Adding Event Class - CPEREvent

Version 0.3

Work In Progress by PMCI

Last Updated 12/7/2021

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

Acknowledgement

- Some of the content in this presentation references and relies on the UEFI 2.9 specification at: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_9_2021_03_18.pdf

Feedback

- Industry feedback on this proposal is encouraged
  - https://www.dmtf.org/standards/feedback

Motivation and Goals

- Platform error data is described using standard format
  - UEFI Spec defines Common Platform Error Record (CPER)
- There are standard methods for reporting the CPER records to the OS, as well as to the BMC
  - See Redfish Schema 2021.3WIP: https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2021.3WIP.99.zip
- However, there is no standard method for transporting such error events from the host to the BMC over MCTP/PLDM
- Goal:
  - Define an event class to transfer platform error events over MCTP with PLDM Type 2
  - The event data should be able to carry CPER industry standard format.

Documents that need to be updated

- DSP0248 – Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification

DSP0248 - PLDM for Platform monitoring and Control specification

- Proposal: Add PLDM Event type
  - 07h : CPEREvent

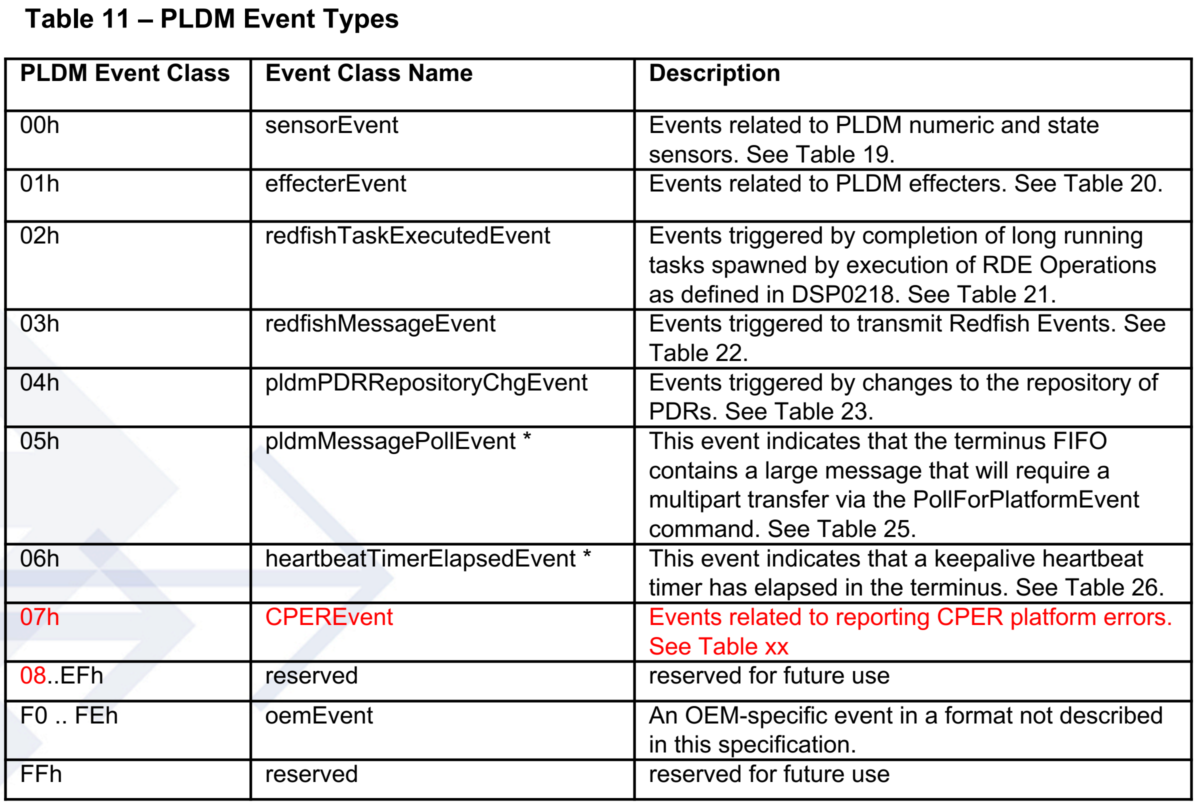


Table 11 – PLDM Event Types

PLDM Event Class	Event Class Name	Description
00h	sensorEvent	Events related to PLDM numeric and state sensors. See Table 19.
01h	effecterEvent	Events related to PLDM effecters. See Table 20.
02h	redfishTaskExecutedEvent	Events triggered by completion of long running tasks spawned by execution of RDE Operations as defined in DSP0218. See Table 21.
03h	redfishMessageEvent	Events triggered to transmit Redfish Events. See Table 22.
04h	pldmPDRRepositoryChgEvent	Events triggered by changes to the repository of PDRs. See Table 23.
05h	pldmMessagePollEvent *	This event indicates that the terminus FIFO contains a large message that will require a multipart transfer via the PollForPlatformEvent command. See Table 25.
06h	heartbeatTimerElapsedEvent *	This event indicates that a keepalive heartbeat timer has elapsed in the terminus. See Table 26.
07h	CPEREvent	Events related to reporting CPER platform errors. See Table xx
08..EFh	reserved	reserved for future use
F0 .. FEh	oemEvent	An OEM-specific event in a format not described in this specification.
FFh	reserved	reserved for future use

