Advancing PLDM:

Structured Sensors, Scalable Modeling, and the Server Host SoC PLDM Specification

Justin King, AMD

Samer El-Haj-Mahmoud, Arm

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.
- This information is a summary of the information that will appear in the specifications. See the specifications for further details

Agenda

- PLDM for Host SoCs (CPUs)
- Scalable Sensors/Effecters
- Scalable Entity Association
- Structured Attributes
- Putting it all together – the PLDM Host SoC Model

PLDM for Host SoCs

- Multiple Host SoC (CPU) vendors have converged on PLDM (particularly PLDM Type 2) as their out-of-band management protocol, but are making conflicting modeling decisions
- **Goal:** Publish a DMTF Informational Document (DSP2070) that provides guidance on how to build a PLDM model for Host SoCs
- PMCI has published three similar Informational Documents:
  - DSP2054 PLDM NIC Modeling
  - DSP2061 PLDM Accelerator Modeling
  - DSP2067 PDLM CXL Memory Modeling

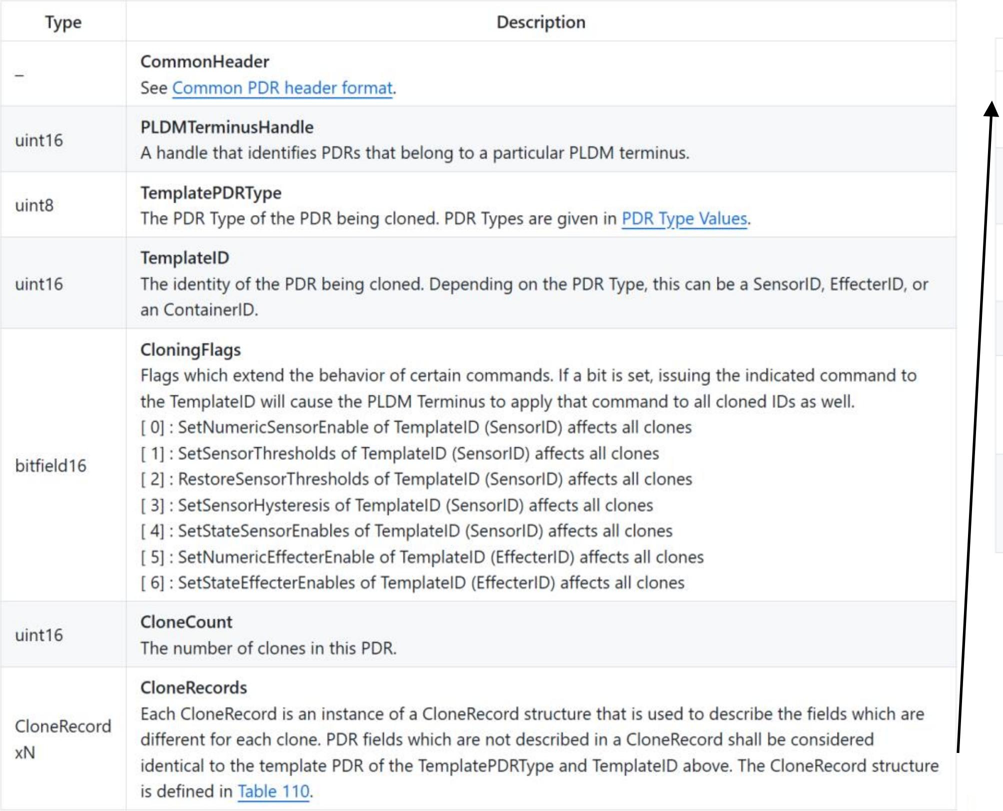
Scaling Challenges

- Modern SoCs contain numerous instances of replicated components – especially CPU cores
- Replicated components → Replicated sensors and effecters
- Example: SoC with 256 processor cores
  - Thermal sensor per core (degrees C)
  - Power sensor per core (milliwatts)
  - → 512 numeric sensors
  - → 512 nearly-identical Numeric Sensor PDRs

