



December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA

# Platform Level Data Model (PLDM) Technical Overview

Hemal V. Shah  
Broadcom Corporation



# Disclaimer



The DMTF was formed to lead the development, adoption and unification of management standards and initiatives for desktop, enterprise and internet environments

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change. The Standard Specifications remain the normative reference for all information.
- For additional information, see the Distributed Management Task Force (DMTF) Web site.  
<http://www.dmtf.org>.



# Agenda

- PLDM Overview
- PLDM for SMBIOS
  - Data structures
  - Commands
- PLDM for BIOS Control/Config
  - Data Structures
  - Commands
- PLDM for sensors
  - Data Structures
  - PLDM for Numeric sensors
  - PLDM for State and Composite state sensors

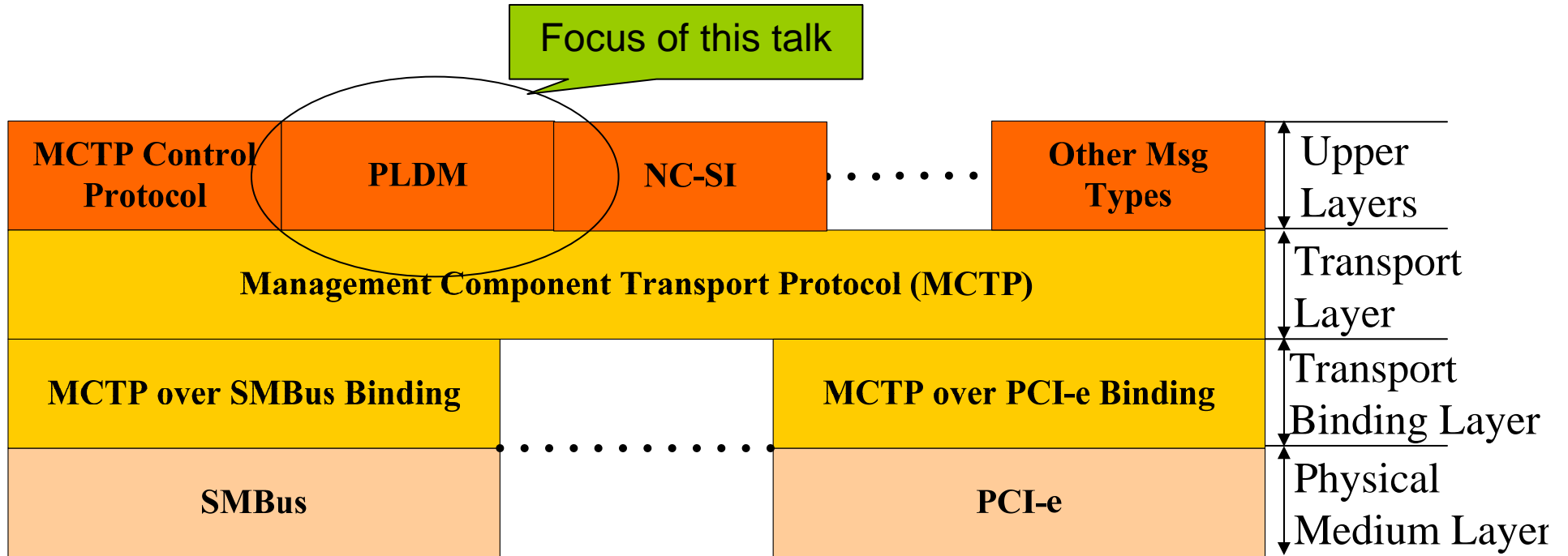


# PMCI Workgroup

- **Platform Management Component Intercommunications**
  - DMTF Pre-OS WG sub-team formed August 2006
  - 17 Member Companies in workgroup (as of 4/11/07, list in Backup)
  - Active participants include AMD, Ample, Avocent, Broadcom, Dell, HP, IBM, Intel, NVIDIA, and SMSC
  - Co-chairs: Hemal Shah, Broadcom; Tom Slaight, Intel
- Scope: “Inside the box” communication and functional interfaces between components within the platform management subsystem
  - Mgmt Controller (MC) to Mgmt Controller
  - Mgmt Controller to Intelligent Management Device
  - Mgmt Controller to Network Controller
  - FW / SW to Mgmt Controller
- Builds on and captures learning's from SMBIOS, ASF, & NC-SI
  - Plus leverages SMBus, IPMI, PCIe, and other industry technologies

