Use Cases for CXL RAS Firmware-First Error Handling

Manjunaatha Harapanahalli

Intel Corporation

the Future of Memory and Storage

Outline

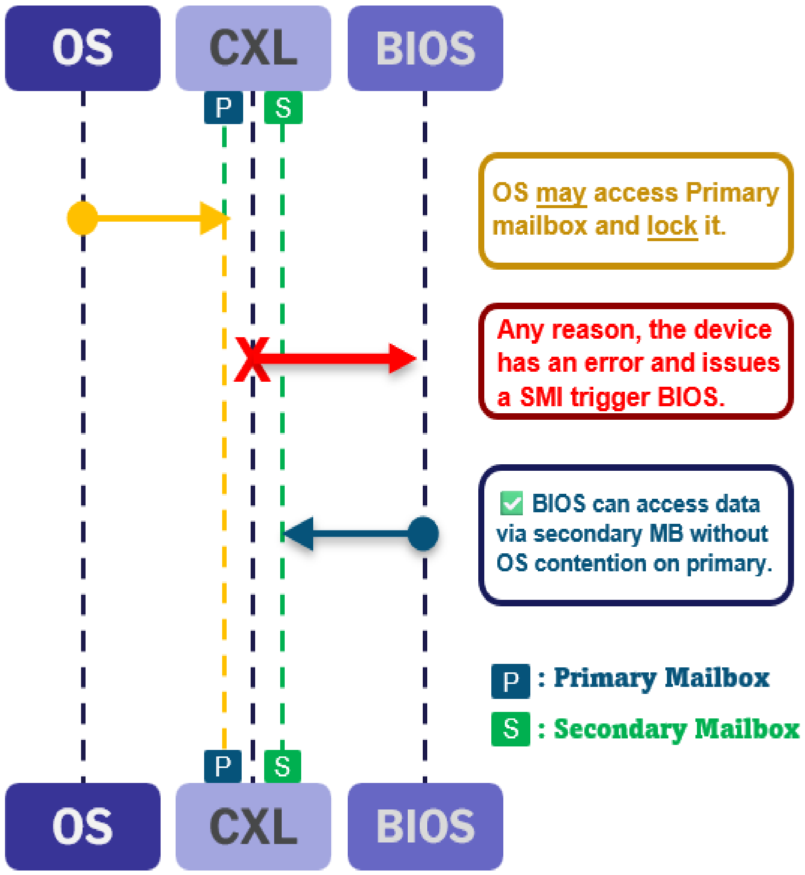
1. Introduction
2. Firmware-First Error Handling in CXL memory: Lessons Learned from Real-World Deployments
3. Impact of SMI Latency on System Performance
4. Common Error Signaling Protocols
5. Interpreting Global Unique Identifier (GUID) and Universal Unique Identifier (UUID) in CXL Specification for Common Platform Error Record(CPER)
6. Notifying the OS of Communication Failures Between CPU and CXL Devices During Boot and Run Time
7. Error pollution with CXL Errors

Introduction

- The CXL ecosystem comprises of multitude of component vendors like SoC, Memory, Storage, Networking, etc.
- The explosive growth of internet content and the resulting data storage and computation requirements has resulted in the deployment of heterogenous and complex solutions in the very large-scale data centers.
- These warehouse sized buildings are packed with server, storage and network hardware.
- Specifically, if there is an uncorrected fatal error detected by hardware that pose a containment risk. The system needs to be reset and restarted, if possible, to enable continued operation.
- The error affects the entire CXL device, a persistent/permanent memory device is considered to have experienced a dirty shut-down.

Lessons Learned from Real-World Deployments

- The idea of primary and secondary Mailbox (MB) is to have Firmware and Operating System (OS) NOT to step on each other.
- Firmware first support requires the CXL Memory device to implement a secondary mailbox. There was a challenge to get this support from CXL Independent Hardware Vendor(IHVs) in the initial stages.
- To support the engineering/debug effort, Intel Firmware added the option to use the primary Mailbox (MB) during runtime to enable the MEFN feature to validate with IHVs (without secondary MB support).
- The primary and secondary Mailbox (MB) queues are different, but the “Device Status Register” is common for both queues to feed the System Firmware (FW) or the OS.



OS may access Primary mailbox and lock it.

Any reason, the device has an error and issues a SMI trigger BIOS.

✓ BIOS can access data via secondary MB without OS contention on primary.