PDR Cloning for Scalable Sensors/Effecters

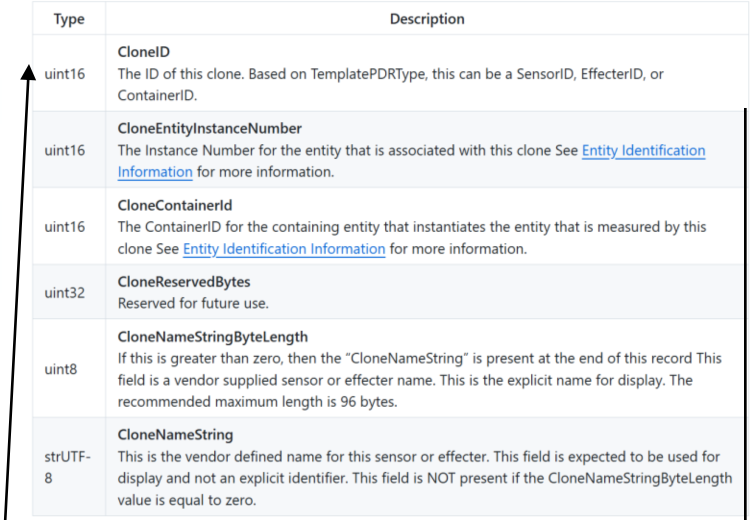
- Reduce the number of PDRs by taking an existing PDR as a template and create several clones from that template.
- Each clone has a new sensor ID or effecter ID, entity Instance, etc but otherwise inherits all other content from the template PDR.
- Reduce the number of calls to enablement commands such as SetNumericSensorEnable by allowing a single call, using the cloned template Id, to enable all clones at once.

PLDM Clone PDR



Type	Description
–	**CommonHeader** See Common PDR header format.
uint16	**PLDMTerminusHandle** A handle that identifies PDRs that belong to a particular PLDM terminus.
uint8	**TemplatePDRType** The PDR Type of the PDR being cloned. PDR Types are given in PDR Type Values.
uint16	**TemplateID** The identity of the PDR being cloned. Depending on the PDR Type, this can be a SensorID, EffecterID, or an ContainerID.
bitfield16	**CloningFlags** Flags which extend the behavior of certain commands. If a bit is set, issuing the indicated command to the TemplateID will cause the PLDM Terminus to apply that command to all cloned IDs as well. [0] : SetNumericSensorEnable of TemplateID (SensorID) affects all clones [1] : SetSensorThresholds of TemplateID (SensorID) affects all clones [2] : RestoreSensorThresholds of TemplateID (SensorID) affects all clones [3] : SetSensorHysteresis of TemplateID (SensorID) affects all clones [4] : SetStateSensorEnables of TemplateID (SensorID) affects all clones [5] : SetNumericEffecterEnable of TemplateID (EffecterID) affects all clones [6] : SetStateEffecterEnables of TemplateID (EffecterID) affects all clones
uint16	**CloneCount** The number of clones in this PDR.
CloneRecord xN	**CloneRecords** Each CloneRecord is an instance of a CloneRecord structure that is used to describe the fields which are different for each clone. PDR fields which are not described in a CloneRecord shall be considered identical to the template PDR of the TemplatePDRType and TemplateID above. The CloneRecord structure is defined in Table 110.