**PMCI technologies and interfaces are complementary to DMTF CIM Profiles and remote access protocols**

# PMCI Protocol Stack



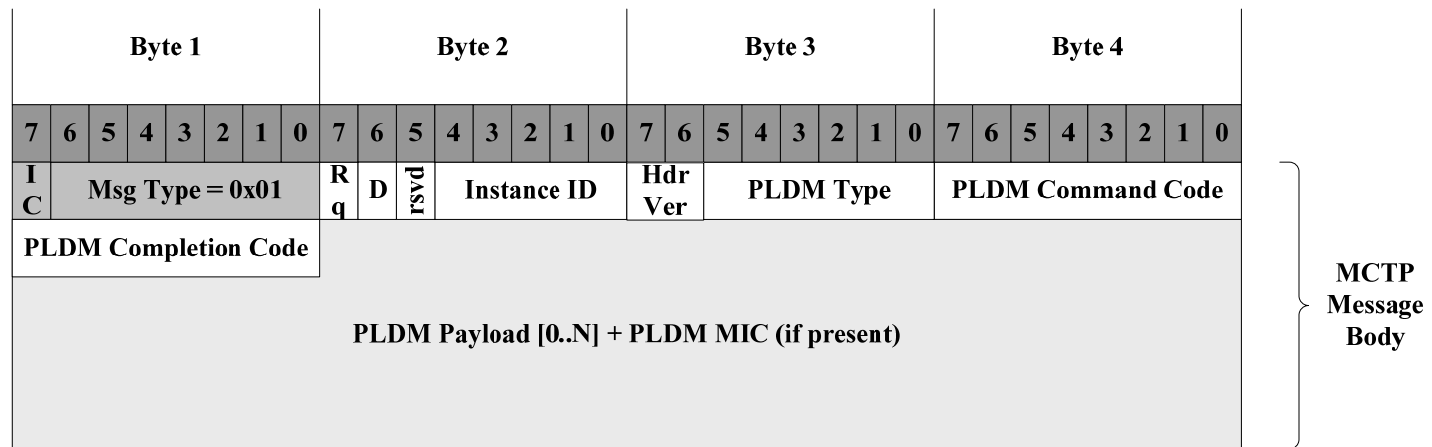


# Platform Level Data Model (PLDM) – What is it?

- Designed to be an effective data/control source for mapping under CIM
- Defines data structs/cmds that abstract platform mgmt subsystem components
- Provides efficient access to
  - Low-level platform inventory
  - BIOS control and configuration data transfer
  - Platform monitoring and control
  - Eventing data
  - Data/parameters transfer functions....
- Supports a sub-type to distinguish various types of PLDM messages
  - Allow messages to be grouped based on the functions
  - Allows the discovery of the functionality supported
  - Examples:
    - PLDM for SMBIOS data transfer
    - PLDM for BIOS control and configuration
    - PLDM for sensors/fans/power supplies
- All the PLDM messages will be transferable using baseline transmission unit
  - Larger message sizes may be negotiated for specific message types



# PLDM over MCTP Binding – How does PLDM map over MCTP?



Field Name	Field Size	Description
Rq	1 bit	Request bit.
D	1 bit	Datagram bit.
rsvd	1 bit	Reserved
Instance ID	5 bits	Used to identify the instances of a PLDM request.
Hdr Ver	2 bits	Identifies the header format of PLDM over MCTP messages.
PLDM Type	6 bits	The PLDM Type field identifies the type of PLDM that is being used.
PLDM Command Code	8 bits	Identifies the type of operation the message (per PLDM type) is requesting.
PLDM Completion Code	8 bits	The PLDM Completion Code field provides the status of the operation
PLDM Message Payload	Variable	Zero or more bytes of PLDM message payload that is specific to a particular payload type, PLDM type, command code, and/or completion code.
PLDM MIC	TBD	PLDM Message Integrity Check – PLDM Type & command/completion specific