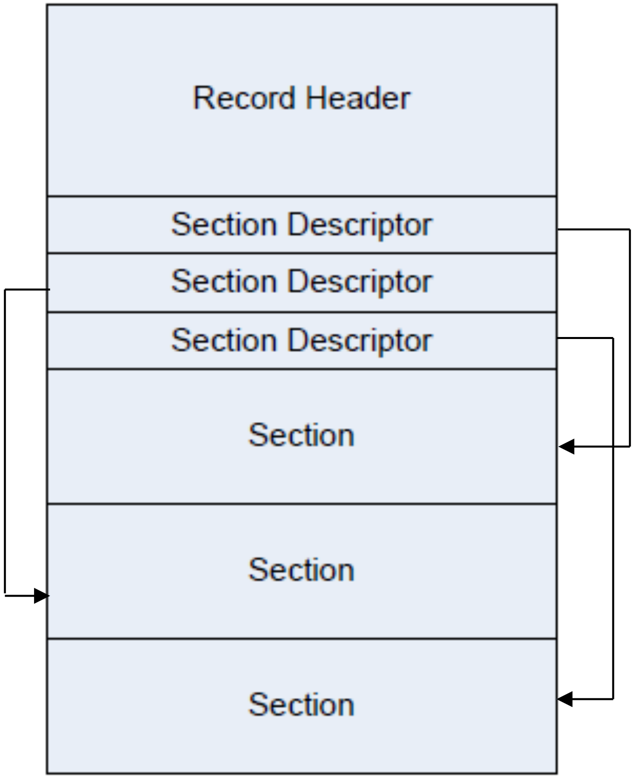
DSP0248 - PLDM for Platform monitoring and Control specification

- Proposal: Add event Data format for CPEREvent

Type	Request data
uint8	**formatVersion** Version of the event format(the format and definition of the following bytes): 0x01 for this specification
uint8	**formatType** Type of Error event in EventData 0x00 = Common Platform Error Record (CPER) – Full Record with Header and one or more Sections 0x01= Single CPER Section 0x02…0xFF = Reserved
uint16	**eventDataLength** Length in bytes of the eventData field below
var	**eventData** formatType = 0x00 A chunk of CPER formatted data including record header, section descriptions and one or more sections, as described in UEFI specification [xx] appendix N – Common Platform Error Record formatType = 0x01 A chunk of CPER formatted data that contains a single section without the header, as described in UEFI specification [xx] appendix N – Common Platform Error Record

CPER background

- UEFI defined standard “Common Platform Error Record” – CPER
  - Format defined in the UEFI 2.9 Specification, Appendix N
  - Implemented by UEFI Linux and Windows (WHEA) for hardware error reporting (by UEFI and the OS)
  - Standard OS code and tools to parse the well-defined error record format
    - Examples: Linux kernel, Windows, WinDbg, DumpRec
  - Reported to the OS via
    - UEFI Variables (HwErrRec####), one for each error record (UEFI 2.9 Specification, section 8.2.4.2 , and Appendix P)
    - ACPI ERST table, BERT table



Record Header

Section Descriptor

Section Descriptor

Section Descriptor

Section

Section

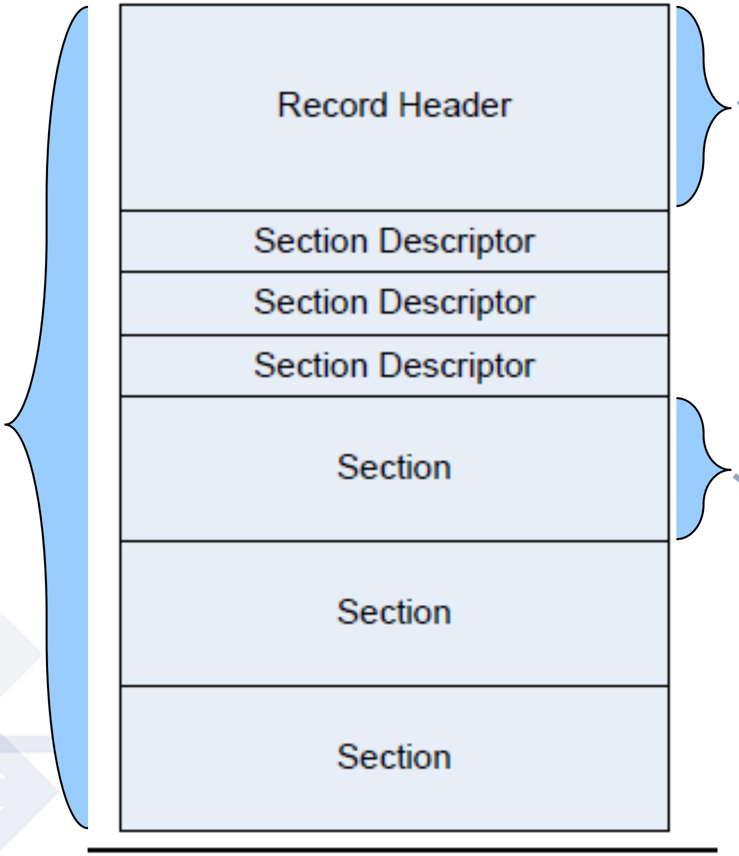
Section

References

- UEFI 2.9 Specification Appendix N - https://uefi.org/sites/default/files/resources/UEFI_Spec_2_9_2021_03_18.pdf
- Server RAS and UEFI CPER, UEFI Plugfest Presentation, March 2017: https://uefi.org/sites/default/files/resources/Spike%20Yuan-%20Server%20RAS%20and%20UEFI%20CPER_final.pdf
- Linux kernel CPER implementation: https://github.com/torvalds/linux/blob/master/drivers/firmware/efi/cper.c
- Windows WHEA error record implementation (based on UEFI CPER): https://docs.microsoft.com/en-us/windows-hardware/drivers/whea/error-records
- Redfish Release 2021.3 Overview: https://www.dmtf.org/sites/default/files/Redfish_Release_2021.3_Overview.pdf

