BROADCOM®

Application Note

# Emulex® Standard PLDM over MCTP

## Revision 12.8.292.0

## 1 Purpose

This document specifies the Platform Level Data Model (PLDM) commands that are supported by Emulex for PLDM over Management Component Transport Protocol (MCTP). The MCTP runs over a PCIe or SMBus.

This document covers only the following Fibre Channel HBAs:

- LPe31000-series
- LPe32000-series
- LPe35000-series
- LPe36000-series

## 2 Scope

This document defines the commands and data structures that are supported by the Emulex® HBAs PLDM implementation.

The implementation is limited to the following portions of PLDM:

- DSP0240: Base Specification; DSP0241: PLDM over MCTP Binding Specification; and DSP0245: IDs and Codes Specification

  These specifications comprise the core content.
- DSP0248: Platform Monitoring and Control Specification

  This portion enumerates components that comprise the Emulex adapter and can define one or more sensors. The sensors map to elements on the adapter and can later generate asynchronous messages to the system to report a change in the sensor. Consider the sensor very generic in nature — a thing that has values and state transitions between those values. There may be knobs that affect the operation of the thing or that force transitions. A simplistic example is a temperature module that can report when it crosses warning thresholds. A more obtuse example is to report about a firmware download — that it has started, is x% done, or is complete. An asynchronous event must be generated to indicate the newly instantiated firmware revision (if only a firmware reset).
- DSP0257: FRU Data Specification

  This specification provides vendor, model number, serial number, firmware version, and other information for the Emulex device.

The expectation is that there is a single PLDM responder per ASIC (not one per function), functional only in Full-Power mode (as on MCTP over PCIe).

# 3 PLDM Implementation

PLDM provides efficient access to low-level platform monitoring, control, and data transfer functions, such as temperature, fan, voltage, inventory data, event data transfer, and boot control. PLDM over MCTP defines data representations and commands that abstract the platform management hardware. PLDM is designed to be an effective source for mapping under the CIM.

The following sections describe the PLDM commands that are accepted by Emulex products, their semantical behavior, and the data that is returned. The sections also describe the Platform Descriptor Records (PDRs) that describe the topology elements of the Emulex adapter, sensors (that can generate events) bound to elements of the Emulex adapter, and descriptions of nonstandard elements required by the Emulex implementation.

The Emulex PLDM implementation does not contain a Discovery Agent or an Initialization Agent. The Emulex Terminus acts only as a PLDM Responder, except when the Emulex Terminus generates events once they are enabled.

Many fields within PDR records or command responses require unique values. Because the PDLM specification explicitly does not describe how a device interplays with the Discovery Agent to transition its PDR records into the Primary PDR Repository, Broadcom has ensured that the fields are unique within the device that is being reported. Some fields that are not used in the Emulex Small PDR Repository (such as Terminus Handle) are set to 0x0. It is expected that the Discovery Agent will query the PDR records from the device and will overwrite values supplied by Broadcom to values that are unique for the Primary PDR Repository.

Terminus values set by PLDM, such as Terminus ID (TID), do not persist across power cycles or PCI hot/ fundamental resets. The values persist across a firmware restart and any individual function resets.

**NOTE:** This statement is intended for Terminus information such as TID. Relative to the actual data elements set, the persistence model is specific to the item changed.

The Emulex PLDM implementation has a single MCTP endpoint per controller. A controller is assumed to be an adapter based on a single ASIC supporting one or more physical links. This document describes a controller with two physical links. Additional links would be supported by scaling out the existing records or relationships in the manner as illustrated in this document. Currently, there are no per-PCI function topology elements described by PLDM. Other Emulex adapter implementations are possible and may require a variation in the Entity Association PDRs.

# 4 PLDM Command Support

The following commands are supported by the Emulex PLDM.

**Table 1: Emulex PLDM Command Support**

| PLDM Type Code | PLDM Command Code | Command Name | Description |
|---|---|---|---|
| **DSP0240 (Type Code 0x00) - Messaging Control and Discovery Command Set** | | | |
| 0x00 | 0x01 | SetTID | Sets the Terminus TID. |
| 0x00 | 0x02 | GetTID | Gets the Terminus TID. |
| 0x00 | 0x03 | GetPLDMVersion | Gets the version(s) of the supported PLDM specifications. |
| 0x00 | 0x04 | GetPLDMTypes | Gets the supported PLDM type codes. |
| 0x00 | 0x05 | GetPLDMCommands | Gets the supported PLDM command codes per the PLDM type code. |

**Table 1: Emulex PLDM Command Support (Continued)**

| PLDM Type Code | PLDM Command Code | Command Name | Description |
|---|---|---|---|
| **DSP0248 (Type Code 0x02) - Platform Monitoring and Control Command Set** | | | |
| 0x02 | 0x01 | SetTID | Sets the Terminus TID (same as Type 0x00 Cmd 0x01). |
| 0x02 | 0x02 | GetTID | Get the Terminus TID (same as Type 0x00 Cmd 0x02). |
| 0x02 | 0x03 | GetTerminusUID | Obtains a unique user identification (UID) for the Emulex device (ASIC). |
| 0x02 | 0x04 | SetEventReceiver | Specifies the PLDM Terminus to direct event messages. Also enables event posting for particular sensors. |
| 0x02 | 0x05 | GetEventReceiver | Queries the Event Receiver values. |
| 0x02 | 0x10 | SetNumericSensorEnable | Sets the operational state and enables events on a numeric sensor. |
| 0x02 | 0x11 | GetSensorReading | Gets the present reading and state of a numeric sensor. |
| 0x02 | 0x21 | GetStateSensorReadings | Reads state sensor values. |
| 0x02 | 0x50 | GetPDRRepositoryInfo | Gets sizing data about Emulex PDR records. |
| 0x02 | 0x51 | GetPDR | Gets Emulex PDR records. |
| 0x02 | 0x0A | PlatformEventMessage | Posts event messages from the Emulex device. |
| **DSP0257 (Type Code 0x04) - FRU Data Command Set** | | | |
| 0x04 | 0x01 | GetFRURecordTableMetadata | Gets sizing data about the FRU record data. |
| 0x04 | 0x02 | GetFRURecordTable | Gets the FRU record data. |

The data returned and any additional behavior for each command are described in the sections that follow. In cases where the command is structured to do a segmented return of a larger data structure, the section describes only the overall data structure being returned. For those commands, it is expected that the implementation will properly handle the segmented request or response relative to the overall data structure.

# 4.1 PLDM Type Code 0x00 Commands

## 4.1.1 SetTID (Type 0x00, Cmd 0x01)

This command sets the TID value to the supplied value.

## 4.1.2 GetTID (Type 0x00, Cmd 0x02)

This command retrieves the present TID value.

Coming out of a power cycle or PCI hot/fundamental reset, the TID is initialized to 0x00.

## 4.1.3 GetPLDMVersion (Type 0x00, Cmd 0x03)

The Terminus returns the following data:

If the type is 0x00 (Base Specification) or if the type is 0x04 (PLDM for FRU Data):

| | |
|---|---|
| Bytes 0..3 | 0xF1F0F000 in big-endian layout (1.0.0) |
| Bytes 4..7 | The checksum of bytes 0..3 |

If the type is 0x02 (PLDM for Monitoring and Control):

| | |
|---|---|
| Bytes 0..3 | 0xF1F1F000 in big-endian layout (1.1.0) |
| Bytes 4..7 | The checksum of bytes 0..3 |

### 4.1.4 GetPLDMTypes (Type 0x00, Cmd 0x04)

The Terminus returns the following data:

| | | |
|---|---|---|
| Bytes 0 | 0x05 (bits 0, 2 corresponding to types 0x0, 0x2) | bitfield8[0] |
| Bytes 1..7 | 0x00 (no bits set) | bitfield8[1-7] |

### 4.1.5 GetPLDMCommands (Type 0x00, Cmd 0x05)

The Terminus returns the following data:

If the request has PLDM type = 0x0 and version = 0xF1F0F000:

| | | |
|---|---|---|
| Bytes 0 | 0x3E (bits 1, 2, 3, 4, 5 corresponding to Cmd Codes 1..5) | bitfield8[0] |
| Bytes 1..31 | 0x00 (no bits set) | bitfield8[1-31] |

If the request has PLDM type = 0x2 and version = 0xF1F0F000:

| | | |
|---|---|---|
| Bytes 0 | 0x3E (bits 1, 2, 3, 4, 5 corresponding to Cmd Codes 1..5) | bitfield8[0] |
| Bytes 1 | 0x04 (bit 3 corresponding to Cmd Code 10) | bitfield8[1] |
| Bytes 2..3 | 0x00 (no bits set) | bitfield8[2..3] |
| Bytes 4 | 0x02 (bit 2 corresponding to Cmd Code 33) | bitfield8[4] |
| Bytes 5..9 | 0x00 (no bits set) | bitfield8[5..9] |
| Bytes 10 | 0x01 (bit 1 corresponding to Cmd Code 80) | bitfield8[10] |
| Bytes 11..31 | 0x00 (no bits set) | bitfield8[11-31] |

If the request has PLDM type = 0x4 and version = 0xF1F0F000:

| | | |
|---|---|---|
| Bytes 0 | 0x06 (bits 1, 2 corresponding to Cmd Codes 1 and 2) | bitfield8[0] |
| Bytes 1..31 | 0x00 (no bits set) | bitfield8[1-31] |

Other request PLDM type/version combinations fail with the appropriate completion code.