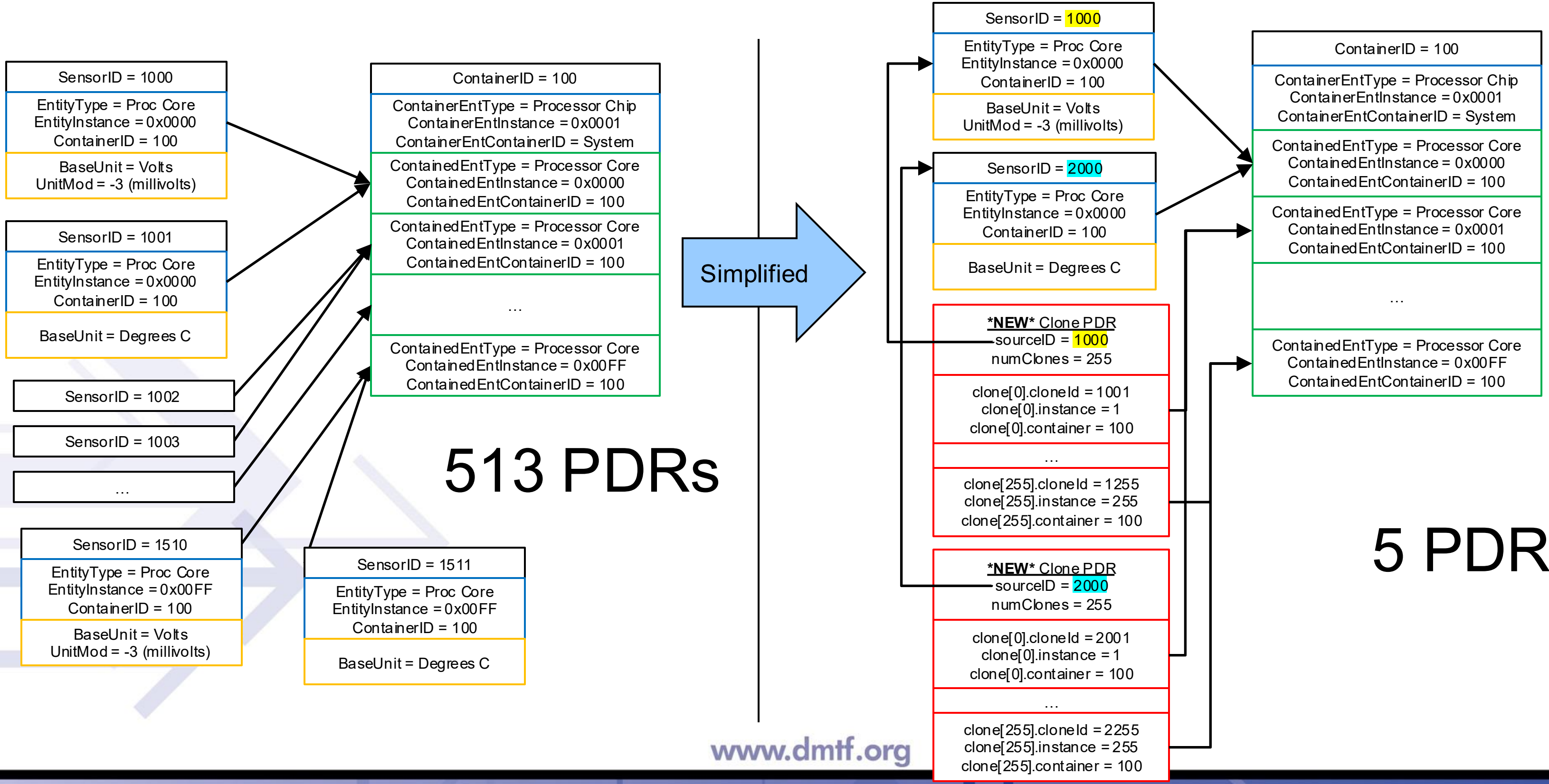
Table 110 — CloneRecord format



Type	Description
uint16	**CloneID** The ID of this clone. Based on TemplatePDRType, this can be a SensorID, EffecterID, or ContainerID.
uint16	**CloneEntityInstanceNumber** The Instance Number for the entity that is associated with this clone See Entity Identification Information for more information.
uint16	**CloneContainerId** The ContainerID for the containing entity that instantiates the entity that is measured by this clone See Entity Identification Information for more information.
uint32	**CloneReservedBytes** Reserved for future use.
uint8	**CloneNameStringByteLength** If this is greater than zero, then the "CloneNameString" is present at the end of this record This field is a vendor supplied sensor or effecter name. This is the explicit name for display. The recommended maximum length is 96 bytes.
strUTF-8	**CloneNameString** This is the vendor defined name for this sensor or effecter. This field is expected to be used for display and not an explicit identifier. This field is NOT present if the CloneNameStringByteLength value is equal to zero.