Backup

CPER format in a glance



Each CPER Record has a header and multiple sections

A system may contain multiple CPER Records

Each Section describes a specific error from a specific source/device

Figure 78. Error Record Format

UEFI CPER Format Definition - Header

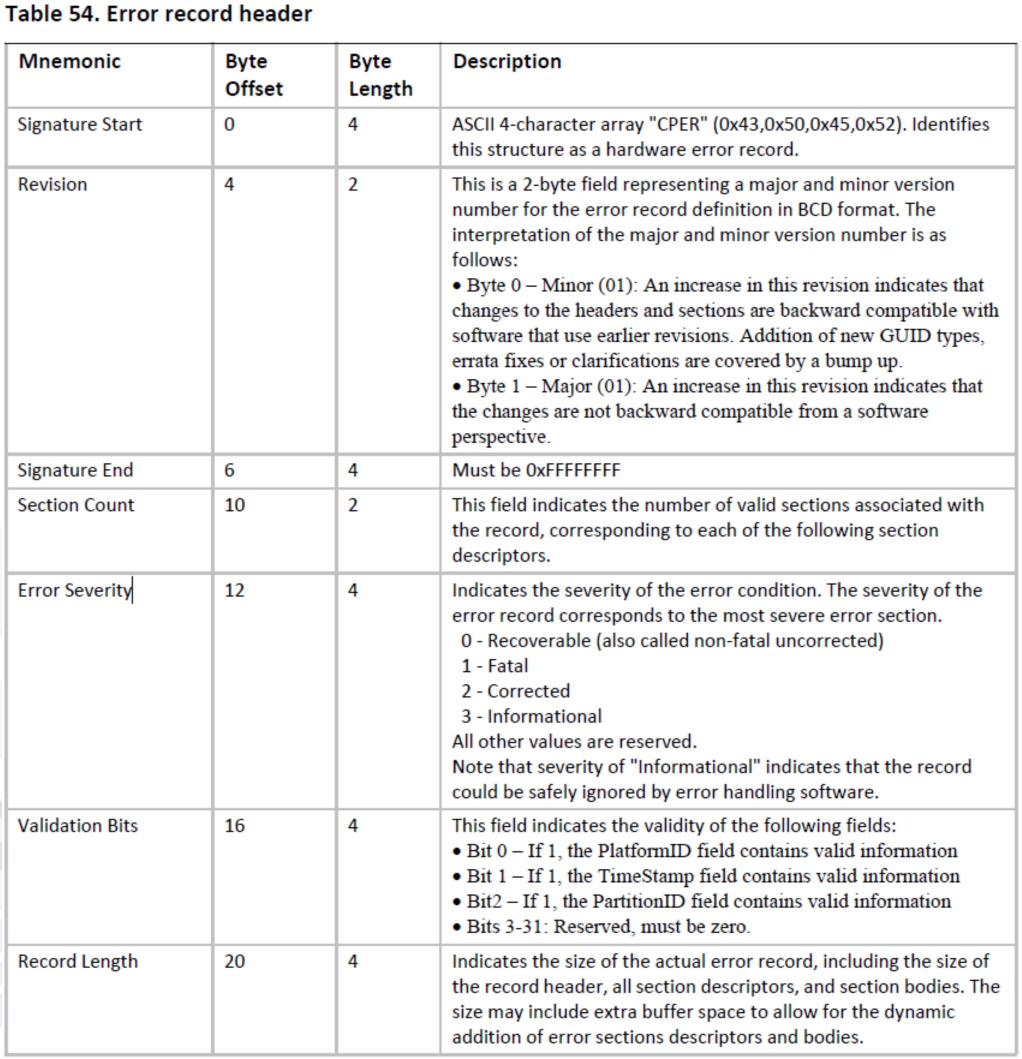
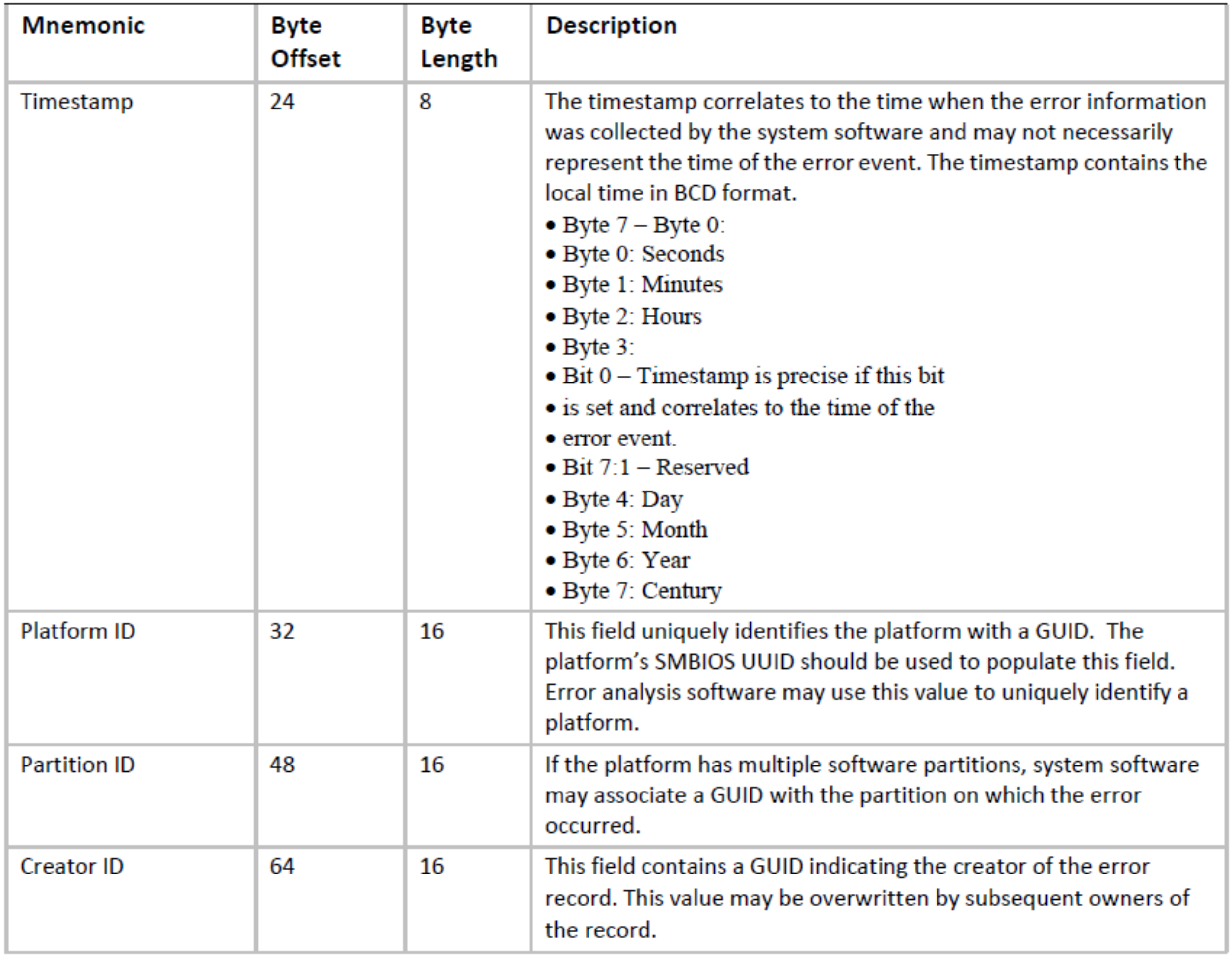