# PLDM for System Management BIOS (SMBIOS)

- **SMBIOS**
  - BIOS extensions to provide platform info
  - Provides key platform asset information
    - Platform info, BIOS info, Processor info, Memory info, etc.
- **PLDM for SMBIOS data transfer**
  - Enables SMBIOS data transfer between BIOS/MC
- **MC can deliver SMBIOS info using CIM**
  - Provide key platform asset management info
  - Can be used for system health monitoring
  - Access to system event log



# SMBIOS Structure Table

- Contains one or more SMBIOS structures
  - Each structure begins with (type, length, and handle) header
  - Structures are not ordered
  - Searching for a specific structure requires walking through & parsing the SMBIOS table
- Low-level data model used to store platform asset info
- Located in system memory



# PLDM for SMBIOS Data Transfer Requirements (1 of 2)

- SMBIOS Structure Table Entry Point
  - Contains SMBIOS table versioning info, checksum info, table length, max struct size...
  - PLDM will define commands to obtain SMBIOS table meta-data information
- SMBIOS structures format
  - SMBIOS structure definition is type specific
  - Handles are assigned by the BIOS
  - Handles can change if the sys config changes
  - PLDM will preserve SMBIOS structure format



# PLDM for SMBIOS Data Transfer Requirements (2 of 2)

- SMBIOS table transfer
  - SMBIOS table can be large
  - SMBIOS table most likely won't fit in a single baseline transmission unit
  - PLDM will define cmds that allow the transfer of entire SMBIOS table
    - Single request/response
    - Multiple requests/multiple responses
  - PLDM will support both Pull/Push models for the SMBIOS table transfer
  - PLDM will define data integrity check to protect SMBIOS data transfer
- SMBIOS structure data transfer
  - PLDM will allow SMBIOS structure data transfer by type or by handle
  - A SMBIOS structure may not fit in a single baseline transmission unit
  - PLDM will define cmds that allow transfer of the entire SMBIOS struct
    - Either single request/response
    - Multiple requests/multiple responses
- PLDM will not define commands to randomly access elements within a SMBIOS structure

# Data Integrity

- Typically, SMBIOS table is 2K-4KB in size
- PLDM will protect SMBIOS data transfer
- Data integrity is provided for the entire table or structures transfers
  - For data transfer using multiple messages, the data integrity is provided for the whole transfer
  - Data integrity algorithm used: CRC-32 – same as the one used by IEEE 802.3



# PLDM Commands for SMBIOS Data Transfer

- Get SMBIOS Table Metadata Information
- Get SMBIOS Table
- Set SMBIOS Table
- Get SMBIOS Structure By Type
- Get SMBIOS Structure By Handle



# PLDM for BIOS Control/Configuration

- Complementary to Boot control/BIOS Management profiles
- Control/configuration of interest to PLDM
  - BIOS settings and attributes transfer
  - Boot configuration settings and ordering
- MC can support BIOS management using CIM



# PLDM Aspects of Boot Control Functions

- Boot control information exchanged between BIOS/MC
- Boot control profile methods:
  - CIM\_BootService.CreateBootConfigSetting()
  - CIM\_BootService.ApplyBootConfigSetting()
  - CIM\_BootConfigSetting.ChangeBootOrder()
- PLDM Aspects
  - Boot configuration setting
  - Getting or setting of boot options/order
  - Boot source setting
    - Definition will support fixed, removable & redirected media
  - PLDM will define
    - Representation of boot config record
    - Messages for performing operations on the boot configuration record