## 4.2 PLDM Type Code 0x02 Commands

### 4.2.1 SetTID (Type 0x02, Cmd 0x01)

See Section SetTID (Type 0x00, Cmd 0x01).

### 4.2.2 GetTID (Type 0x02, Cmd 0x02)

See Section GetTID (Type 0x00, Cmd 0x02).

### 4.2.3 GetTerminusUID (Type 0x02, Cmd 0x03)

This command returns the UID for the Terminus.

Broadcom generates the UID for a controller based on the base MA*C* address for the controller as set by manufacturing data. The method used to create the UID is based on version 1 (based on the timestamp) with a timestamp value of 0. The UID must be unique because the Emulex MAC address is globally unique.

### 4.2.4 SetEventReceiver (Type 0x02, Cmd 0x04)

This command sets the transport-specific address to which the Platform Event Messages are sent. This command is also used to enable or disable event delivery.

The Terminus records the TransportProtocolType and EventReceiverAddressInfo. The EventMessageGlobalEnable is inspected to enable or disable all applicable sensors.

### 4.2.5 GetEventReceiver (Type 0x02, Cmd 0x05)

This command queries the settings on the Terminus for the transport-specific address for Platform Event Messages.

The Terminus replies with its current values for the Event Receiver.

**NOTE:** The initial values for Event Receiver data are as follows:

| | |
|---|---|
| TransportProtocolType = 0x00 | (MCTP) |
| EventReceiverAddressInfo = 0xFF | (Endpoint EID - this is the MCTP Broadcast ID, which is not supported. Therefore it is an invalid value.) |

### 4.2.6 SetNumericSensorEnable (Type 0x02, Cmd 0x10)

This command sets the state of a numeric sensor and enables or disables event delivery from the sensor.

For the indicated sensorID (Temperature, FW Download Percent, Link #*n* Speed), the Terminus sets the state of the sensor and sets whether Event Messages from the sensor are disabled or enabled. By default, the sensor is enabled and events are enabled.

**NOTE:** If the sensor is disabled, no events can be generated by it.

### 4.2.7 GetSensorReading (Type 0x02, Cmd 0x11)

This command queries the value of a particular numeric sensor.

The Terminus replies with its current values for the indicated numeric sensor. The rearmEventState is ignored because the sensor re-arm is automatic.

**NOTE:** If a state set is disabled, presentState, previousState, and eventState must be set to 0x0 (no data).

#### 4.2.7.1 GetSensorReading: Temperature Numeric Sensor

The following fields are returned if the Temperature Numeric Sensor (SensorID # 0x21) is queried:

| | | |
|---|---|---|
| sensorDataSize | 0x01 | enum for “sint8” |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| sensorEventMessageEnable=*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Normal”) or 0x7 (“Upper Warning”) based on the current temperature level. |
| previousState=0x*nn* | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to the same value as presentState. |
| presentReading=0x*nn* | | Set to the current temperature value. |

#### 4.2.7.2 GetSensorReading: Link #1 Numeric Sensor

The following fields are returned if the Link #1 Speed Numeric Sensor (SensorID # 0x31) is queried:

| | | |
|---|---|---|
| sensorDataSize | 0x02 | enum for “uint16”. |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| sensorEventMessageEnable=*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Normal”) or 0x7 (“Upper Warning”) based on the current temperature level. |
| previousState=0x*nn* | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to the same value as presentState. |
| presentReading=0x*nnnn* | | Set to the current Link #1 Speed. |

#### 4.2.7.3 GetSensorReading: Link #2 Speed Numeric Sensor

The following fields are returned if the Link #2 Speed Numeric Sensor (SensorID # 0x32) is queried:

| | | |
|---|---|---|
| sensorDataSize | 0x02 | enum for “uint16”. |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| sensorEventMessageEnable=*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetNumericSensorEnable. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Normal”) or 0x7 (“Upper Warning”) based on the current temperature level. |

| | |
|---|---|
| previousState=0x*nn* | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | Set to the same value as presentState. |
| presentReading=0x*nnnn* | Set to the current Link #2 Speed. |

## 4.2.8 SetStateSensorEnables (Type 0x02, Cmd 0x20)

This command enables/disables a sensor set, or it enables/disables events from a sensor set.

The Terminus adjusts the targeted sensor (based on the SensorID), it changes the state of the indicated state set, and it enables/disables the event posting attribute of the state set.

## 4.2.9 GetStateSensorReadings (Type 0x02, Cmd 0x21)

This command reads one or more of the state sets on a sensor. The Terminus ignores the rearmEventState field. The sensor re-arm is automatic.

The Terminus returns one of the following responses based on the SensorID queried.

**NOTE:** If a state set is “disabled”, presentState, previousState, and eventState must be set to 0x0 (no data).

### 4.2.9.1 GetStateSensorReadings: Controller Device State Sensor

The following fields are returned if the Controller Device State Sensor (SensorID # 0x80) is queried:

| | | |
|---|---|---|
| CompositeSensorCount | 0x08 | The number state sets in the sensor |
| StateFields: | | |
| State Set (#1) | | (Health State) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Normal”) or 0x7 (“Upper Warning”) based on the current temperature level. |
| previousState=0x*nn* | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to the same value as presentState. |
| State Set (#2) | | (Availability) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x01 | | Set to the current state. Value is 0x1 (“Enabled”). |
| previousState=0x01 | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x01 | | Set to the same value as presentState. |
| State Set (#3) | | (Predictive Condition) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x01 | | Set to the current state. Value is 0x1 (“Normal”). |
| previousState=0x01 | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x01 | | Set to the same value as presentState. |

| | |
|---|---|
| State Set (#4) | (Configuration State) |
| sensorOperationalState=0x*nn* | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x01 | Set to the current state. Value is 0x1 (“Valid Config”). |
| previousState=0x01 | Set to the previous state value. Initially equal to presentState. |
| eventState=0x01 | Set to the same value as presentState. |
| State Set (#5) | (Changed Configuration) |
| sensorOperationalState=0x*nn* | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x01 | Set to the current state. Value is 0x1 (“Normal”). |
| previousState=0x01 | Set to the previous state value. Initially equal to presentState. |
| eventState=0x01 | Set to the same value as presentState. |
| State Set (#6) | (Version) |
| sensorOperationalState=0x*nn* | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | Set to the current state. Values are 0x1 (“Normal”) or 0x2 (“Version Change Detected - no conflict”). |
| previousState=0x*nn* | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | Set to the same value as presentState. |
| State Set (#7) | (Device Power State) |
| sensorOperationalState=0x*nn* | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x01 | Set to the current state. Value is 0x1 (“D0”). |
| previousState=0x01 | Set to the previous state value. Initially equal to presentState. |
| eventState=0x01 | Set to the same value as presentState. |
| State Set (#8) | (Emulex FW Download State) |
| sensorOperationalState=0x*nn* | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | Set to the current state. Will initially be 0x1 (“Firmware Update Not Started”). Will change based on the firmware update action. |
| previousState=0x*nn* | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | Set to the same value as presentState. |

#### 4.2.9.2 GetStateSensorReadings: Link #n State Sensor

The following fields are returned if the Link #1 State Sensor (SensorID # 0x81) or the Link #2 State Sensor (SensorID # 0x82) is queried. Values reported are relative to the respective Link #.

| | | |
|---|---|---|
| CompositeSensorCount | 0x03 | The number of state sets in the sensor. |
| StateFields: | | |
| State Set (#1) | | (Link State) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Connected”) or 0x2 (“Disconnected”). |
| previousState=0x*nn* | | Set to the previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to the same value as presentState. |
| State Set (#2) | | (Communication Leash Status) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | | Set to the current state (presence of the SFP). Values are 0x1 (“Leash Connected”) or 0x2 (“Leash Disconnected”). |
| previousState=0x*nn* | | Set to previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to same value as presentState. |
| State Set (#3) | | (Emulex Link Duplex State) |
| sensorOperationalState=0x*nn* | | Set to the current state. Values are 0x0 (enum for “enabled”) or 0x1 (enum for “disabled”). The initial value is “enabled”. This field can be updated by SetStateSensorEnables. |
| presentState=0x*nn* | | Set to the current state. Values are 0x1 (“Full Duplex”) or 0x2 (“Half Duplex”). |
| previousState=0x*nn* | | Set to previous state value. Initially equal to presentState. |
| eventState=0x*nn* | | Set to same value as presentState. |

### 4.2.10 GetPDRRepositoryInfo (Type 0x02, Cmd 0x50)

This command is used to size the PDR data that is to be returned by the Emulex device.

The Terminus returns the following fields:

| | | |
|---|---|---|
| RepositoryState | 0x00 | enum for “available”. |
| UpdateTime | ?TS104? | Set to the creation timestamp. Pick a time in the past. |
| OEMUpdateTime | ?TS104? | Set to same value as UpdateTime. |
| recordCount | 0x*nnnnnnnn* | Set to the count of PDR records. |
| repositorySize | 0x*nnnnnnnn* | Set to the sum of lengths of all PDR records. |
| largestRecordSize | 0x*nnnnnnnn* | Set to the size of the largest PDR record. |
| dataTransferHandleTimeout | 0x*nn* | <TBD> |

## 4.2.11 GetPDR (Type 0x02, Cmd 0x51)

This command retrieves individual PDR records from the Emulex device. The Terminus returns the Entity Association PDRs first, followed by the remainder of the PDR records.