Note: cloneEntityType is implicit/cloned from TemplateID PDR

Clone PDR – Host SoC w/ 256 Cores



SensorID = 1000

EntityType = Proc Core
EntityInstance = 0x0000
ContainerID = 100

BaseUnit = Volts
UnitMod = -3 (millivolts)

SensorID = 1001

EntityType = Proc Core
EntityInstance = 0x0000
ContainerID = 100

BaseUnit = Degrees C

SensorID = 1002

SensorID = 1003

…

SensorID = 1510

EntityType = Proc Core
EntityInstance = 0x00FF
ContainerID = 100

BaseUnit = Volts
UnitMod = -3 (millivolts)

SensorID = 1511

EntityType = Proc Core
EntityInstance = 0x00FF
ContainerID = 100

BaseUnit = Degrees C

ContainerID = 100

ContainerEntType = Processor Chip
ContainerEntInstance = 0x0001
ContainerEntContainerID = System

ContainedEntType = Processor Core
ContainedEntInstance = 0x0000
ContainedEntContainerID = 100

ContainedEntType = Processor Core
ContainedEntInstance = 0x0001
ContainedEntContainerID = 100

…

ContainedEntType = Processor Core
ContainedEntInstance = 0x00FF
ContainedEntContainerID = 100

513 PDRs

Simplified

SensorID = 1000

EntityType = Proc Core
EntityInstance = 0x0000
ContainerID = 100

BaseUnit = Volts
UnitMod = -3 (millivolts)

SensorID = 2000

EntityType = Proc Core
EntityInstance = 0x0000
ContainerID = 100

BaseUnit = Degrees C

***NEW* Clone PDR**
sourceID = 1000
numClones = 255

clone[0].cloneId = 1001
clone[0].instance = 1
clone[0].container = 100

…

clone[255].cloneId = 1255
clone[255].instance = 255
clone[255].container = 100

***NEW* Clone PDR**
sourceID = 2000
numClones = 255

clone[0].cloneId = 2001
clone[0].instance = 1
clone[0].container = 100

…

clone[255].cloneId = 2255
clone[255].instance = 255
clone[255].container = 100

ContainerID = 100

ContainerEntType = Processor Chip
ContainerEntInstance = 0x0001
ContainerEntContainerID = System

ContainedEntType = Processor Core
ContainedEntInstance = 0x0000
ContainedEntContainerID = 100

ContainedEntType = Processor Core
ContainedEntInstance = 0x0001
ContainedEntContainerID = 100

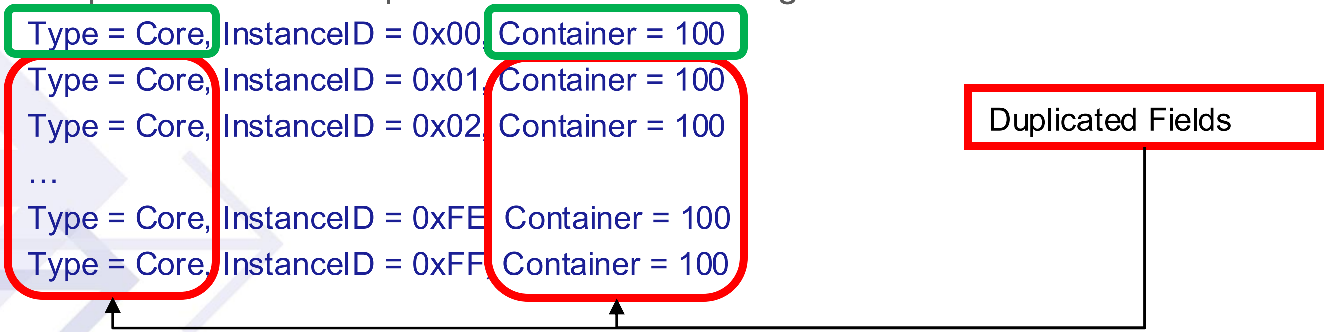
…

ContainedEntType = Processor Core
ContainedEntInstance = 0x00FF
ContainedEntContainerID = 100