# BIOS Modeling

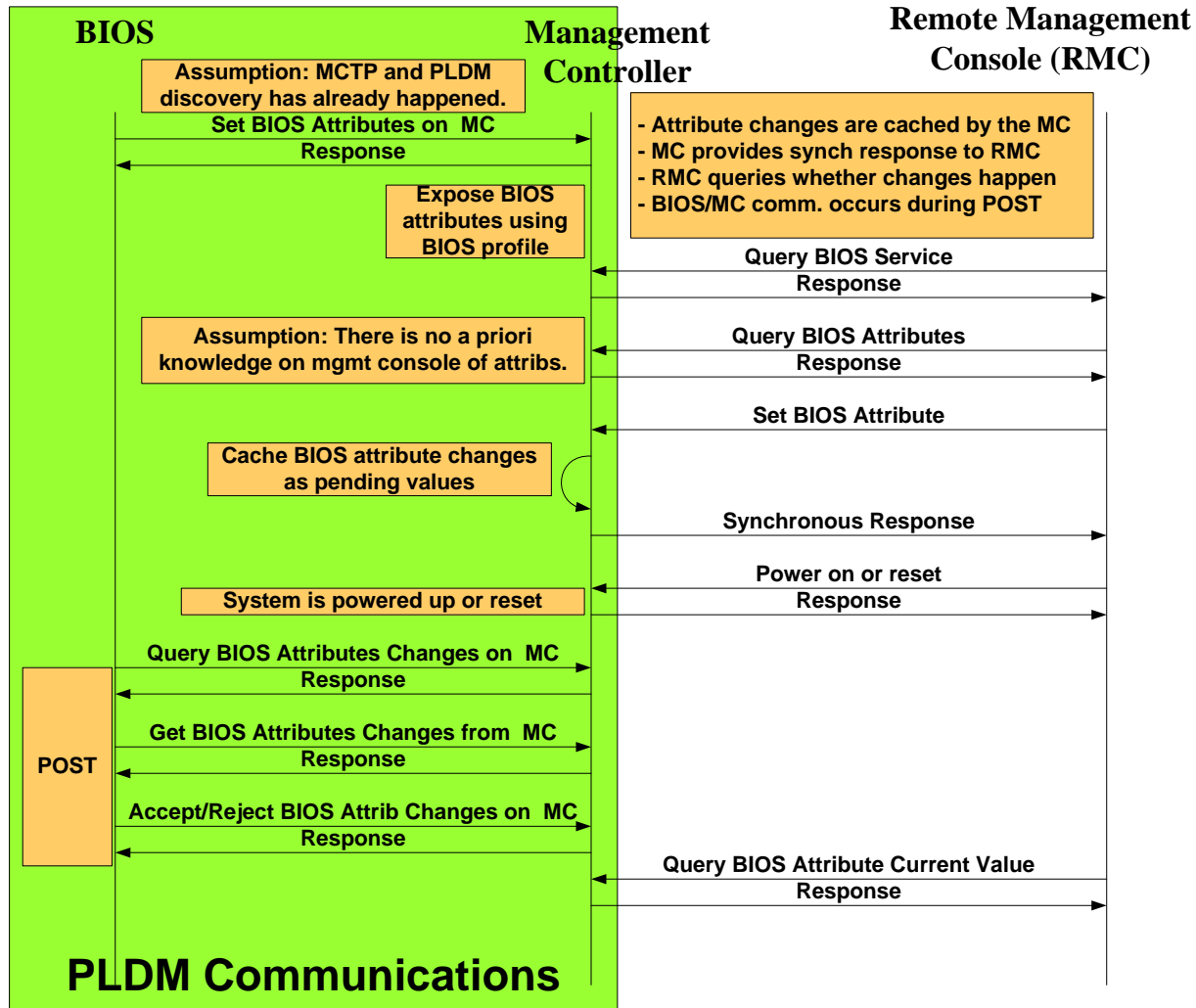
- BIOS profile is based on the attribute instance approach
- CIM classes are defined at the BIOS attribute level
  - One instance per BIOS attribute
- BIOSService affects the BIOS attrib values
- BIOSService methods
  - SetBIOSAttribute()
  - SetBIOSAttributeEmbeddedInstance( )
  - RestoreBIOSDefaults()
  - ...
- Types of BIOS Attributes
  - BIOSEnumeration, BIOSInteger, BIOSPassword, BIOSString
- Each BIOS attribute has
  - Current value(s), Default value(s), Pending value(s)