## 4.2.12 PlatformEventMessage (Type 0x02, Cmd 0x0A)

The following event messages can be generated for the applicable sensors.

### 4.2.12.1 PlatformEventMessage: Temperature Event

The Terminus returns the following fields:

| | | |
|---|---|---|
| FormatVersion | 0x01 | |
| TID | 0x*nn* | Set to the current TID value. |
| EventClass | 0x00 | enum for “sensorEvent”. |
| **eventData:** | | |
| sensorID | 0x0021 | Sensor ID for Temperature. |
| sensorEventClass | 0x02 | enum for “numericSensorState”. |
| **When reporting cross over/above the warning threshold:** | | |
| eventState | 0x08 | enum for “UpperWarning”. |
| previousEventState | 0x01 | enum for “Normal” |
| **When reporting cross below the warning threshold:** | | |
| eventState | 0x01 | enum for “Normal”. |
| PreviousEventState | 0x08 | enum for “UpperWarning”. |
| sensorDataSize | 0x01 | enum for “sint8”. |
| presentReading | 0x*nn* | Set to temperature value. |

### 4.2.12.2 PlatformEventMessage: Link Speed Change Event

Link Speed Change events are generated whenever the link speed changes. If the link state transitions to “down”, an event is generated with a “0” speed. When the link state transitions to “up”, an event is generated to indicate the link speed.

The Terminus returns the following fields:

| | | |
|---|---|---|
| FormatVersion | 0x01 | |
| TID | 0x*nn* | Set to current TID value. |
| EventClass | 0x00 | enum for “sensorEvent”. |
| **eventData:** | | |
| **If event on Link #1:** | | |
| sensorID | 0x0031 | Sensor ID for Link #1 Speed. |
| **If event on Link #2:** | | |
| sensorID | 0x0032 | Sensor ID for Link #2 Speed. |
| sensorEventClass | 0x02 | enum for “numericSensorState”. |
| eventState | 0x01 | enum for “Normal”. |

| | | |
|---|---|---|
| PreviousEventState | 0x01 | enum for “Normal”. |
| sensorDataSize | 0x01 | enum for “sint8”. |
| presentReading | 0x*nn* | Set to the link speed (units 1Gb/s) on the appropriate link #. |

### 4.2.12.3 PlatformEventMessage: Controller Device State Event

This event message is used to report changes in one of the state sets contained within the Controller Device State Sensor. The following state sets exist:

- Health State
  An event for this state set is generated under the following conditions:
  – If the temperature becomes equal to or greater than the warning threshold, report 0x7 (“UpperWarning”).
  – If the temperature becomes less than the warning threshold, report 0x1 (“Normal”).

  **NOTE:** Initially set to 0x1 (“Normal”) or 0x7 (“Upper Warning”) based on the existing temperature versus the warning threshold.
- Availability
  No event is generated. The Emulex state is fixed.

  **NOTE:** Initially set to 0x1 (“Enabled”).
- Predictive Condition
  No event is generated. The Emulex state is fixed.

  **NOTE:** Initially set to 0x1 (“Normal”).
- Configuration State
  No event is generated. The Emulex state is fixed.

  **NOTE:** Initially set to 0x1 (“Valid Config”).
- Changed Configuration
  No event is generated. The Emulex state is fixed.

  **NOTE:** Initially set to 0x1 (“Normal”).
- Version
  An event for this state set is generated under the following conditions:
  – If a firmware reset occurs, prior to initiating the reset, it shall report 0x2 (“Version Change Detected - no conflict”).
  **NOTE:** This assumes that host-directed resets are done to instantiate the firmware.

  **NOTE:** Initially set to 0x1 (“Normal”).
- Device Power State
  No event is generated. The Emulex state is fixed. For the HBA, MCTP over PCIe is available only in the D0 state.

  **NOTE:** Initially set to 0x1 (“D0”).
- Emulex FW Download State.
  An event for this state set is generated under the following conditions:
  – Initially when the event is enabled. It shall report 0x1 (“Firmware Update Not Started”).
  – Whenever one of the following FW Update transitions occurs, it reports the new state:
    - 0x1 (not started) -> 0x2 (“Firmware Update Started”)
    - 0x2 (started) -> 0x3 (“Firmware Update Stopped”)
    - 0x2 (started) -> 0x5 (“Firmware Update Failed”)
    - 0x2 (started) -> 0x7 (“Firmware Written Successfully, Awaiting Activation”)
    - 0x3 (update stopped) -> 0x2 (“Firmware Update Started”)
    - 0x5 (update failed) -> 0x2 (“Firmware Update Started”)
    - 0x7 (done, ready to activate) -> 0x2 (“Firmware Update Started”)

  **NOTE:** Initially set to 0x1 (“Firmware Update Not Started”).

**NOTE:** Not all products can accurately detect each and every state. Therefore it is not unusual for the download state to persist for some time. If the download state transitions to an active/in-progress state, the device generates (assuming the event is enabled) Firmware Download Percentage Events.

**NOTE:** If the device is unable to accurately detect the Firmware Download state, the Controller State Sensor does not include the OEM State Set for the Firmware Download state, and the Numerical Sensor for Firmware Download Percentage is also not returned to the system.

The Terminus returns the following fields:

| | | |
|---|---|---|
| FormatVersion | 0x01 | |
| TID | 0x*nn* | Set to current TID value. |
| EventClass | 0x00 | enum for “sensorEvent”. |
| **eventData:** | | |
| sensorID | 0x0080 | The sensor ID for the Controller Device State. |
| sensorEventClass | 0x01 | enum for “StateSensorState”. |
| sensorOffset | 0x0*n* | Set to the offset/index of the state set being reported. |
| eventState | 0x*nn* | Set to the state value being transitioned to. |
| previousEventState | 0x*nn* | Set to the prior event state value. |

### 4.2.12.4 PlatformEventMessage: Link #n State Event

This event message reports changes in one of the state sets contained within the Link #1 State Sensor or the Link #2 State Sensor. The following state sets exist:

- Link State
  An event for this state set is generated under the following conditions:
  – On any Link Up/Down transition. The event reports the ending state.

  **NOTE:** Initially set to 0x2 (“Disconnected”).
- Communication Leash Status
  This state reports the presence of the SFP if supported by the Stock Keeping Unit (SKU).

  **NOTE:** Initially set to 0x1 (“Leash Connected”).
- Emulex Link Duplex
  An event for this state set is generated under the following conditions:
  – Any change between Full Duplex or Half Duplex operation. The event reports the ending state.

  **NOTE:** Initially set to 0x1 (“Full Duplex”).

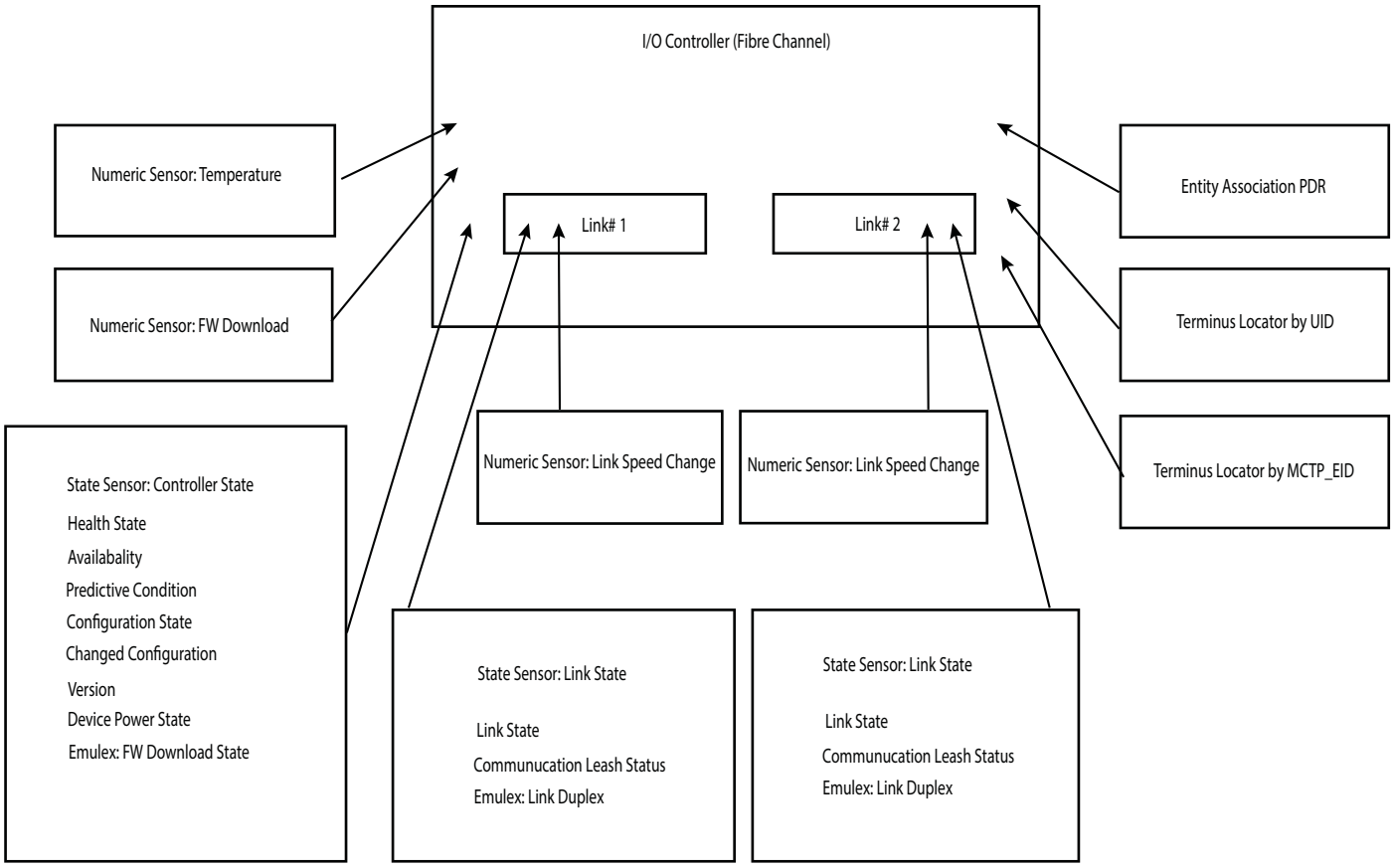
The Terminus returns the following fields:

| | | |
|---|---|---|
| FormatVersion | 0x01 | |
| TID | 0x*nn* | Set to current TID value. |
| EventClass | 0x00 | enum for “sensorEvent”. |
| **eventData:** | | |
| **If for Link #1** | | |
| sensorID | 0x0081 | Sensor ID for Link #1 State. |
| **If for Link #2** | | |
| sensorID | 0x0082 | Sensor ID for Link #2 State. |
| sensorEventClass | 0x01 | enum for “StateSensorState”. |
| sensorOffset | 0x0? | Set to offset/index of the state set being reported. |
| eventState | 0x*nn* | Set to the state value being transitioned to. |
| previousEventState | 0x*nn* | Set to the prior event state value. |

## 4.3 PLDM Type Code 0x02 PDR Records

The PDRs in this section describe what is pictorially represented in Figure 1:

**Figure 1: PDR Records**




**NOTE:** If the product supports additional Links, the topology will reflect additional Link instances.

The following PDRs are generated by the Emulex PLDM. Some PDRs are specific to an SKU type and only the applicable PDRs are returned:

**Table 2: Emulex-Generated PDRs**

| Emulex Record Handle | Emulex Container ID | Type | Instance Number | Contained in Container ID | Description |
|---|---|---|---|---|---|
| 0x00000001.. 0x0000000F | Type 0x0F: Entity Association PDRs | | | | Creates the base “topology tree objects” associated with the device that subsequent PDRs can reference. |
| 0x00000001 | 0x0002 | EntityType: 0x0091: Physical + I/O Controller (145/ 0x91) | 1 | 0x0000 Unknown, so use value 0x0 | This PDR is the Controller (ASIC) association PDR for an Emulex add-in card if it is an FC-based adapter. Thus the selection of I/O controller.<br>Contains two entities that map to physical FC links. |
| 0x00000010.. 0x0000001F | Type 0x01: Terminus Locator PDRs | | | | Reports Connection or Endpoint information to locate the Emulex device. |
| 0x00000010 | N/A | LocatorType 0x00 | N/A | 0x0002 | Locator PDR based on the UID. |
| 0x00000011 | N/A | LocatorType 0x01 | N/A | 0x0002 | Locator PDR based on the MCTP EID. |
| 0x00000020.. 0x0000007F | Type 0x02: Numeric Sensor PDRs | | | | PDR that describes each sensor supported. |
| 0x00000021 | N/A | SensorID 0x21 | 1 | 0x0002 | Temperature Sensor. |
| 0x00000031 | N/A | SensorID 0x31 | 1 | 0x0002 | Link #1 Speed Sensor. |
| 0x00000032 | N/A | SensorID 0x32 | 2 | 0x0002 | Link #2 Speed Sensor. |
| 0x00000080.. 0x000000FF | Type 0x04: State Sensor PDRs | | | | PDR that describes each sensor supported. |
| 0x00000080 | N/A | SensorID 0x80 | | 0x0002 | Controller Device State. |
| 0x00000081 | N/A | SensorID 0x81 | | 0x0002 | Link #1 State. |
| 0x00000082 | N/A | SensorID 0x82 | | 0x0002 | Link #2 State. |
| 0x00000100.. 0x0000011F | Type 0x06: Sensor Auxiliary Names PDRs | | | | Provides “name” strings (per language) to identify the sensors. |
| 0x00000100 | N/A | SensorID 0x21 | N/A | N/A | Temperature Sensor Names. |
| 0x00000110 | N/A | SensorID 0x31 | N/A | N/A | Link #1 Speed. |
| 0x00000111 | N/A | SensorID 0x32 | N/A | N/A | Link #2 Speed. |
| 0x00000120.. 0x0000013F | Type 0x7: OEM Unit PDR | | | | Describes units that are nonstandard. |
| | | | | | (none) |

**Table 2: Emulex-Generated PDRs (Continued)**

| Emulex Record Handle | Emulex Container ID | Type | Instance Number | Contained in Container ID | Description |
|---|---|---|---|---|---|
| 0x00000140.. 0x0000015F | Type 0x08: OEM State Set PDR | | | | Describes OEM-specific state sets. |
| 0x00000140 | N/A | N/A | | 0x0002 | Firmware Download State. |
| 0x00000141 | N/A | N/A | | 0x0002 | Link Duplex State. |
| 0x00000180 | Type 0x14: FRU Record Set PDRs | | | | Describes the FRU Data Record that can be obtained from the Emulex device. |
| 0x00000180 | N/A | FRURecordSet Identifier 0x0001 | | 0x0002 | Specifies the relationship with FRU Record for Controller. |

## 4.3.1 Entity Association PDRs (PDR Type 0x0F)

The Emulex adapter supplies one or more Entity Association PDRs. It is mandatory to define at least one “container” corresponding to the add-in card. Multiple Entity PDRs may be necessary if Broadcom elects to break out components, such as the ASICs, into separate “containers” that reference the main container.

The main “add-in” card Entity Association PDR contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000001 | Emulex-specific PDR ID - for the Entity Association PDR for the network or I/O controller. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x0F | Entity Association PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of PDR data. |
| ContainerID | 0x0002 | Emulex-specific unique container ID. |
| AssociationType | 0x00 | Assumed enum for physical to physical=0; logical=1 based on “P/L bit” references. |
| ContainerEntityType | 0x0091 | Emulex entity type: bit15=P/L, which is 0 for Physical; bits14:0 are the Entity ID, which is 145/0x91 for “I/O Controller”. |
| ContainerEntityInstanceNumber | 0x0001 | Emulex instance: always the first instance. |
| ContainerEntityContainerID | 0x0000 | |
| ContainedEntityCount | 0x0002 | |
| Contained Entity (#1) | (FC link #1) | |
| Type=0x0002 (Physical(0)/Network (2/ 0x02)) | InstanceNumber =0x0001 | ContainerID=0x0002 |
| Contained Entity (#2) | (FC link #2) | |
| Type=0x0002 (Physical(0)/Network (2/ 0x02)) | InstanceNumber =0x0002 | ContainerID=0x0002 |

## 4.3.2 Terminus Locator PDRs (PDR Type 0x01)

The Emulex adapter supplies two Terminus Locator PDRs. One is based on the Emulex UID, and the other is based on the MCTP_EID.

**NOTE:** MCTP may be on top of PCIe or SMBus.

### 4.3.2.1 Terminus Locator PDR (PDR Type 0x01) Based on the UID

The Terminus Locator PDR (UID) for the main add-in card contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000010 | Emulex-specific PDR ID - for the Terminus Locator PDR for the network or<br>I/O controller. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x01 | Terminus Locator PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| Validity | 0x01 | notValid=0, valid=1. |
| TID | 0x*nn* | Set to the current value of the Emulex TID. |
| containerID | 0x0002 | Value from the Controller Entity Association PDR. |
| terminusLocatorType | 0x00 | enum for “UID”. |
| terminusLocatorValueSize | 0x*nn* | Set to the size in bytes of the terminusInstance and deviceUID fields. |
| terminusInstance | 0x01 | Emulex PLDM instance number - always 1. |
| deviceUID | <uid> | Set to UUID for the Emulex device, based on the base MAC address. |

### 4.3.2.2 Terminus Locator PDR (PDR Type 0x01) Based on the MCTP_EID

The Terminus Locator PDR (UID) for the main add-in card contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000011 | Emulex-specific PDR ID - for the Terminus Locator PDR for the network or I/O controller. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x01 | Terminus Locator PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| Validity | 0x01 | notValid=0, valid=1. |
| TID | 0x*nn* | Set to the current value of the Emulex TID. |
| containerID | 0x0002 | Value from the Controller Entity Association PDR. |
| terminusLocatorType | 0x01 | enum for “MCTP_EID”. |
| terminusLocatorValueSize | 0x*nn* | Set to the size in bytes of the EID field. |
| EID | 0x*nn* | Set to the EID for the Emulex device from MCTP. |

### 4.3.3 Numeric Sensor PDRs (PDR Type 0x02)

The Emulex adapter supplies the following Numeric Sensor PDRs as appropriate for the SKU.

