

draft-quittek-power-mib-00

J. Quittek, R. Winter, T. Dietz, D. Dudkowski

Power Management

- We need means for power management
  - rising energy cost
  - increased awareness of ecological impact of running IT and NW equipment
- Basic objective
  - run networks and attached equipment with minimal amount of energy (and cost)

Essential first step: Power Monitoring

- Monitoring does not directly save energy
  - it rather consumes additional energy
- However, it is needed for
  - investigating power saving potential
  - deriving, implementing, testing and evaluating energy saving policies and measures
  - input to control loop for dynamic power management
  - accounting the total power consumption of a network element, a network, a service, ...

History of this activity

- Kick-off presentation at IETF 75
- Requirements discussed at IETF 76
  - draft-quittek-power-monitoring-requirements-00
- 4 MIB modules submitted for discussion at IETF 77
  - draft-teraoka-powerconsumption-mib
  - draft-sreek-powerconsumption-mib
  - draft-claise-energy-monitoring-mib
  - draft-quittek-power-mib-00
    - straight forward design according to requirements

What needs to be monitored?

- Requirements identified in draft-quittek-power-monitoring-requirements-00
- Three areas of requirements
  - power state monitoring
  - energy consumption monitoring
  - battery monitoring
- Approach: one MIB module for each area
  - independent of each other
  - short, simple, and clear modules

Issue of all modules: Identification

- Problem: identification of monitored unit
- Convenient approach: use Entity MIB (RFC4133)
  - entPhysicalTable for identifying units to be monitored
  - identifying entities by entPhysicalIndex
- Questions
  - Can we assume that the Entity MIB is always available then implementing Energy MIB modules?
    - No, but the overhead of implementing it may be acceptable?
  - Sparse augment or pointer to entPhysicalTable?
    - Do we need to report on remote entities?
      - Cannot be modeled by Entity MIB
    - Requirements draft says yes
    - Example: PoE switch

Power State MIB: Objects

- Current power state table:

```
powerCurrentStateTable
+--powerCurrentStateEntry(1) [entPhysicalIndex]
   +-- r-n EntityStandbyStatus powerCurrentState(1)
```

- Per power state statistics table

```
powerStateTable(2)
+--powerStateEntry(1) [entPhysicalIndex,powerState]
   +-- --- EntityStandbyStatus powerState(1)
   +-- r-n TimeTicks           powerStateTotalTime(2)
   +-- r-n TimeStamp           powerStateLastEnterTime(3)
   +-- r-n SnmpAdminString     powerStateLastEnterReason(4)
   +-- r-n Counter64           powerStateEnterCount(5)
```

Power State MIB: Open Issue

- **How many power states do we need?**
- The Entity State MIB (RFC4268) defines just three states:
  - unknown(1), hotStandby(2)
    coldStandby(3), providingService(4)
- The Advanced Configuration & Power Interface (ACPI) defines several more
  - claise-energy-monitoring-mib lists 12 states:
    - non-operational:
      mechoff(1), softoff(2), hibernate(3), sleep(4), standby(5), ready(6),
    - operational:
      low(7), frugal(8), medium(9), reduced(10), high(11), full(12)
- Alternative: don't use SMI enumeration, but define operational states in a states table

Energy Consumption MIB

- What functionality is needed?
  - reporting actual power
  - accumulated energy consumption
    - In total and per power state
  - reporting time series of actual power values
    - probably required if used for smart meters
    - push: SNMP notifications, IPFIX records
    - pull: table with stored time series
      - like in traceroute / ping / lookup MIBs (RFC 4560)

Energy Consumption Table

- Using textual conventions of Entity Sensor MIB (RFC 3433)

```
energyConsumpTable(1)
+--energyConsumpEntry(1) [entPhysicalIndex]
+-- r-n EntitySensorStatus    energyConsumpSensorOperStatus(1)
+-- r-n Unsigned32            energyConsumpSampleInterval(2)
+-- r-n Unsigned32            energyConsumpNominalSupplyVoltage(3)
+-- r-n Enumeration           energyConsumpElectricSupplyType(4)
+-- r-n EntitySensorValue     energyConsumpTotalEnergy(5)
+-- r-n EntitySensorDataScale energyConsumpEnergyScale(6)
+-- r-n EntitySensorPrecision energyConsumpEnergyPrecision(7)
+-- r-n TimeStamp             energyConsumpDiscontinuityTime(8)
+-- r-n EntitySensorDataScale energyConsumpPowerScale(9)
+-- r-n EntitySensorPrecision energyConsumpPowerPrecision(10)
+-- r-n EntitySensorValue     energyConsumpRealPower(11)
+-- r-n EntitySensorValue     energyConsumpPeakRealPower(12)
+-- r-n EntitySensorValue     energyConsumpReactivePower(13)
+-- r-n EntitySensorValue     energyConsumpApparentPower(14)
+-- r-n EntitySensorValue     energyConsumpPhaseAngle(15)
+-- r-n EntitySensorPrecision energyConsumpPhaseAnglePrecision(16)
```

Energy Consumption Per State Table

- Using data scale and precision from energy consumption table

```
energyConsumpPSTable(2)
+--energyConsumpPSEntry(1) [entPhysicalIndex,powerState]
   +-- r-n EntitySensorValue energyConsumpPSTotalEnergy(1)
```

Notifications

- Power State MIB
  - powerStateChangeEvent
    - powerStateLastEnterReason
- Battery MIB
  - batteryLowNotification
    - batteryCurrentChargePercentage
    - batteryCurrentVoltage
  - batteryAgingNotification
    - batteryRemainingCapacity
    - batteryChargingCycleCount

Battery MIB

- First Shot
- Not covered by the other drafts
- Still homework to be done
  - alignment with UPS MIB
  - comparison with existing private MIB modules

Battery Table

```
batteryTable(1)
+--batteryEntry(1) [entPhysicalIndex]
   +-- r-n Enumeration batteryType(1)
   +-- r-n Enumeration batteryTechnology(2)
   +-- r-n Unsigned32  batteryNominalVoltage(3)
   +-- r-n Unsigned32  batteryNumberOfCells(4)
   +-- r-n Unsigned32  batteryNominalCapacity(5)
   +-- r-n Unsigned32  batteryRemainingCapacity(6)
   +-- r-n Counter32   batteryChargingCycleCount(7)
   +-- r-n DateAndTime batteryLastChargingCycleTime(8)
   +-- r-n Enumeration batteryState(9)
   +-- r-n Unsigned32  batteryCurrentCharge(10)
   +-- r-n Unsigned32  batteryCurrentChargePercentage(11)
   +-- r-n Unsigned32  batteryCurrentVoltage(12)
   +-- r-n Integer32   batteryCurrentCurrent(13)
   +-- r-n Unsigned32  batteryLowAlarmPercentage(14)
   +-- r-n Unsigned32  batteryLowAlarmVoltage(15)
   +-- r-n Unsigned32  batteryReplacementAlarmCapacity(16)
   +-- r-n Unsigned32  batteryReplacementAlarmCycles(17)
```

Next Steps

- Merge with draft-claise-energy-monitoring-mib
  - use pmPowerUsageCaliber
  - use table for storing time series of energy measurements (optional)
  - compromise by using flexible operational states
  - more issues to be solved, but no show stoppers
    - read-write vs. read-only, etc.
- Propose merged draft to become OPSAREA WG work item

Outlook

- Shall we include smart meters at home in our scope?
- Shouldn’t we start caring about other components of energy management?
  - configuration, scheduling, control, ...