# BIOS Attribute Update Models

- Immediate update model:
  - MC acting as a pass-through device
  - Immediate update with BIOS active
  - MC responds to console after BIOS processes updates
- Deferred update model:
  - MC acting as a cache of BIOS settings
  - MC caches changes initiated remotely
  - MC provides changes to BIOS during next boot
- Direct update model:
  - MC updates BIOS settings directly without communicating with the BIOS
  - This model is out of scope for PLDM 1.0

# BIOS/MC Communication Example





# PLDM aspects related to BIOS Attributes Transfer

- BIOS attribute related data structures definitions
- Define messages for BIOS attribute transfers
- Notifications to BIOS related to data transfers
- Query ops to check whether BIOS is active
- Discovery of well known BIOS attributes
- Ordering/dependency among attribute data transfers will not be covered by PLDM
  - PLDM commands simply transfer BIOS attributes changes or the entire BIOS table
  - PLDM does not track or control the order in which BIOS or MC applies changes
- Aggregation of BIOS attributes data transfer is not handled at PLDM



# Representation of Strings used for BIOS

- BIOS Attribute names are represented by strings
- BIOS Attribute values are represented as strings
- However, transferring strings all the time can lead to inefficiency
- Well known strings can be identified using a table based approach
- Tables are used for pre-configured strings
  - Idea is to communicate strings information on initialization and updates
  - For the rest of the operations or messages, a handle can be used to represent the pre-configured strings
- An Attribute name handle refers to a BIOS attribute name
  - 16-bit handle seems to be sufficient
- An Attribute Value Handle refers to a BIOS attribute value string
  - Attribute values are common (not as a set of handles per attribute)
  - 16-bit handle seems to be sufficient



# BIOS String, Attribute, and Value Tables

## BIOS String Table

BIOS String Handle	BIOS String
Handle 1	String 1
Handle 2	String 2
...	...

- Useful for efficient BIOS attributes data transfer between BIOS/MC
- Not exposed to remote mgmt console
- Provide common mechanism for BIOS/MC to communicate mapping

## BIOS Attribute Table

Attribute Handle	Attribute Name Handle	Attribute Type	Type Specific Fields
Attrib 1 Handle	Attrib1 Name Handle	Attrib 1 Type	....
Attrib 2 Handle	Attrib2 Name Handle	Attrib 2 Type	...
...	...	...	...

## BIOS Attribute Value Table

Attribute Handle	Attribute Type	Type Specific Value(s)
Attrib 1 Handle	Attrib 1 Type	...
Attrib 2 Handle	Attrib 2 Type	...
...	...	...

# An Example of BIOS Tables

## BIOS String Table

BIOS String Handle	BIOS String
1	“Enabled”
2	“On”
3	“Off”
...	...
20	“NumLock LED”
21	“USB Emulation”
...	...

## BIOS Attribute Value Table

Attribute Handle	Attribute Type	Type Specific Current Value(s)
1	0(BIOSEnum)	0 (index into the array of possible values)
2	0(BIOSEnum)	0 (index into the array of possible values)
...	...	...

## BIOS Attribute Table

Attribute Handle	Attribute Name Handle	Attribute Type	Type Specific Possible Values	Type Specific Default Value(s)
1	20 (NumLockLED)	0(BIOSEnum)	{2 (On), 3 (Off)}	0 (index into the array of possible values)
2	21 (USB Emulation)	0(BIOSEnum)	{1 (Enabled), 3 (Off)}	0 (index into the array of the possible values)
...	...	...	...	...