Possible sensors are:

- Temperature Level warnings
- Link #1 Speed
- Link #2 Speed

#### 4.3.3.1 Numeric Sensor PDR: Temperature

The Numeric Sensor PDR for Temperature reporting contains the following data:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000021 | Emulex-specific PDR ID - for the Temperature Numeric Sensor PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x02 | Numeric Sensor PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x21 | |
| EntityType | 0x0091 | Set to the value in the Controller Entity Association PDR. |
| EntityInstanceNumber | 0x0001 | Set to the value in the Controller Entity Association PDR. |
| containerID | 0x0002 | Set to the value in the Controller Entity Association PDR. |
| SensorInit | 0x00 | enum for “noInit”. |
| SensorAuxiliaryNamesPDR | 0x01 | enum for “true” - does have a Sensor Auxiliary Names PDR. |
| baseUnit | 0x02 | Unit is “Degree C”. |
| unitModifier | 0x00 | Power of 10 multiplier (for example: =”1”). |
| rateUnit | 0x00 | enum for “none”. |
| baseOEMUnitHandle | 0x00 | Not used. |
| auxUnit | 0x00 | enum for “none” |
| auxUnitModifier | 0x00 | Power of 10 multiplier (for example. = “1”). |
| auxRateUnit | 0x00 | enum for “none” |
| rel | 0x00 | enum for multipledBy - as driving to “*1” scenario. |
| auxOEMUnitHandle | 0x00 | Not used. |
| isLinear | 0x01 | enum for “true”. |
| sensorDataSize | 0x01 | enum for “sint8”. |
| resolution | R:0x00000001 | Resolution of 1 - no change to reading. |
| offset | R:0x00000000 | Constant value always added to the sensor value. |
| accuracy | 0x0000 | +/- accuracy - no deviation. |
| plusTolerance | 0x00 | + variation - no deviation. |
| minusTolerance | 0x00 | - variance - no deviation. |
| hysteresis | 0x00 | 0 = no hysteresis, specified at sint8 (same as SensorDataSize). |
| SupportedThresholds | 0x00 | no bits set - no thresholds supported. |
| thresholdAndHysteresisVolatility | 0x00 | 0 = non-volatile; thresholds set regardless of state. |
| stateTransitionInterval | R:0x00000001 | 1s = length of time for the sensor to enable state change in seconds. |

| | | |
|---|---|---|
| updateInterval | R:0x00000001 | 1s = recommending polling update interval. |
| maxReadable | 0x80 | 128: maximum value that could be returned. Size based on “sint8”. |
| minReadable | 0x00 | Minimum value that could be returned. Size based on “sint8”. |
| rangeFieldFormat | 0x01 | enum for “sint8”. |
| rangeFieldSupport | 0x26 | Bit 1 set = NormalMax and warnHigh supported. |
| nominalValue | 0x00 | Unsupported; size based on “sint8”. |
| normalMax | 0x*nn* | Set to the temperature warning threshold; size based on “sint8” |
| normalMin | 0x00 | Unsupported; size based on “sint8”. |
| warningHigh | 0x*nn* | Set to the temperature warning threshold; size based on “sint8”. |
| warningLow | 0x00 | Unsupported; size based on “sint8”. |
| criticalHigh | 0x*nn* | Unsupported; size based on “sint8”. |
| criticalLow | 0x00 | Unsupported; size based on “sint8”. |
| fatalHigh | 0x*nn* | Unsupported; size based on “sint8”. |
| fatalLow | 0x00 | Unsupported; size based on “sint8”. |

### 4.3.3.2 Numeric Sensor PDR: Link #1 Speed

**NOTE:** Because units are nonstandard, Gb/s/ Mb/s has been chosen as the reporting unit. However, to keep records small (8-bit values not 16), Gb/s is used.

The Numeric Sensor PDR for Link #1 speed reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000031 | Emulex-specific PDR ID - for the Link #1 Speed Numeric Sensor PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x02 | Numeric Sensor PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x31 | |
| EntityType | 0x0002 | Physical (0)/Network (2/0x02): set to the contained link value in Controllers Entity Association PDR. |
| EntityInstanceNumber | 0x0001 | Set to the Link #. |
| containerID | 0x0002 | Set to the value in the Controller Entity Association PDR. |
| SensorInit | 0x00 | enum for “noInit”. |
| SensorAuxiliaryNamesPDR | 0x01 | enum for “true” - does have a Sensor Auxiliary Names PDR. |
| baseUnit | 0x3C | Unit is “Bits”. |
| unitModifier | 0x07 | Power of 10 multiplier (for example 10^7 = 10Mbits). |
| rateUnit | 0x03 | enum for “Per Second”. |
| baseOEMUnitHandle | 0x00 | Unused. |
| auxUnit | 0x00 | enum for “none”. |
| auxUnitModifier | 0x00 | Power of 10 multiplier (for example. = “1”). |
| auxRateUnit | 0x00 | enum for “none”. |
| rel | 0x00 | enum for multipledBy - as driving to “*1” scenario. |
| auxOEMUnitHandle | 0x00 | Not used. |

| | | |
|---|---|---|
| isLinear | 0x01 | enum for “true”. |
| sensorDataSize | 0x02 | enum for “uint16”. |
| resolution | R:0x00000001 | Resolution of 1 - no change to reading. |
| offset | R:0x00000000 | Constant value always added to the sensor value. |
| accuracy | 0x0000 | +/- accuracy - no deviation. |
| plusTolerance | 0x00 | Variation - no deviation. |
| minusTolerance | 0x00 | - variance - no deviation. |
| hysteresis | 0x0000 | 0 = no hysteresis, specified at uint16 (same as SensorDataSize). |
| SupportedThresholds | 0x00 | No bits set - no thresholds supported. |
| thresholdAndHysteresisVolatility | 0x00 | 0 = non-volatile; thresholds set regardless of state. |
| stateTransitionInterval | R:0x00000001 | 1s = length of time for the sensor to enable state change in seconds. |
| updateInterval | R:0x0000000A | 10s = Recommended polling update interval. |
| maxReadable | 0x2710 | 10000: max value (10000 = 100Gb/s) that could be returned; size based on “uint16”. |
| minReadable | 0x0000 | Minimum value that could be returned. Size based on “uint16”. |
| rangeFieldFormat | 0x02 | Enum for “uint16”. |
| rangeFieldSupport | 0x00 | No bits set - no range fields supported. |
| nominalValue | 0x0000 | Unsupported; size based on “uint16”. |
| normalMax | 0x0000 | Unsupported; size based on “uint16”. |
| normalMin | 0x0000 | Unsupported; size based on “uint16”. |
| warningHigh | 0x0000 | Unsupported; size based on “uint16”. |
| warningLow | 0x0000 | Unsupported; size based on “uint16”. |
| criticalHigh | 0x0000 | Unsupported; size based on “uint16”. |
| criticalLow | 0x0000 | Unsupported; size based on “uint16”. |
| fatalHigh | 0x0000 | Unsupported; size based on “uint16”. |
| fatalLow | 0x0000 | Unsupported; size based on “uint16”. |

#### 4.3.3.3 Numeric Sensor PDR: Link #2 Speed

The Numeric Sensor PDR for Link #2 speed reporting contains the same fields as the Link #1 Speed Numeric Sensor PDR, with the following changes:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000032 | Emulex-specific PDR ID - for the Link #2 Speed Numeric Sensor PDR. |
| SensorID | 0x32 | |
| EntityInstanceNumber | 0x0002 | Set to the Link #. |

### 4.3.4 State Sensor PDRs (PDR Type 0x04)

The Emulex adapter supplies the following State Sensor PDRs as appropriate for the SKU.