P : Primary Mailbox

S : Secondary Mailbox

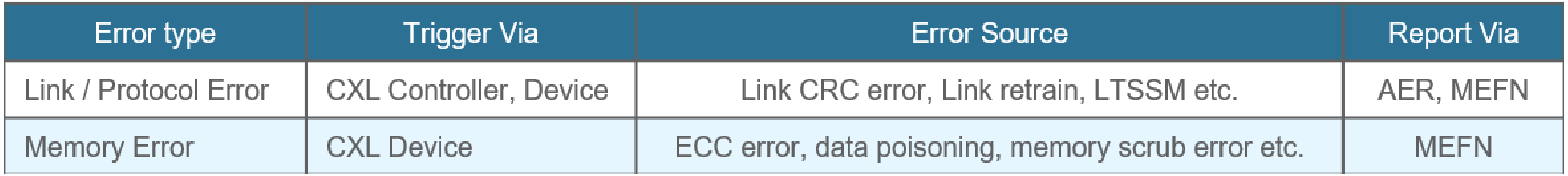
With Secondary

Impact of SMI Latency on System Performance

- Latency has been a known challenge for System Management Interrupt (SMI) handlers. Longer latencies can impact system performance.
- When enabling Memory Error Firmware Notification (MEFN) feature, System Management Model (SMM) latencies were initially high due to 8-bit register access for Mailbox (MB) transactions, error pollution (CXL Memory Error Reporting (MER) along with poison containment and Advanced Error Reporting(AER)).
- When creating Common Platform Error Record (CPER) the DVSEC registers need to be read multiple times to fill the CPER format, which resulted in higher SMI latencies.
- Caching the DVSEC registers and using 64-bit accesses resulted in significant reduction of SMI latency (in terms of ms).
- Alternative approaches such as using a separate controller (example: Baseboard Management Controller (BMC)) for handling CXL memory errors could be explored.

Common Error Signaling Protocols

- When a protocol error happens on a device there are 2 Common Platform Error Record (CPER) created
  - 1 for CacheMem($M) which outlines the CacheMem($M) side
  - 1 for CXL.io to provide the OS/external agent the details of the error as per the CPER format.
- CXL Memory Error Reporting (MER) does not result in an Advanced Error Reporting (AER)/CXL.io error.

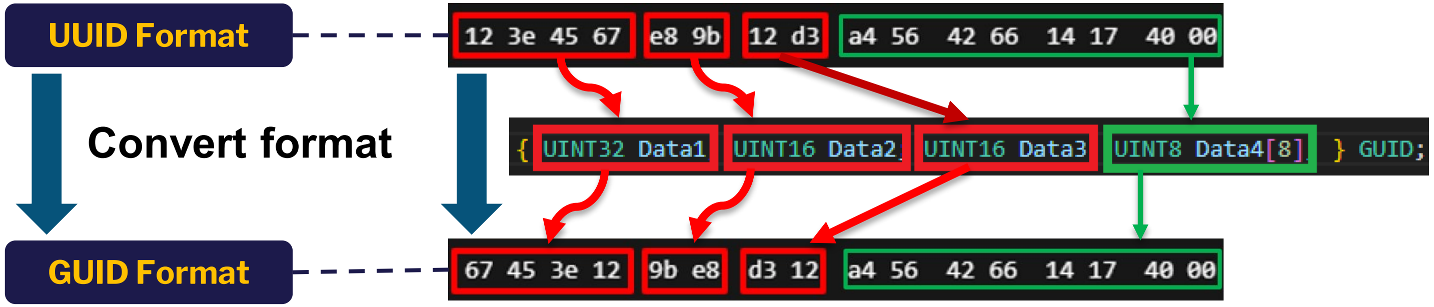


Error type	Trigger Via	Error Source	Report Via
Link / Protocol Error	CXL Controller, Device	Link CRC error, Link retrain, LTSSM etc.	AER, MEFN
Memory Error	CXL Device	ECC error, data poisoning, memory scrub error etc.	MEFN

Global Unique Identifier (GUID) and Universal Unique Identifier (UUID) in CXL Spec for Common Platform Error Record(CPER)

- The section type GUID in the CPER record's section descriptor references the first field from the CXL event's common event record format.

- It is important to note that the CXL UUID is transformed to a CPER GUID, where a GUID byte swaps the 1st double-word (32-bit), the 2nd and 3rd word (16-bit) compared to a UUID.



UUID Format

```
12 3e 45 67  e8 9b  12 d3  a4 56  42 66  14 17  40 00
```

Convert format

```
{ UINT32 Data1  UINT16 Data2  UINT16 Data3  UINT8 Data4[8]  } GUID;
```

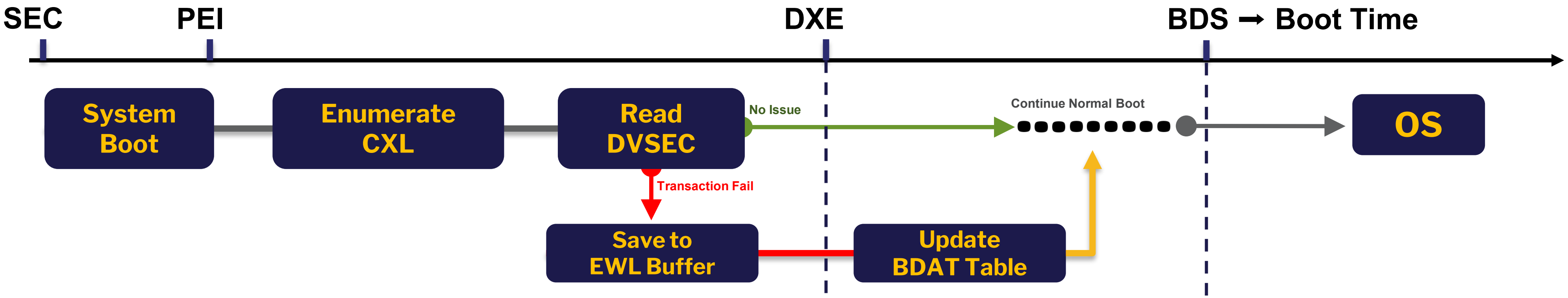
GUID Format

```
67 45 3e 12  9b e8  d3 12  a4 56  42 66  14 17  40 00
```

- The format of the CPER record mirrors the format of CXL event record after offset 0x10, that is the UUID that was translated into the CPER section type GUID is truncated leaving the remainder of the CXL record format as the CPER record format.