# PLDM Representation of BIOS Tables

## BIOS String Table

byte	type	Field
0-1	uint16	BIOS String 1 Handle
2-3	uint16	BIOS String 1 Length
Variable		BIOS String 1
...	uint16	BIOS String 2 Handle
...	uint16	BIOS String 2 Length
...		BIOS String 2
...	...	...
L:L+3	uint32	BIOSStringTableIntegrityChecksum (CRC-32 - same as the one used by IEEE 802.3)

## BIOS Attribute Value Table

byte	type	Field
0-1	uint16	Attrib 1 Handle
2	enum8	Attrib 1 Type
Variable		Attrib 1 Type specific fields
...	...	...
L:L+3	uint32	BIOSAttributeTableIntegrityChecksum (CRC-32 - same as the one used by IEEE 802.3)

## BIOS Attribute Table

byte	type	Field
0-1	uint16	Attrib 1 Handle
2-3	uint16	Attrib 1 Name Handle
4	enum8	Attrib 1 Type
Variable		Attrib 1 Type specific fields (see below)
	uint16	Attrib 2 Handle
	uint16	Attrib 2 Name Handle
	enum8	Attrib 2 Type
		Attrib 2 Type specific fields (see below)
...	...	...
L:L+3	uint32	BIOSAttributeTableIntegrityChecksum (CRC-32 - same as the one used by IEEE 802.3)

# PLDM for Sensors

- Complementary to Sensors/Fan/Power Supply profiles
- Provides Platform communication model for monitoring
  - Sensors readings
  - Threshold settings
  - Event status and event messages



# PLDM Numeric Sensors

- Used to represent a numeric reading
  - Reading is represented as an integer
- Sensor commands do not identify units/scaling
  - MAP performs the conversion
- Numeric sensor thresholds
  - Thresholds is associated with a severity
    - Warning, critical, and fatal
  - Readable and settable thresholds
- Numeric sensor status
  - Present status: Based on comparing reading against thresholds
  - Event status: Based on transitions btwn different monitored states
    - Auto-arm sensors auto update event status based on state transitions
    - Manual-rearm sensors auto update event status on detecting a worsening transition