5 PDRs

Large/Scaled Entity Association PDRs

- Entity Association Records/PDRs are limited and inefficient in scalable environments
- Limit: 256 children/PDR (Can use linked EA PDRs for bigger container)
- Inefficient: duplicated child entities == duplicated fields
  - Example: SoC has 256 processor cores in a single container. Entries:

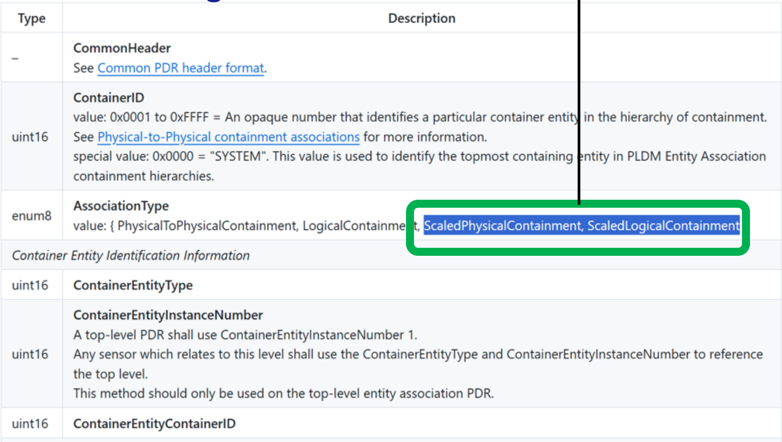


Type = Core, InstanceID = 0x00, Container = 100
Type = Core, InstanceID = 0x01, Container = 100
Type = Core, InstanceID = 0x02, Container = 100
…
Type = Core, InstanceID = 0xFE, Container = 100
Type = Core, InstanceID = 0xFF, Container = 100

  - Duplicated data consumes **65.6%** of a max-size Entity Association PDR (1020/1556 bytes)