Table 54. Error record header

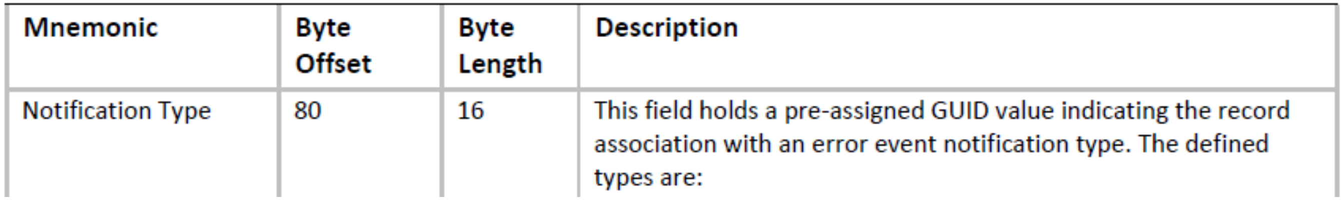
Mnemonic	Byte Offset	Byte Length	Description
Signature Start	0	4	ASCII 4-character array "CPER" (0x43,0x50,0x45,0x52). Identifies this structure as a hardware error record.
Revision	4	2	This is a 2-byte field representing a major and minor version number for the error record definition in BCD format. The interpretation of the major and minor version number is as follows: • Byte 0 – Minor (01): An increase in this revision indicates that changes to the headers and sections are backward compatible with software that use earlier revisions. Addition of new GUID types, errata fixes or clarifications are covered by a bump up. • Byte 1 – Major (01): An increase in this revision indicates that the changes are not backward compatible from a software perspective.
Signature End	6	4	Must be 0xFFFFFFFF
Section Count	10	2	This field indicates the number of valid sections associated with the record, corresponding to each of the following section descriptors.
Error Severity	12	4	Indicates the severity of the error condition. The severity of the error record corresponds to the most severe error section. 0 - Recoverable (also called non-fatal uncorrected) 1 - Fatal 2 - Corrected 3 - Informational All other values are reserved. Note that severity of "Informational" indicates that the record could be safely ignored by error handling software.
Validation Bits	16	4	This field indicates the validity of the following fields: • Bit 0 – If 1, the PlatformID field contains valid information • Bit 1 – If 1, the TimeStamp field contains valid information • Bit2 – If 1, the PartitionID field contains valid information • Bits 3-31: Reserved, must be zero.
Record Length	20	4	Indicates the size of the actual error record, including the size of the record header, all section descriptors, and section bodies. The size may include extra buffer space to allow for the dynamic addition of error sections descriptors and bodies.

UEFI CPER Format Definition - Header



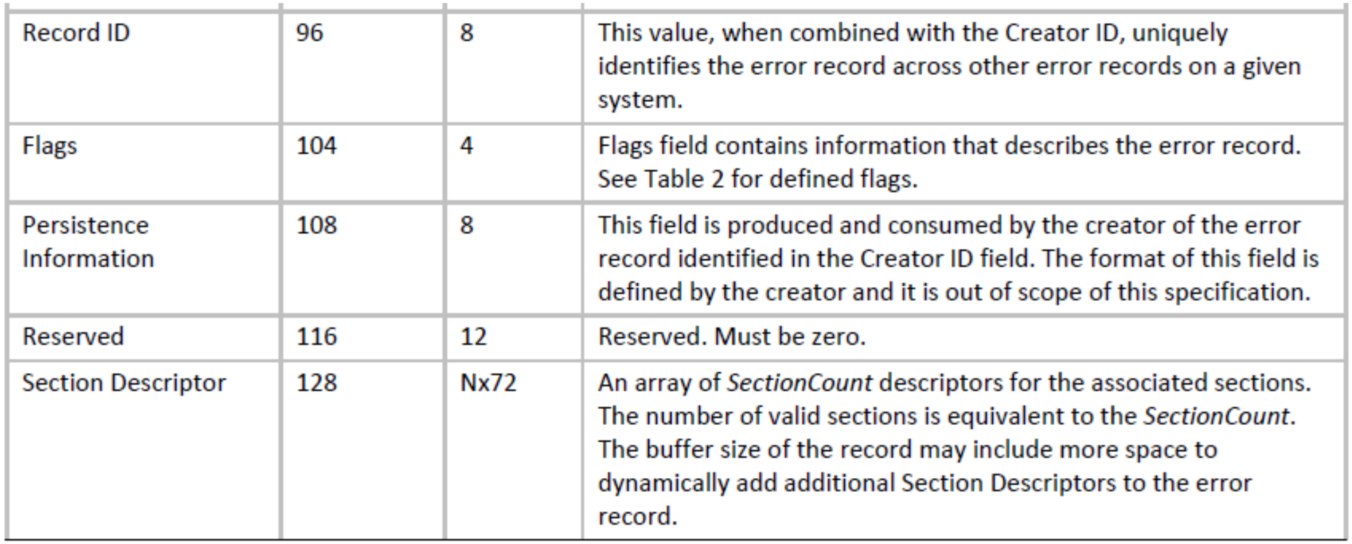
Mnemonic	Byte Offset	Byte Length	Description
Timestamp	24	8	The timestamp correlates to the time when the error information was collected by the system software and may not necessarily represent the time of the error event. The timestamp contains the local time in BCD format. • Byte 7 – Byte 0: • Byte 0: Seconds • Byte 1: Minutes • Byte 2: Hours • Byte 3: • Bit 0 – Timestamp is precise if this bit • is set and correlates to the time of the • error event. • Bit 7:1 – Reserved • Byte 4: Day • Byte 5: Month • Byte 6: Year • Byte 7: Century
Platform ID	32	16	This field uniquely identifies the platform with a GUID. The platform's SMBIOS UUID should be used to populate this field. Error analysis software may use this value to uniquely identify a platform.
Partition ID	48	16	If the platform has multiple software partitions, system software may associate a GUID with the partition on which the error occurred.
Creator ID	64	16	This field contains a GUID indicating the creator of the error record. This value may be overwritten by subsequent owners of the record.