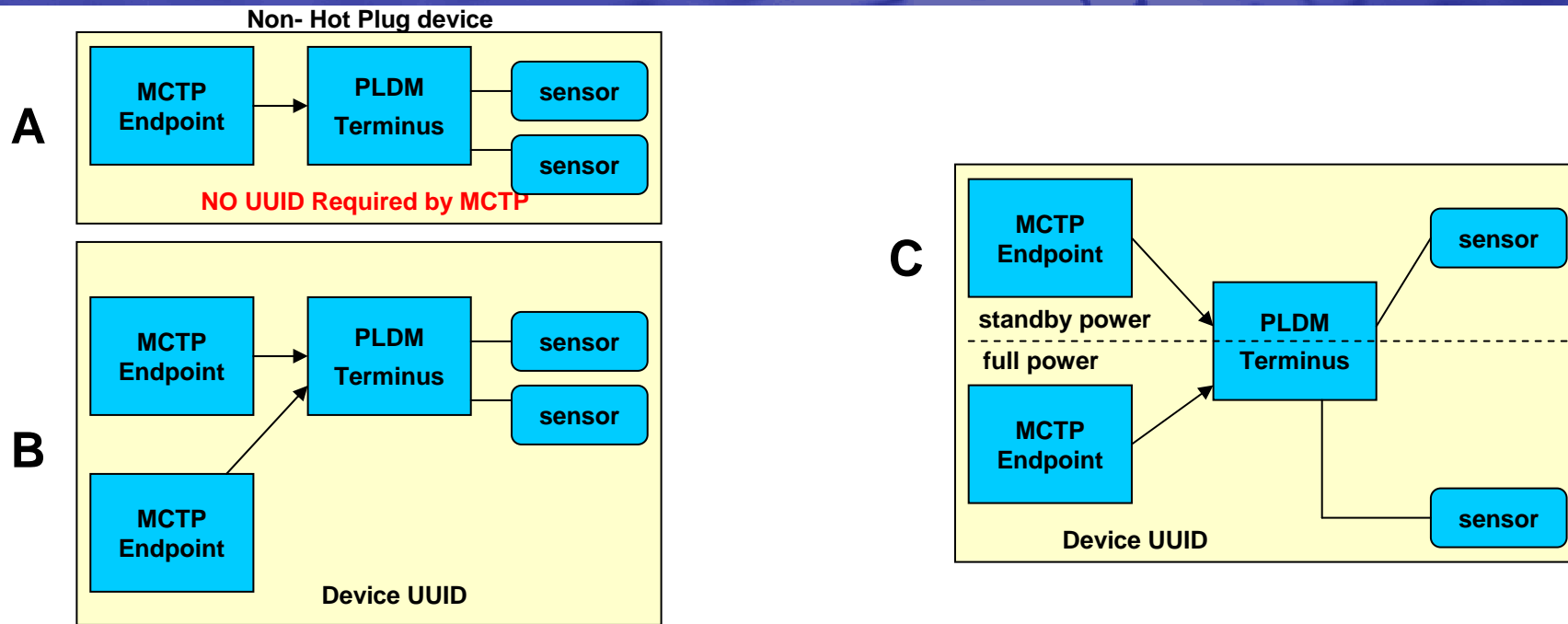
# State sensors

- Represents a particular state from a set of state information
  - The set can be small, e.g. on | off, present | absent
  - Or lengthy, e.g. "Other", "Unknown", "Running/Full Power", "Warning", "In Test", "Not Applicable", "Power Off", "Off Line", "Off Duty", "Degraded", "Not Installed", "Install Error", "Power Save - Unknown", "Power Save - Low Power Mode", "Power Save - Standby", "Power Cycle", "Power Save - Warning", "Paused", "Not Ready", "Not Configured", "Quiesced"
- States are typically exclusive & are accessed in CIM as enumerations
- Returning the value for an enumeration may be considered to be a special case of returning a numeric sensor value
- State monitoring should also be able to generate platform events
  - Similar to numeric sensors for maintaining 'threshold crossing' status and generating threshold crossing events
- Composite state sensors
  - Enables multiple sets of state information to be returned in a single reading
  - Individual state sensor typically map to an individual instance of CIM\_Sensor
  - 'mutually exclusive' bits were used when enumerated info was returned
  - Same number of bytes always returned regardless of which sensors are enabled
  - Individual sensors are indexed under Sensor Access ID number, starting from 0.
    - "Sensor Number" changes to "Sensor Access Number" + "Sensor Offset"

# Sensor Commands

- `GetStateSensorReadings`
- `SetStateSensorEnables`
- `GetNumericSensorReading`
- `SetNumericSensorEnable`
- `SetNumericSensorThresholds`
- `GetNumericSensorThresholds`
- `RestoreNumericSensorThresholds`
- `SetNumericSensorHysteresis`
- `GetNumericSensorHysteresis`

# Sensor Addressing



- Use device-relative identification of sensors
  - Using a “SensorID” field
- Static SMBus/I2C Device
  - Assign a system-relative device ID Or
  - Bus Owner maintains the device ID
- Identify hot-plug devices by UUID

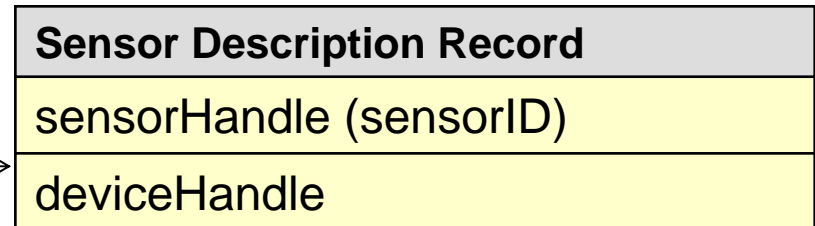
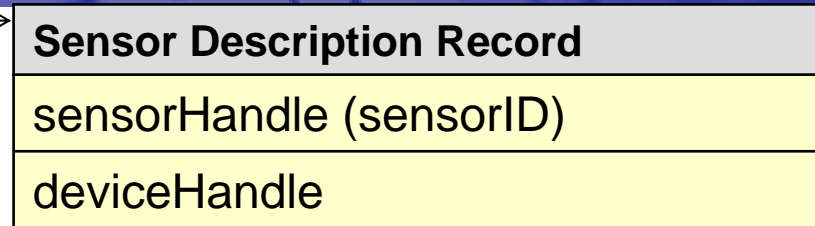
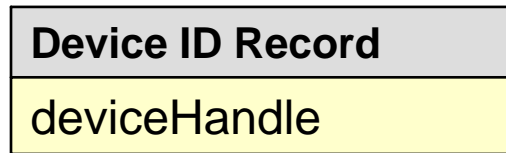