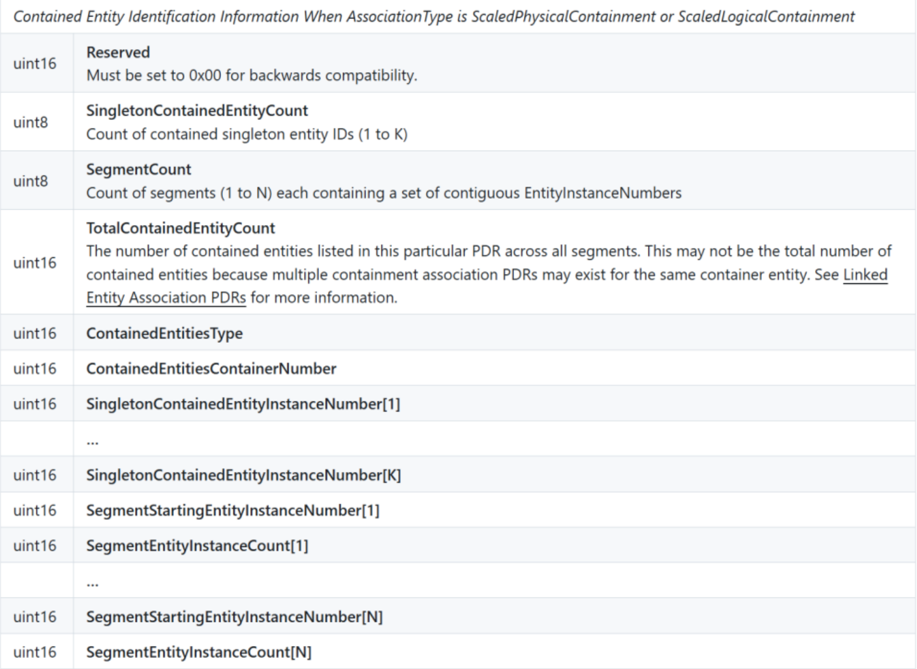
New Scalable Entity Association PDR

Existing EA PDR Front Matter



Type	Description
–	**CommonHeader** See Common PDR header format.
uint16	**ContainerID** value: 0x0001 to 0xFFFF = An opaque number that identifies a particular container entity in the hierarchy of containment. See Physical-to-Physical containment associations for more information. special value: 0x0000 = "SYSTEM". This value is used to identify the topmost containing entity in PLDM Entity Association containment hierarchies.
enum8	**AssociationType** value: { PhysicalToPhysicalContainment, LogicalContainment, ScaledPhysicalContainment, ScaledLogicalContainment
Container Entity Identification Information	
uint16	**ContainerEntityType**
uint16	**ContainerEntityInstanceNumber** A top-level PDR shall use ContainerEntityInstanceNumber 1. Any sensor which relates to this level shall use the ContainerEntityType and ContainerEntityInstanceNumber to reference the top level. This method should only be used on the top-level entity association PDR.
uint16	**ContainerEntityContainerID**

New Child Entity Section (ScaledPhysical, ScaledLogical)



Contained Entity Identification Information When AssociationType is ScaledPhysicalContainment or ScaledLogicalContainment	
uint16	**Reserved** Must be set to 0x00 for backwards compatibility.
uint8	**SingletonContainedEntityCount** Count of contained singleton entity IDs (1 to K)
uint8	**SegmentCount** Count of segments (1 to N) each containing a set of contiguous EntityInstanceNumbers
uint16	**TotalContainedEntityCount** The number of contained entities listed in this particular PDR across all segments. This may not be the total number of contained entities because multiple containment association PDRs may exist for the same container entity. See Linked Entity Association PDRs for more information.
uint16	**ContainedEntitiesType**
uint16	**ContainedEntitiesContainerNumber**
uint16	**SingletonContainedEntityInstanceNumber[1]**
	...
uint16	**SingletonContainedEntityInstanceNumber[K]**
uint16	**SegmentStartingEntityInstanceNumber[1]**
uint16	**SegmentEntityInstanceCount[1]**
	...
uint16	**SegmentStartingEntityInstanceNumber[N]**
uint16	**SegmentEntityInstanceCount[N]**

Atomically accessible attributes

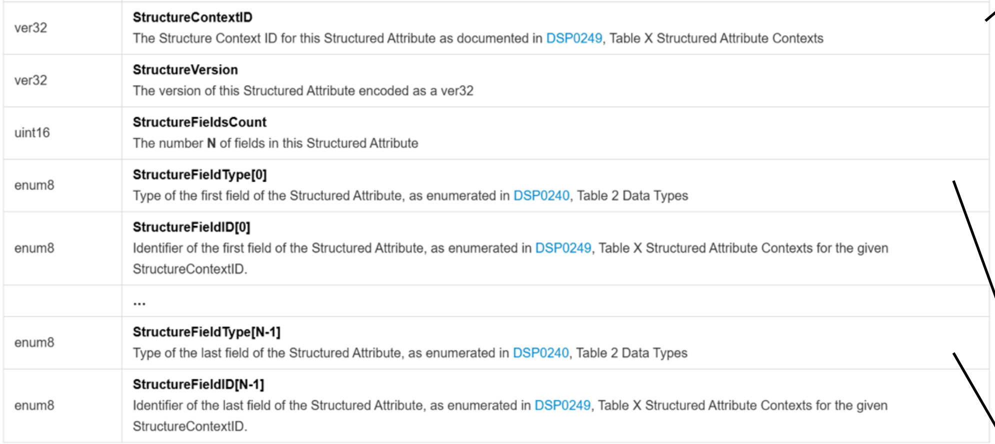
- PLDM lacks a simple way to “atomically” read structured data
  - “Atomic” from perspective of MC.
- Example: UEFI boot progress sensor (for Host SoC model)
  - 8 byte status, 4 byte instance
  - Two PLDM numeric sensors does not give atomicity
  - Extending PLDM numeric sensors to a uint128 does not enable self-description (understanding the structure) via PDRs
- Additional examples:
  - timestamp104 (13 bytes), interval72 (9 bytes), ver32 (4 bytes structured), UUID (16 bytes)
  - 3-axis accelerometer