UEFI CPER Format Definition – Header



Mnemonic	Byte Offset	Byte Length	Description
Notification Type	80	16	This field holds a pre-assigned GUID value indicating the record association with an error event notification type. The defined types are:

See definitions next slide



Record ID	96	8	This value, when combined with the Creator ID, uniquely identifies the error record across other error records on a given system.
Flags	104	4	Flags field contains information that describes the error record. See Table 2 for defined flags.
Persistence Information	108	8	This field is produced and consumed by the creator of the error record identified in the Creator ID field. The format of this field is defined by the creator and it is out of scope of this specification.
Reserved	116	12	Reserved. Must be zero.
Section Descriptor	128	Nx72	An array of *SectionCount* descriptors for the associated sections. The number of valid sections is equivalent to the *SectionCount*. The buffer size of the record may include more space to dynamically add additional Section Descriptors to the error record.

UEFI CPER Format Definition – Header Flags

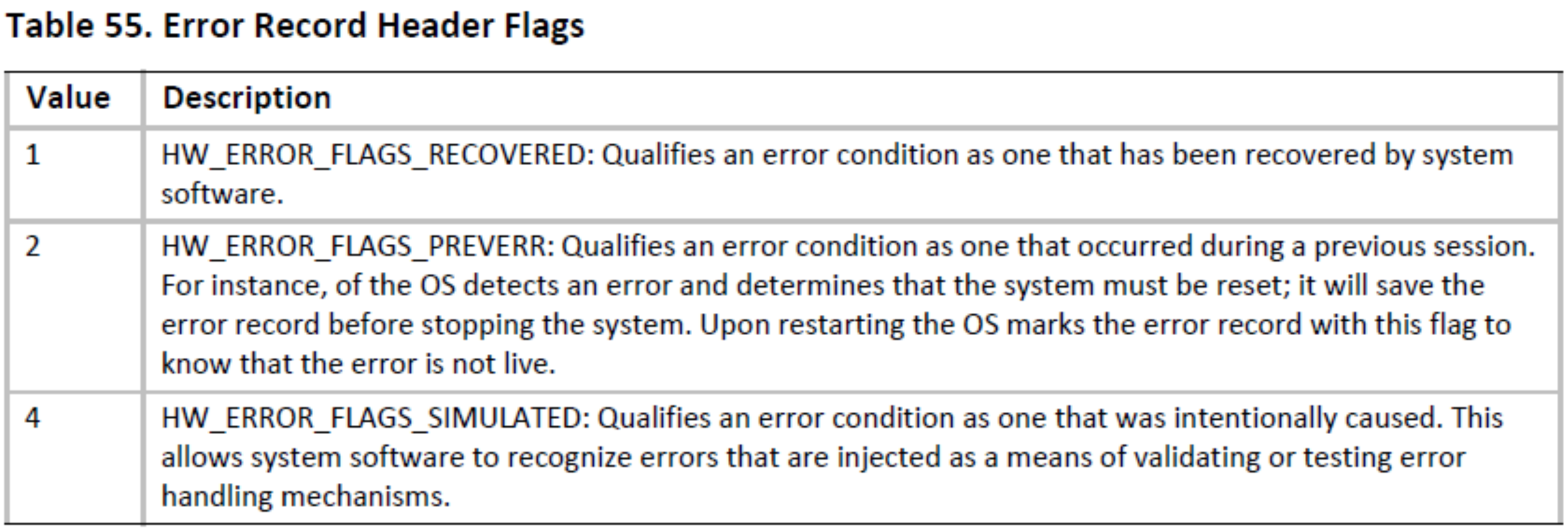


Table 55. Error Record Header Flags

Value	Description
1	HW_ERROR_FLAGS_RECOVERED: Qualifies an error condition as one that has been recovered by system software.
2	HW_ERROR_FLAGS_PREVERR: Qualifies an error condition as one that occurred during a previous session. For instance, of the OS detects an error and determines that the system must be reset; it will save the error record before stopping the system. Upon restarting the OS marks the error record with this flag to know that the error is not live.
4	HW_ERROR_FLAGS_SIMULATED: Qualifies an error condition as one that was intentionally caused. This allows system software to recognize errors that are injected as a means of validating or testing error handling mechanisms.