Possible sensors are:

- Controller Device State
- Link #1 State
- Link #2 State

### 4.3.4.1 State Sensor PDR: Controller Device State

The State Sensor PDR for Controller Device State reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000080 | Emulex-specific PDR ID - for the Controller State Sensor PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x04 | State Sensor PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x80 | |
| EntityType | 0x0091 | Set to the value in the Controller Entity Association PDR. |
| EntityInstanceNumber | 0x0001 | Set to the value in the Controller Entity Association PDR. |
| containerID | 0x0002 | Set to the value in the Controller Entity Association PDR. |
| SensorInit | 0x00 | enum for “noInit”. |
| SensorAuxiliaryNamesPDR | 0x00 | enum for “false” - does not have a Sensor Auxiliary Names PDR. |
| compositeSensorCount | 0x08 | Eight sets of possibleStates. |
| possibleStates (#1): | | |
| stateSetID=0x0001 (Health State) | | possibleStatesSize=2 (2 bytes to cover 10 states) |
| possibleStates (#2): | | |
| stateSetID=0x0002 (Availability) | | possibleStatesSize=3 (3 bytes to cover 16 states) |
| possibleStates (#3): | | |
| stateSetID=0x0003 (Predictive Condition) | | possibleStatesSize=1 (1 byte to cover 2 states) |
| possibleStates (#4): | | |
| stateSetID=0x000F (Configuration State) | | possibleStatesSize=1 (1 byte to cover 4 states) |
| possibleStates (#5): | | |
| stateSetID=0x0010 (Changed Configuration) | | possibleStatesSize=1 (1 byte to cover 2 states) |
| possibleStates (#6): | | |
| stateSetID=0x0012 (Version) | | possibleStatesSize=1 (1 byte to cover 3 states) |
| possibleStates (#7): | | |
| stateSetID=0x0102 (Device Power State) | | possibleStatesSize=1 (1 byte to cover 4 states) |
| possibleStates (#8): | | |
| stateSetID=0x1140 (Emulex FW Download State) | | possibleStatesSize=1 (1 byte to cover 7 states) |

### 4.3.4.2 State Sensor PDR: Link #1 State

The State Sensor PDR for Link #1 Device State reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000081 | Emulex-specific PDR ID - for the Link #1 State Sensor PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x04 | State Sensor PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x81 | |
| EntityType | 0x0002 | Physical (0)/Network (2/0x02): set to the contained link value in the Controller Entity Association PDR. |
| EntityInstanceNumber | 0x0001 | Set to the Link #. |
| containerID | 0x0002 | Set to the value in the Controller Entity Association PDR. |
| SensorInit | 0x00 | enum for “noInit”. |
| SensorAuxiliaryNamesPDR | 0x00 | enum for “false” - does not have a Sensor Auxiliary Names PDR. |
| compositeSensorCount | 0x03 | Three sets of possibleStates. |
| possibleStates (#1): | | |
| stateSetID=0x0021 (Link State) | | possibleStatesSize=1 (1 byte to cover 2 states) |
| possibleStates (#2): | | |
| stateSetID=0x0065 (Communication Leash Status) | | possibleStatesSize=1 (1 byte to cover 2 states)<br>**NOTE:** This reports the presence or not of the SFP. |
| possibleStates (#3): | | |
| stateSetID=0x1141 (Emulex Link Duplex state) | | possibleStatesSize=1 (1 byte to cover 2 states) |

### 4.3.4.3 State Sensor PDR: Link #2 State

The State Sensor PDR for Link #2 Device State reporting contains the same data as the Link #1 State Sensor PDR, with the following changes:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000082 | Emulex-specific PDR ID - for the Link #2 State Sensor PDR. |
| SensorID | 0x82 | |
| EntityInstanceNumber | 0x0002 | Set to the Link #. |

## 4.3.5 Sensor Auxiliary Names PDRs (PDR Type 0x06)

The Emulex adapter provides a Sensor Auxiliary Names PDR for each sensor PDR that is reported.

### 4.3.5.1 Sensor Auxiliary Names PDR: Temperature

The Sensor Auxiliary Names PDR for Temperature reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000100 | Emulex-specific PDR ID - for the Sensor Auxiliary Names PDR for the Temperature. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x06 | Sensor Auxiliary Names PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x0021 | Sensor ID for Temperature. |
| SensorCount | 0x01 | Not a composite sensor. |
| nameString Count | 0x01 | Only one language/name field (for now). |
| Name (#1) | | |
| nameLanguageTag=”en | | “sensorName=”Temperature” |

### 4.3.5.2 Sensor Auxiliary Names PDR: Link #1 Speed

The Sensor Auxiliary Names PDR for Link #1 speed reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000110 | Emulex-specific PDR ID - for the Sensor Auxiliary Names PDR for the Link #1 speed. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x06 | Sensor Auxiliary Names PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| SensorID | 0x0031 | Sensor ID for the Link #1 speed. |
| SensorCount | 0x01 | Not a composite sensor. |
| nameString Count | 0x01 | Only one language/name field (for now). |
| Name (#1) | | |
| nameLanguageTag=”en | | “sensorName=”Link Speed” |

### 4.3.5.3 Sensor Auxiliary Names PDR: Link #2 Speed

The Sensor Auxiliary Names PDR for Link #2 speed reporting contains the same data as the Link #1 Speed Sensor Auxiliary Names PDR, with the following changes:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000111 | Emulex-specific PDR ID - for the Sensor Auxiliary Names PDR for the Link #2 speed. |
| SensorID | 0x0032 | Sensor ID for the Link #2 speed. |

## 4.3.6 OEM Unit PDRs (PDR Type 0x07)

The Emulex adapter generates OEM Unit PDRs for any unit not defined in the standards.

## 4.3.7 OEM State Set PDRs (PDR Type 0x08)

The Emulex adapter supplies the following OEM State Set PDRs as appropriate for the SKU.

Possible sensors are:

- Emulex Firmware Download State
- Emulex Link Duplex State

### 4.3.7.1 OEM State Set PDR: Firmware Download State

The OEM State Set PDR for Firmware Download State reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000140 | Emulex-specific PDR ID - for the Firmware Download State Set PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x08 | OEM State Set PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| OEMStateSetIDHandle | 0x1140 | Set to the Emulex OEM State Set identifier OR'd with 0x1000 to make OEM-specific. |
| vendorIANA | 0x0000006C | Emulex IANA number. |
| OEMStateSetID | 0x0140 | Emulex State Set ID for the FW Download State. |
| unspecifiedValueHint | 0x00 | enum for “treatAsUnspecified”. |
| stateCount | 0x07 | For the seven states below. |
| StateValue Record (#1) | | |
| minStateValue=0x01 | | Lowest state value. |
| maxStateValue=0x01 | | Largest state value; only one string for a state. |
| stringCount=0x01 | | Only one language string (for now). |
| String (#1) | | |
| stateLanguageTag=”en | | “stateName=”Firmware Update Not Started” |
| StateValue Record (#2) | | |
| minStateValue=0x02 | | Lowest state value. |
| maxStateValue=0x02 | | Largest state value; Only one string for a state. |
| stringCount=0x01 | | Only one language string (for now). |
| String (#1) | | |
| stateLanguageTag=”en | | “stateName=”Firmware Update Started” |

| | |
|---|---|
| StateValue Record (#3) | |
| minStateValue=0x03 | Lowest state value. |
| maxStateValue=0x03 | Largest state value; only one string for a state. |
| stringCount=0x01 | Only one language string (for now). |
| String (#1) | |
| stateLanguageTag=”en | “stateName=”Firmware Update Stopped” |
| StateValue Record (#4) | |
| minStateValue=0x04 | Lowest state value. |
| maxStateValue=0x04 | Largest state value; only one string for a state. |
| stringCount=0x01 | Only one language string (for now). |
| String (#1) | |
| stateLanguageTag=”en | “stateName=”Firmware Updated Successfully” |
| StateValue Record (#5) | |
| minStateValue=0x05 | Lowest state value. |
| maxStateValue=0x05 | Largest state value; only one string for a state. |
| stringCount=0x01 | Only one language string (for now). |
| String (#1) | |
| stateLanguageTag=”en | “stateName=”Firmware Update Failed” |
| StateValue Record (#6) | |
| minStateValue=0x06 | Lowest state value. |
| maxStateValue=0x06 | Largest state value; only one string for a state. |
| stringCount=0x01 | Only one language string (for now). |
| String (#1) | |
| stateLanguageTag=”en | “stateName=”Firmware Update In Progress” |
| StateValue Record (#7) | |
| minStateValue=0x07 | Lowest state value. |
| maxStateValue=0x07 | Largest state value; only one string for a state. |
| stringCount=0x01 | Only one language string (for now). |
| String (#1) | |
| stateLanguageTag=”en | “stateName=”Firmware Written Successfully, Awaiting Activation” |

### 4.3.7.2 OEM State Set PDR: Link Duplex State

The OEM State Set PDR for Link Duplex State reporting contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000141 | Emulex-specific PDR ID - for the Link Duplex State Set PDR. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x08 | OEM State Set PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0xnnnn | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| OEMStateSetIDHandle | 0x1141 | Set to the Emulex OEM State Set identifier OR'd with 0x1000 to make OEM-specific. |

| | | |
|---|---|---|
| vendorIANA | 0x0000006C | Emulex IANA number. |
| OEMStateSetID | 0x0141 | Emulex State Set ID for the FW Download State. |
| unspecifiedValueHint | 0x00 | enum for “treatAsUnspecified”. |
| stateCount | 0x02 | For the two states below. |
| StateValue Record (#1) | | |
| minStateValue=0x01 | | Lowest state value. |
| maxStateValue=0x01 | | Largest state value; only one string for a state. |
| stringCount=0x01 | | Only one language string (for now). |
| String (#1) | | |
| stateLanguageTag=”en | | “stateName=”Full Duplex” |
| StateValue Record (#2) | | |
| minStateValue=0x02 | | Lowest state value. |
| maxStateValue=0x02 | | Largest state value; only one string for a state. |
| stringCount=0x01 | | Only one language string (for now). |
| String (#1) | | |
| stateLanguageTag=”en | | “stateName=”Half Duplex” |

### 4.3.8 FRU Record Set PDRs (PDR Type 0x14)

This PDR specifies the binding of the FRU Record Data set with the PLDM object to which it corresponds.