# Platform Data Record (PDR)

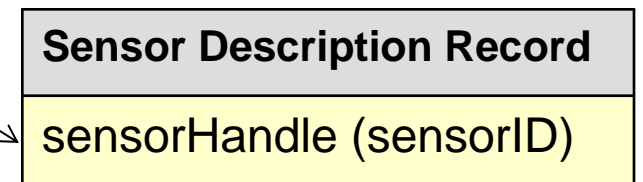
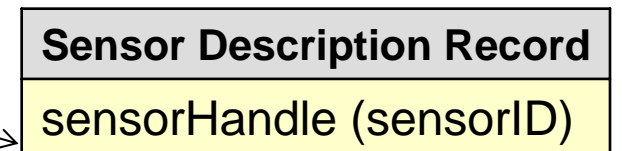
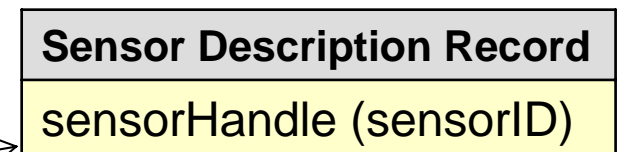
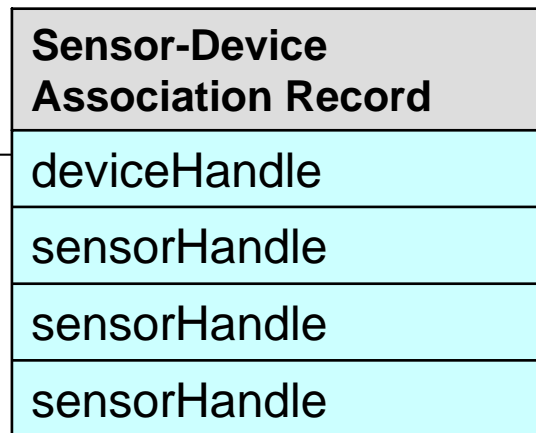
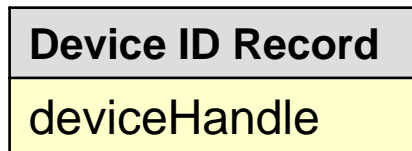
- Collections of semantic and association information about platform
  - Semantic information about sensors
  - Information about the associations between sensors & monitored entities
  - Other platform associations or capabilities
- PDR Types
  - PLDM Numeric Sensor Record
  - PLDM State Sensor Record
  - Sensor / Entity Association
  - ...
- PDR Commands
  - GetPDRRepositoryInfo
  - GetPDR
  - FindPDR
  - GetSensorPDR...

# Record Associations

Direct 'internal' association



Indirect 'external' association



# Event Information

- An event message should only carry ‘transient’ info E.g.:
  - Value (reading) that triggered the event
  - Present state for the event
  - Previous state (state that the event was entered from)
- Plus information that can identify the event source
  - Device ID: Can be inferred from EID for MCTP devices
  - Sensor ID
  - Sensor Offset
- Same format for both Numeric/State-only Sensors
- Sensor Events do not carry semantic data
- Virtual “Event Only” sensors
  - Sensors that only existed as a convention for delivering event data



# Summary

- PLDM is designed to be an effective data/control source for mapping under CIM
- PLDM is complementary to CIM profiles
- Current PLDM definition work in progress
  - PLDM for SMBIOS
  - PLDM for BIOS
  - PLDM for sensors
- More PLDM definitions are planned in the future