UEFI CPER Format Definition – Section Descriptor

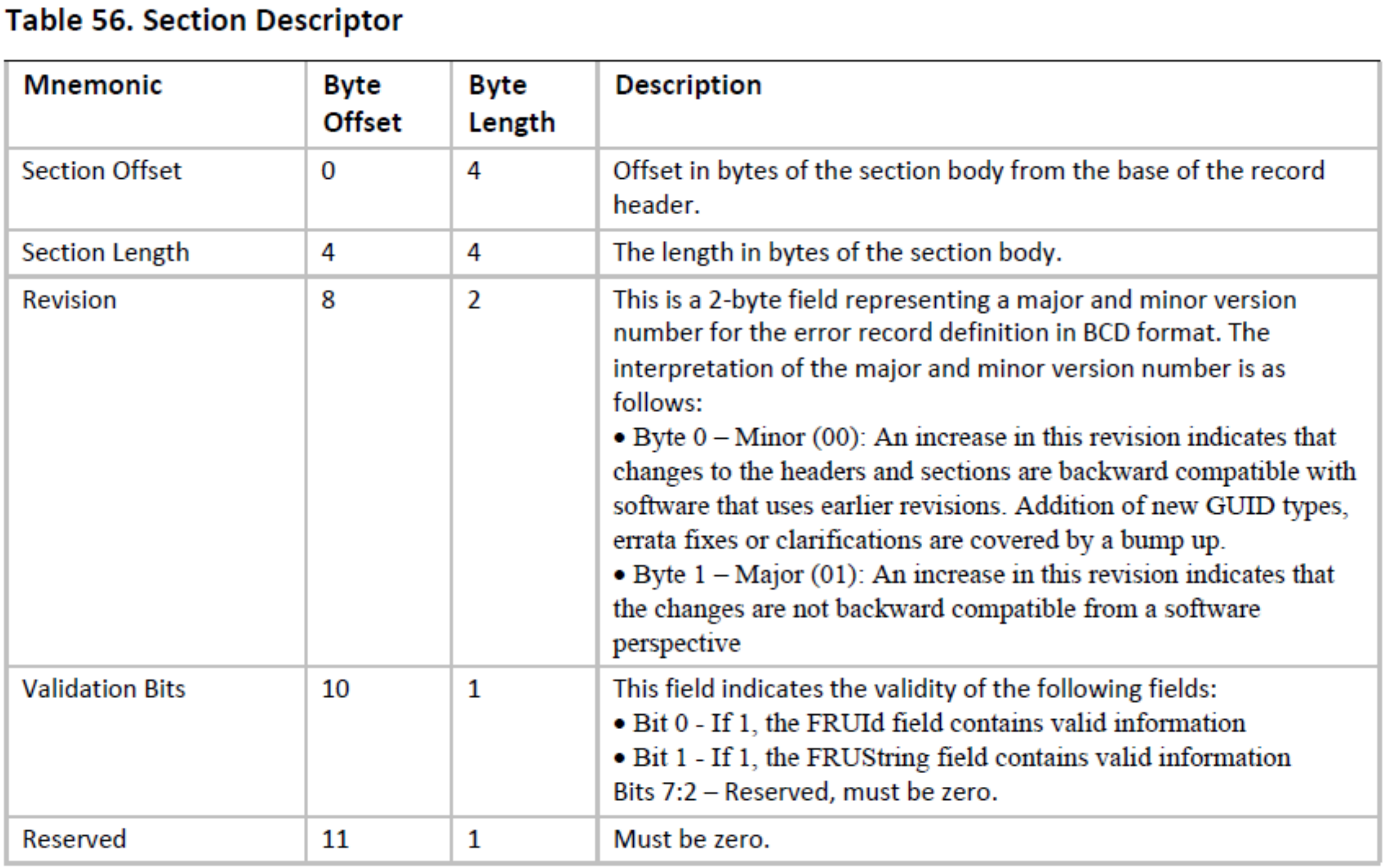
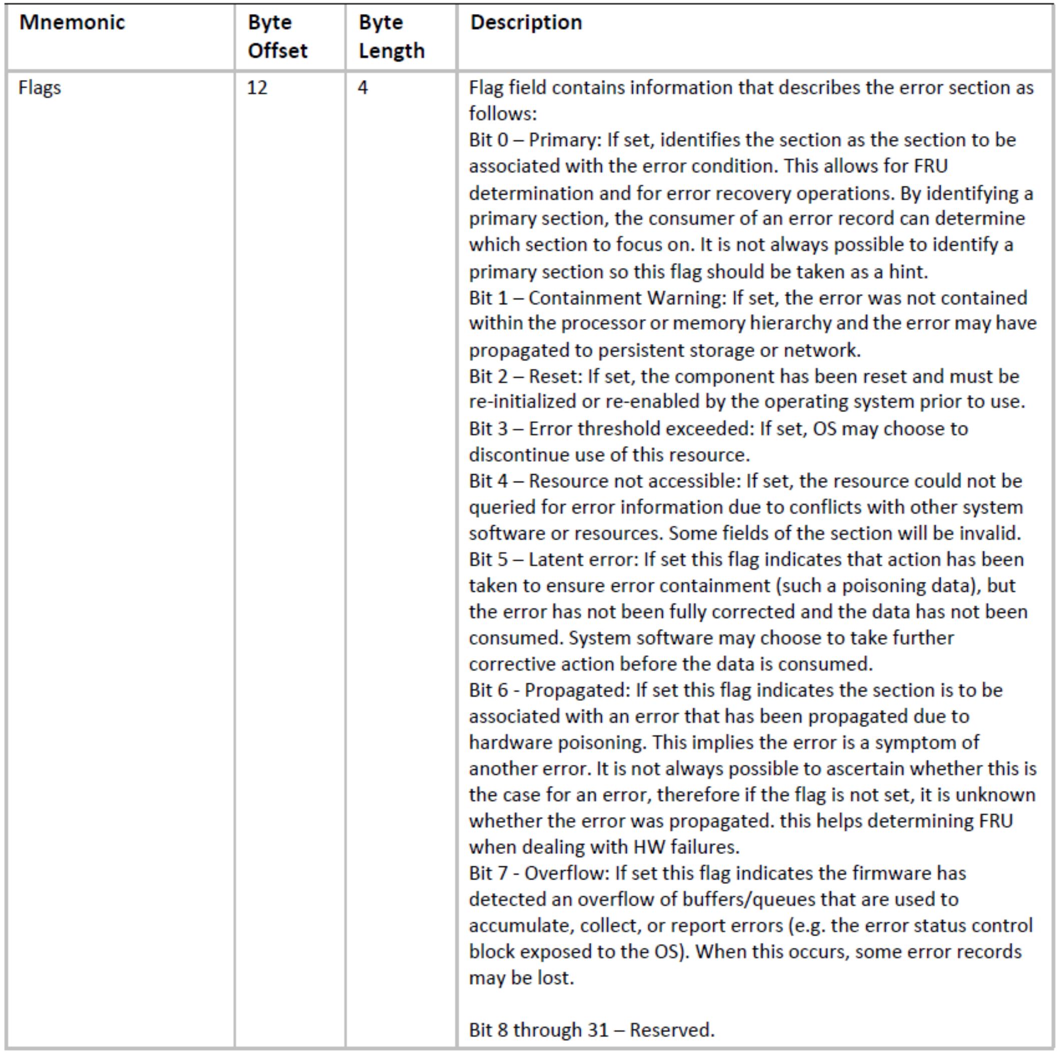