The FRU Record Set PDR contains the following fields:

| | | |
|---|---|---|
| Hdr: RecordHandle | 0x00000180 | Emulex-specific PDR ID - for the FRU Record Set PDR for the network or I/O controller. |
| Hdr: PDRHeaderVersion | 0x01 | |
| Hdr PDRType | 0x14 | FRU Record Set PDR. |
| Hdr RecordChangeNumber | 0x0000 | |
| Hdr DataLength | 0x*nnnn* | Size of the PDR data. |
| PLDMTerminusHandle | 0x0000 | Unique handle - all set to 0x0000. |
| FRURecordSetIdentifier | 0x0001 | Set to an Emulex/Terminus-specific value for the FRU Record. |
| EntityType | 0x0091 | Emulex entity type from the Controller Entity Association PDR. |
| EntityInstanceNumber | 0x0001 | Emulex entity instance # from the Controller Entity Association PDR. |
| ContainerID | 0x0002 | Emulex container ID from the Controller Entity Association PDR. |

# 4.4 PLDM Type Code 0x04 Commands

## 4.4.1 GetFRURecordTableMetadata (Type 0x04, Cmd 0x01)

The Terminus returns the following fields:

| | |
|---|---|
| FRUDataMajorVersion | 0x01 |
| FRUDataMinorVersion | 0x00 |
| FRUTableMaximumSize | 0x00000000 (SetFRURecordTable not supported) |
| FRUTableLength | Length of the FRU Record Table described in GetFRURecordTable. |
| Total Number of Record Set IDs | Based on the FRU Record Table data described in GetFRURecordTable. |
| Total number of records in table | 1 |
| Integrity Checksum | CRC |

## 4.4.2 GetFRURecordTable (Type 0x04, Cmd 0x02)

This section provides an indication of how revision 12.8 Emulex branded Fibre Channel HBA firmware responds to inquiries described in the DMTF standard DSP0257 rev 1.0.0: Platform Level Data Model (PLDM) for FRU Data Specification. This information can be used to properly interpret the responses of HBAs equipped with this firmware.

**NOTE:** Broadcom retains the right, over time, to modify this table and/or to add additional FRU functionality. Any such changes must comply with DSP0257. For example, an additional field might be added to one of the FRU record sets, causing the number of FRU fields to be incremented and the additional field Type/Length/Value to be returned. To avoid misinterpretation of the data, follow the DSP0257 guidelines on field counts and record lengths.

The Terminus returns the FRU Record Data Table as shown in Table 3.

**Table 3: MCTP PLDM FRU Record Table Definitions**

| **Field** | | | **Value** |
|---|---|---|---|
| FRU Record Set ID | | | 0x0001 (Emulex-Specific Unique Number - for Generic PLDM: General FRU Record) |
| FRU Record Type | | | 0x1 (General FRU Record) |
| Number of FRU Fields | | | 14 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 0x01 (Chassis Type) | 0x0 | “” |
| 2 | 0x02 (Model) | 0x0 | “” |
| 3 | 0x03 (Part Number) | Length of Value string | “LPe32002” (example) |
| 4 | 0x04 (Serial Number) | Length of Value string | “<serial number>” |
| 5 | 0x05 (Manufacturer) | 8 | “BROADCOM” |
| 6 | 0x06 (Manufacture Date) | 13 | “2019-05-04-18” (example) |
| 7 | 0x07 (Vendor) | 8 | “BROADCOM” |
| 8 | 0x08 (Name) | Length of Value string | “LPe32002” (example) |
| 9 | 0x09 (SKU) | 0x0 | “” |
| 10 | 0x0A (Version) | 2 | “<version>” |
| 11 | 0x0B (Asset Tag) | 0x0 | “” |

**Table 3: MCTP PLDM FRU Record Table Definitions (Continued)**

| **Field** | | | **Value** |
|---|---|---|---|
| 12 | 0x0C (Description) | Length of Value string | “Emulex LPe32002 2-Port 32Gb Fibre Channel Adapter” (example) |
| 13 | 0x0D (Eng Change Lvl) | 0x0 | “” |
| **#** | **Type** | **Length** | **Value** |
| 14 | 0x0E (Other_Info) | Length of Value string | “<string>” (for Broadcom use) |
| FRU Record Set ID | | | 0x0003 (Emulex-Specific Unique Number - for Emulex PLDM FRU Table - CHIP) |
| FRU Record Type | | | 0xFE (OEM FRU Record) |
| Number of FRU Fields | | | 3 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 0x01 (Vendor IANA) | 4 | 0x0000006C |
| 2 | 0x02 (OEM) | Length of Value string | “FW Version: 12.6.192.11” (example) |
| 3 | 0x03 (OEM) | Length of Value string | “PCIe Link Speed: XXX” (example) |
| FRU Record Set ID | | | 0x0004 (Emulex-Specific Unique Number - for Emulex PLDM FRU Table - Port 0) |
| FRU Record Type | | | 0xFE (OEM FRU Record) |
| Number of FRU Fields | | | 8 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 0x01 (Vendor IANA) | 4 | 0x0000006C |
| 2 | 0x81 (OEM) | 12 | “Port Name: 0” (name dependent on SKU) |
| 3 | 0x82 (OEM) | 13 | “Link Type: FC” |
| 4 | 0x83 (OEM) | 22 | “WWNN: 20000000C9142356” (example) |
| 5 | 0x84 (OEM) | 22 | “WWPN: 10000000C9142356” (example) |
| 6 | 0x85 (OEM) | 30 | “Factory WWNN: 20000000C9142356” (example) |
| 7 | 0x86 (OEM) | 30 | “Factory WWPN: 10000000C9142356” (example) |
| 8 | 0x87 (OEM) | Length of Value string | “FC Link Speed Capabilities: 32/16/8” (example) |
| FRU Record Set ID | | | 0x0005 (Emulex-Specific Unique Number - for Emulex PLDM FRU Table - Port 1) (if present) |
| FRU Record Type | | | 0xFE (OEM FRU Record) |
| Number of FRU Fields | | | 8 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 0x01 (Vendor IANA) | 4 | 0x0000006C |
| 2 | 0x81 (OEM) | 12 | “Port Name: 1” (name dependent on SKU) |
| 3 | 0x82 (OEM) | 13 | “Link Type: FC” |
| 4 | 0x83 (OEM) | 22 | “WWNN: 20000000C9142357” (example) |
| 5 | 0x84 (OEM) | 22 | “WWPN: 10000000C9142357” (example) |
| 6 | 0x85 (OEM) | 30 | “Factory WWNN: 20000000C9142357” (example) |
| 7 | 0x86 (OEM) | 30 | “Factory WWPN: 10000000C9142357” (example) |
| 8 | 0x87 (OEM) | Length of Value string | “FC Link Speed Capabilities: 32/16/8” (example) |

**Table 3: MCTP PLDM FRU Record Table Definitions (Continued)**

| **Field** | | | **Value** |
|---|---|---|---|
| FRU Record Set ID | | | 0x0006 (Emulex-Specific Unique Number - for Emulex PLDM FRU Table - Port 2) (if present) |
| FRU Record Type | | | 0xFE (OEM FRU Record) |
| Number of FRU Fields | | | 8 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 0x01 (Vendor IANA) | 4 | 0x0000006C |
| 2 | 0x81 (OEM) | 12 | “Port Name: 2” (name dependent on SKU) |
| 3 | 0x82 (OEM) | 13 | “Link Type: FC” |
| 4 | 0x83 (OEM) | 22 | “WWNN: 20000000C9142358” (example) |
| 5 | 0x84 (OEM) | 22 | “WWPN: 10000000C9142358” (example) |
| 6 | 0x85 (OEM) | 30 | “Factory WWNN: 20000000C9142358” (example) |
| 7 | 0x86 (OEM) | 30 | “Factory WWPN: 10000000C9142358” (example) |
| 8 | 0x87 (OEM) | Length of Value string | “FC Link Speed Capabilities: 32/16/8” (example) |
| FRU Record Set ID | | | 0x0007 (Emulex-Specific Unique Number - for Emulex PLDM FRU Table - Port 3) (if present) |
| FRU Record Type | | | 0xFE (OEM FRU Record) |
| Number of FRU Fields | | | 8 |
| Encoding Type for FRU Fields | | | 0x01 (ASCII) |
| **#** | **Type** | **Length** | **Value** |
| 1 | 1 (Vendor IANA) | 4 | 0x0000006C |
| 2 | 0x01 (Vendor IANA) | 12 | “Port Name: 3” (name dependent on SKU) |
| 3 | 0x81 (OEM) | 13 | “Link Type: FC” |
| 4 | 0x82 (OEM) | 22 | “WWNN: 20000000C9142359” (example) |
| 5 | 0x83 (OEM) | 22 | “WWPN: 10000000C9142359” (example) |
| 6 | 0x84 (OEM) | 30 | “Factory WWNN: 20000000C9142359” (example) |
| 7 | 0x85 (OEM) | 30 | “Factory WWPN: 10000000C9142359” (example) |
| 8 | 0x86 (OEM) | Length of Value string | “FC Link Speed Capabilities: 32/16/8” (example) |