Proposed Solution – PLDM Structured Attributes

- New type of reading (in addition to numeric and state sensors)
- New PDR type, new read/enable commands
- PDR enumerates a structureType – describes format of data
- For downstream device vendors, this is opt-in.
  - If your current device functions well without Structured Attributes, you do not have to use them.
- Can be cloned, just like sensors, effecters, and Entity Assoc. PDRs

PLDM Structured Attribute PDR

Top Matter: Header, Handle, AttributeID, Entity Type/Instance/Container



ver32	**StructureContextID** The Structure Context ID for this Structured Attribute as documented in DSP0249, Table X Structured Attribute Contexts
ver32	**StructureVersion** The version of this Structured Attribute encoded as a ver32
uint16	**StructureFieldsCount** The number **N** of fields in this Structured Attribute
enum8	**StructureFieldType[0]** Type of the first field of the Structured Attribute, as enumerated in DSP0240, Table 2 Data Types
enum8	**StructureFieldID[0]** Identifier of the first field of the Structured Attribute, as enumerated in DSP0249, Table X Structured Attribute Contexts for the given StructureContextID.
	...
enum8	**StructureFieldType[N-1]** Type of the last field of the Structured Attribute, as enumerated in DSP0240, Table 2 Data Types
enum8	**StructureFieldID[N-1]** Identifier of the last field of the Structured Attribute, as enumerated in DSP0249, Table X Structured Attribute Contexts for the given StructureContextID.

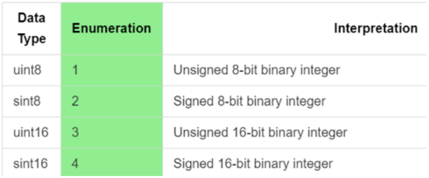
New Structured Attribute Context Table in DSP0249

Structure Context ID	Description		
0x01	Boot Status and Instance as defined in UEFI Platform Initialization Spec Chapter 6 v1.0		
	F_ID	**F_Type**	**Field Name**
	1	uint8	status_code_type
	2	uint8	status_code_severity
	3	uint32	instance
…	…		
0xF000 - 0xFFFF	<Range for OEM Structs>		

```
0x 01
0x f1f0f000 03
0x 0301
0x 0302
0x 0503
```

```
struct uefi_boot_status_v1_0_0
{
     uint8  status_code_type;
     uint8  status_code_severity;
     uint32 instance;
};
```

Enhanced PLDM Types Table in DSP0240



Data Type	Enumeration	Interpretation
uint8	1	Unsigned 8-bit binary integer
sint8	2	Signed 8-bit binary integer
uint16	3	Unsigned 16-bit binary integer
sint16	4	Signed 16-bit binary integer

Putting it all together – the PLDM Host SoC Model

- Overall SoC Sensor Collection
  - Temperature, Power, Health, Boot Status (Structure)
- SoC Processor Cores Collection – Define once, scale
  - Frequency, Utilization
- SoC Memory Collection – Define once, scale
  - Temperature, Power, Utilization, Health
- SoC IO Collections - Define once, scale
  - PCIe, Network Ports
  - Health, Utilization, Speed/Performance

How to get involved

- Contribute to PLDM Host SoC Modeling / Monitoring & Control
  - Host SoC modeling – Common base of function vendors
  - Monitoring & Control v1.4.0 – Scaling for modern devices

- Direct through the PMCI Workgroup Group of DMTF
  - https://www.dmtf.org/standards/pmci

Or

- DMTF Feedback Portal
  - https://www.dmtf.org/standards/feedback

Acknowledgement

Thanks to all the contributors and participants of the PMCI Working Group!

Questions?

DMTF®

**For more information,
visit dmtf.org**

Learn about the PMCI working group at dmtf.org/standards/pmci

Thank you!