Table 56. Section Descriptor

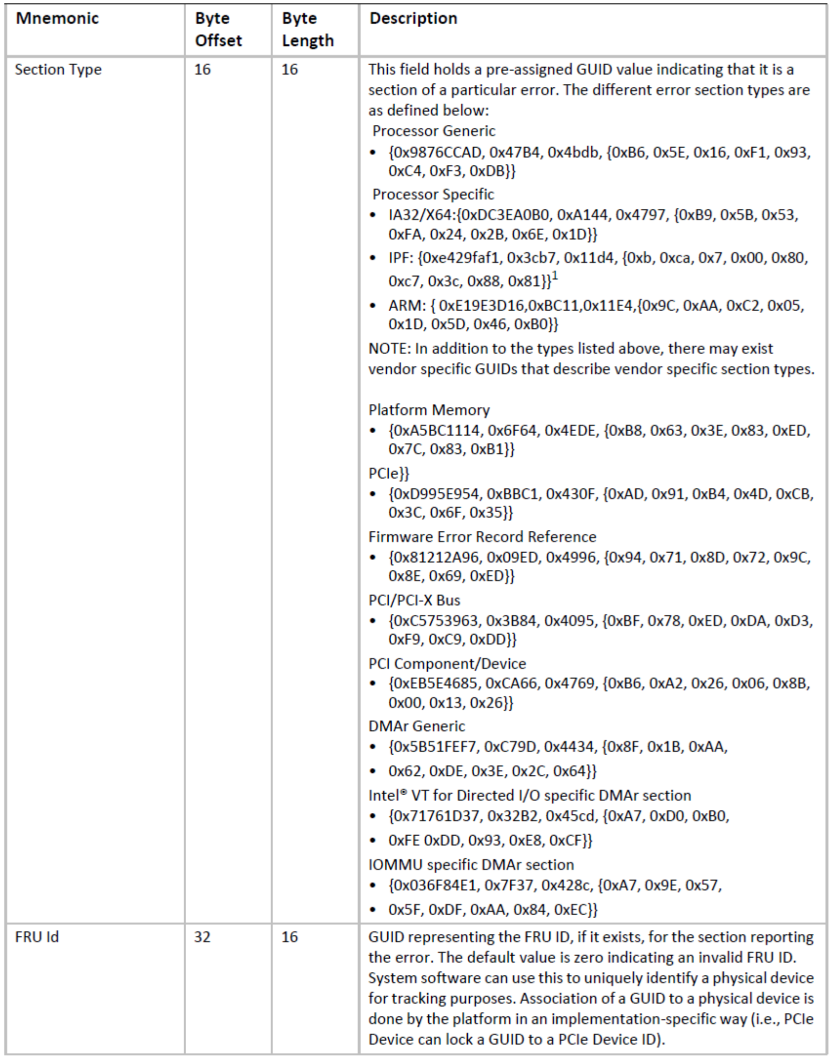
Mnemonic	Byte Offset	Byte Length	Description
Section Offset	0	4	Offset in bytes of the section body from the base of the record header.
Section Length	4	4	The length in bytes of the section body.
Revision	8	2	This is a 2-byte field representing a major and minor version number for the error record definition in BCD format. The interpretation of the major and minor version number is as follows: • Byte 0 – Minor (00): An increase in this revision indicates that changes to the headers and sections are backward compatible with software that uses earlier revisions. Addition of new GUID types, errata fixes or clarifications are covered by a bump up. • Byte 1 – Major (01): An increase in this revision indicates that the changes are not backward compatible from a software perspective
Validation Bits	10	1	This field indicates the validity of the following fields: • Bit 0 - If 1, the FRUId field contains valid information • Bit 1 - If 1, the FRUString field contains valid information Bits 7:2 – Reserved, must be zero.
Reserved	11	1	Must be zero.