# 5 Packet Examples for PDLM Messages Using MCTP/PCIe

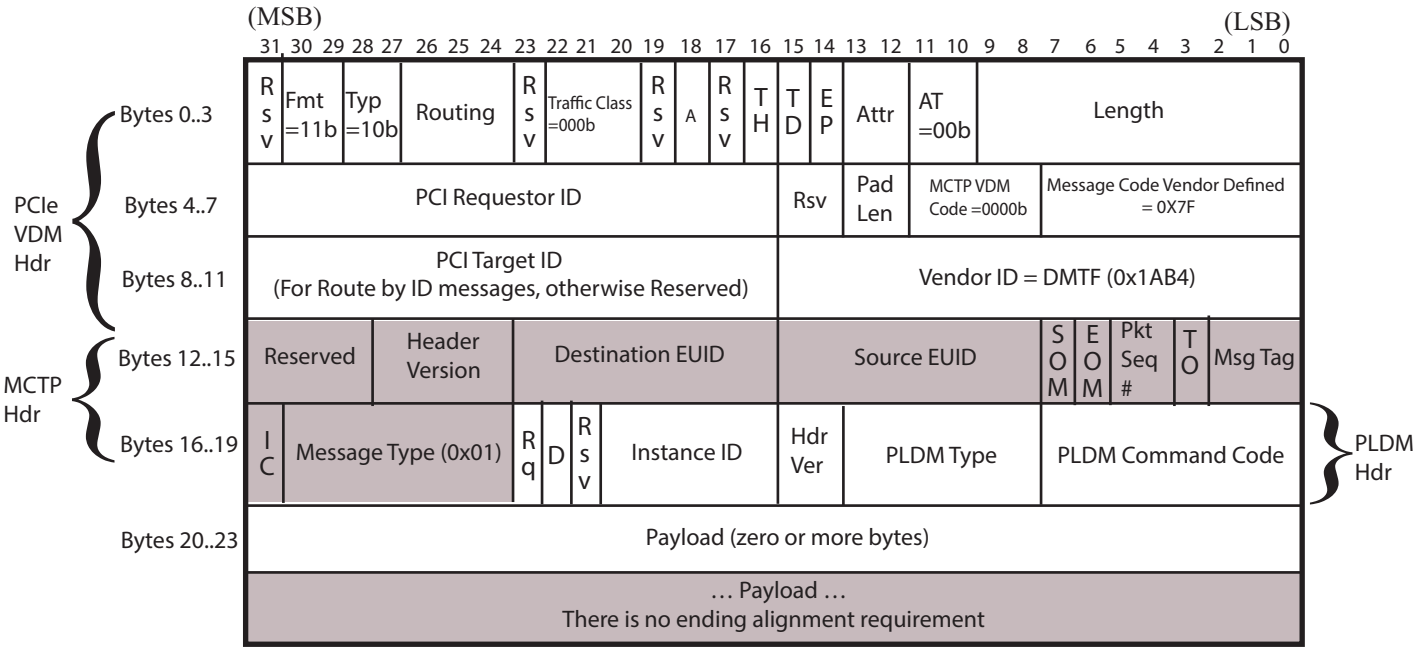
This section provides illustrations of the total PCIe packet format, which contains the PCIe VDM header for MCTP, the MCTP header, and the PLDM request or PLDM response.

**NOTE:** PLDM messages might be fragmented into multiple MCTP messages.

The PLDM request on MCTP on PCIe packet is illustrated in Figure 2.

**NOTE:** The structure is a big-endian byte stream.

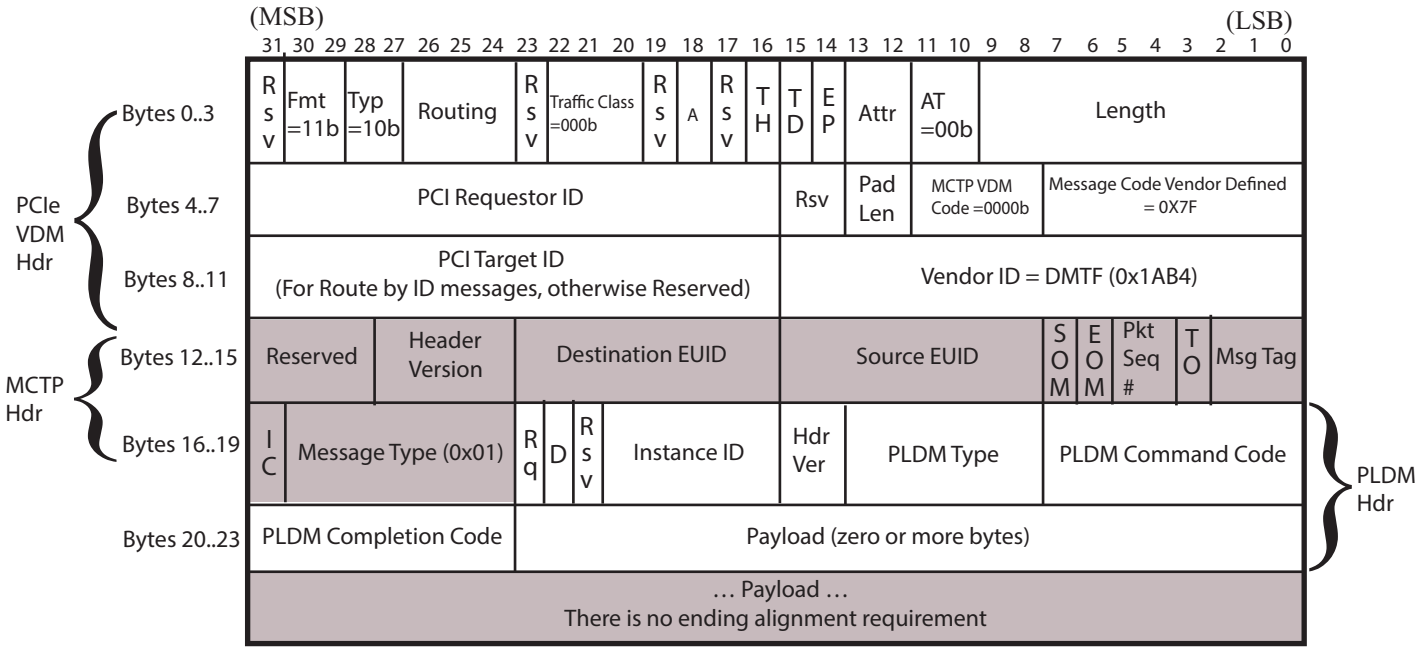
**Figure 2: PCIe Packet Using MCTP - PLDM Request Format**



The PLDM response (which has one more byte in the PDLM header) on MCTP on PCIe packet is illustrated in Figure 3.

**NOTE:** The structure is a big-endian byte stream.

**Figure 3: PCIe Packet Using MCTP - PLDM Response Format**



# 6 References

| Publication | Description |
|---|---|
| DMTF DSP0236<br>Management Component Transport Protocol (MCTP) Base Specification<br>Version 1.2.0<br>January 24, 2013 | www.dmtf.org/sites/default/files/standards/documents/DSP0236_1.2.0.pdf |
| DMTF DSP0256<br>Management Component Transport Protocol (MCTP) Host Interface Specification<br>Version 1.0.0<br>July 21, 2010 | www.dmtf.org/sites/default/files/standards/documents/DSP0256_1.0.0.pdf |
| DMTF DSP0239<br>Management Component Transport Protocol (MCTP) IDs and Codes<br>Version 1.2.0<br>August 28, 2012 | www.dmtf.org/sites/default/files/standards/documents/DSP0239_1.2.0.pdf |
| DMTF DSP0238<br>Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification<br>Version 1.0.1<br>December 11, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0238_1.0.1.pdf |
| DMTF DSP0240<br>Platform Level Data Model (PLDM) Base Specification<br>Version 1.0.0<br>April 23, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0240_1.0.0.pdf |
| DMTF DSP0245<br>PLDM IDs and Codes Specification<br>Version 1.1.0<br>January 26, 2011 | www.dmtf.org/sites/default/files/standards/documents/DSP0245_1.1.0.pdf |
| DMTF DSP0241<br>PLDM Over MCTP Binding Specification<br>Version 1.0.0<br>April 23, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0241_1.0.0.pdf |
| DMTF DSP0248<br>PLDM for Platform Monitoring and Control Specification<br>Version 1.1.0<br>November 8, 2011 | www.dmtf.org/sites/default/files/standards/documents/DSP0248_1.1.0.pdf |
| DMTF DSP0249<br>PLDM State Set Specification<br>Version 1.0.0<br>March 16, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0249_1.0.0.pdf |
| DMTF DSP0257<br>PLDM for FRU Data Specification<br>Version 1.0.0<br>October 26, 2011 | www.dmtf.org/sites/default/files/standards/documents/DSP0257_1.0.0.pdf |

| Publication | Description |
|---|---|
| DMTF DSP0267<br>PLDM for Firmware Update Specification<br>Version 1.0.0<br>August 23, 2016 | www.dmtf.org/sites/default/files/standards/documents/DSP0267_1.0.0a.pdf |
| DMTF DSP0247<br>PLDM for BIOS Control and Configuration Specification<br>Version 1.0.0<br>April 23, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0247_1.0.0.pdf |
| DMTF DSP0246<br>PLDM for SMBIOS Data Transfer Specification<br>Version 1.0.1<br>December 11, 2009 | www.dmtf.org/sites/default/files/standards/documents/DSP0246_1.0.1.pdf |
| Official IANA Enterprise Numbers | www.iana.org/assignments/enterprise-numbers |