UEFI CPER Format Definition – Section Descriptor



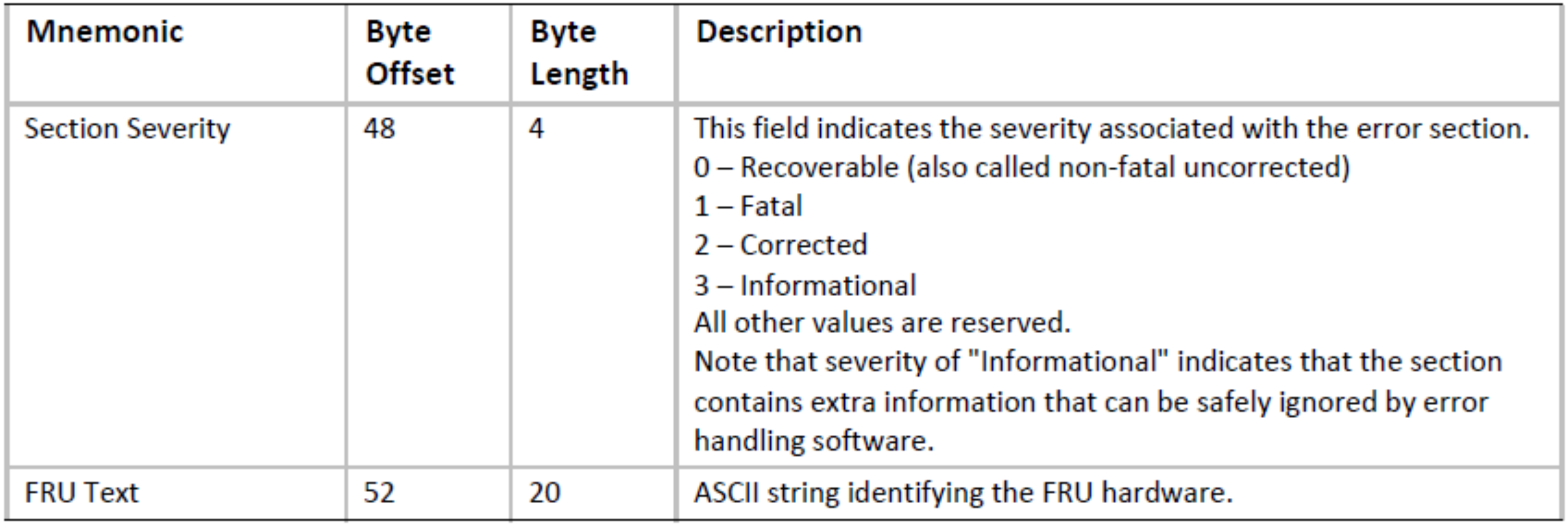
Mnemonic	Byte Offset	Byte Length	Description
Flags	12	4	Flag field contains information that describes the error section as follows: Bit 0 – Primary: If set, identifies the section as the section to be associated with the error condition. This allows for FRU determination and for error recovery operations. By identifying a primary section, the consumer of an error record can determine which section to focus on. It is not always possible to identify a primary section so this flag should be taken as a hint. Bit 1 – Containment Warning: If set, the error was not contained within the processor or memory hierarchy and the error may have propagated to persistent storage or network. Bit 2 – Reset: If set, the component has been reset and must be re-initialized or re-enabled by the operating system prior to use. Bit 3 – Error threshold exceeded: If set, OS may choose to discontinue use of this resource. Bit 4 – Resource not accessible: If set, the resource could not be queried for error information due to conflicts with other system software or resources. Some fields of the section will be invalid. Bit 5 – Latent error: If set this flag indicates that action has been taken to ensure error containment (such a poisoning data), but the error has not been fully corrected and the data has not been consumed. System software may choose to take further corrective action before the data is consumed. Bit 6 - Propagated: If set this flag indicates the section is to be associated with an error that has been propagated due to hardware poisoning. This implies the error is a symptom of another error. It is not always possible to ascertain whether this is the case for an error, therefore if the flag is not set, it is unknown whether the error was propagated. this helps determining FRU when dealing with HW failures. Bit 7 - Overflow: If set this flag indicates the firmware has detected an overflow of buffers/queues that are used to accumulate, collect, or report errors (e.g. the error status control block exposed to the OS). When this occurs, some error records may be lost. Bit 8 through 31 – Reserved.

UEFI CPER Format Definition – Section Descriptor



Mnemonic	Byte Offset	Byte Length	Description
Section Type	16	16	This field holds a pre-assigned GUID value indicating that it is a section of a particular error. The different error section types are as defined below: Processor Generic • {0x9876CCAD, 0x47B4, 0x4bdb, {0xB6, 0x5E, 0x16, 0xF1, 0x93, 0xC4, 0xF3, 0xDB}} Processor Specific • IA32/X64:{0xDC3EA0B0, 0xA144, 0x4797, {0xB9, 0x5B, 0x53, 0xFA, 0x24, 0x2B, 0x6E, 0x1D}} • IPF: {0xe429faf1, 0x3cb7, 0x11d4, {0xb, 0xca, 0x7, 0x00, 0x80, 0xc7, 0x3c, 0x88, 0x81}}[1] • ARM: { 0xE19E3D16,0xBC11,0x11E4,{0x9C, 0xAA, 0xC2, 0x05, 0x1D, 0x5D, 0x46, 0xB0}} NOTE: In addition to the types listed above, there may exist vendor specific GUIDs that describe vendor specific section types. Platform Memory • {0xA5BC1114, 0x6F64, 0x4EDE, {0xB8, 0x63, 0x3E, 0x83, 0xED, 0x7C, 0x83, 0xB1}} PCIe}} • {0xD995E954, 0xBBC1, 0x430F, {0xAD, 0x91, 0xB4, 0x4D, 0xCB, 0x3C, 0x6F, 0x35}} Firmware Error Record Reference • {0x81212A96, 0x09ED, 0x4996, {0x94, 0x71, 0x8D, 0x72, 0x9C, 0x8E, 0x69, 0xED}} PCI/PCI-X Bus • {0xC5753963, 0x3B84, 0x4095, {0xBF, 0x78, 0xED, 0xDA, 0xD3, 0xF9, 0xC9, 0xDD}} PCI Component/Device • {0xEB5E4685, 0xCA66, 0x4769, {0xB6, 0xA2, 0x26, 0x06, 0x8B, 0x00, 0x13, 0x26}} DMAr Generic • {0x5B51FEF7, 0xC79D, 0x4434, {0x8F, 0x1B, 0xAA, • 0x62, 0xDE, 0x3E, 0x2C, 0x64}} Intel® VT for Directed I/O specific DMAr section • {0x71761D37, 0x32B2, 0x45cd, {0xA7, 0xD0, 0xB0, • 0xFE 0xDD, 0x93, 0xE8, 0xCF}} IOMMU specific DMAr section • {0x036F84E1, 0x7F37, 0x428c, {0xA7, 0x9E, 0x57, • 0x5F, 0xDF, 0xAA, 0x84, 0xEC}}
FRU Id	32	16	GUID representing the FRU ID, if it exists, for the section reporting the error. The default value is zero indicating an invalid FRU ID. System software can use this to uniquely identify a physical device for tracking purposes. Association of a GUID to a physical device is done by the platform in an implementation-specific way (i.e., PCIe Device can lock a GUID to a PCIe Device ID).

UEFI CPER Format Definition – Section Descriptor



Mnemonic	Byte Offset	Byte Length	Description
Section Severity	48	4	This field indicates the severity associated with the error section. 0 – Recoverable (also called non-fatal uncorrected) 1 – Fatal 2 – Corrected 3 – Informational All other values are reserved. Note that severity of "Informational" indicates that the section contains extra information that can be safely ignored by error handling software.
FRU Text	52	20	ASCII string identifying the FRU hardware.