Notifying the OS of Communication Failures: CPU and CXL Devices During Boot Time

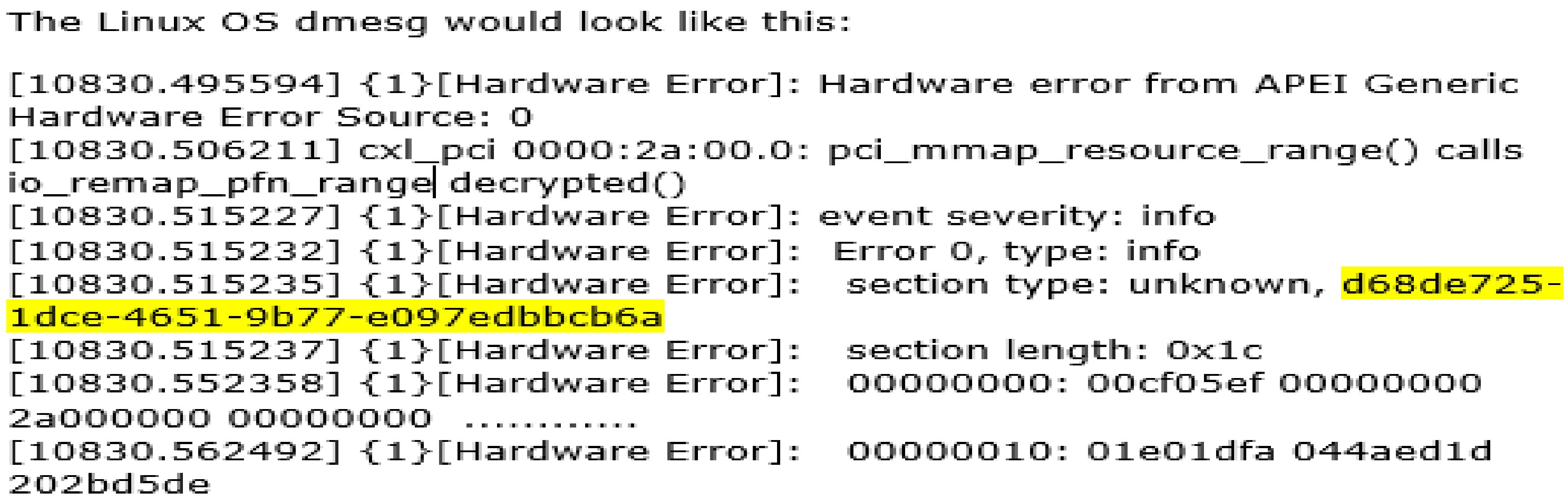
- During boot time communication failures like mailbox transaction not successful, failure to read DVSEC registers for MEFN setup between the CPU and CXL devices can be logged into Enhanced Warning Log(EWL) buffer.
- Later in the boot process the EWL buffer is used to update BIOS Data Attributes (BDAT) Table, which can be parsed by OS to expose the boot failures occurred during boot time.



Acronyms : **EWL** : Enhanced Warning Log **BDAT** : BIOS Data Attributes Table **SEC** : Security Phase **PEI** : Pre-EFI Initialization Phase **DXE** : Driver Execution Environment Phase **BDS** : Boot Device Selection Phase **SCI** : System Control Interrupt

Notifying the OS of Communication Failures: CPU and CXL Devices During Run Time

- During run time handling of the CXL errors by host using MEFN the CXL event records are retrieved via mailbox transactions and if there is any unsuccessful transaction that is logged using a non-standard GUID and a CPER is recorded.
- The CPER is notified to OS via SCI for user consumption to know about the failures that happened during runtime event.



The Linux OS dmesg would look like this:

```
[10830.495594] {1}[Hardware Error]: Hardware error from APEI Generic
Hardware Error Source: 0
[10830.506211] cxl_pci 0000:2a:00.0: pci_mmap_resource_range() calls
io_remap_pfn_range decrypted()
[10830.515227] {1}[Hardware Error]: event severity: info
[10830.515232] {1}[Hardware Error]:  Error 0, type: info
[10830.515235] {1}[Hardware Error]:   section type: unknown, d68de725-
1dce-4651-9b77-e097edbbcb6a
[10830.515237] {1}[Hardware Error]:   section length: 0x1c
[10830.552358] {1}[Hardware Error]:   00000000: 00cf05ef 00000000
2a000000 00000000  ............
[10830.562492] {1}[Hardware Error]:   00000010: 01e01dfa 044aed1d
202bd5de
```

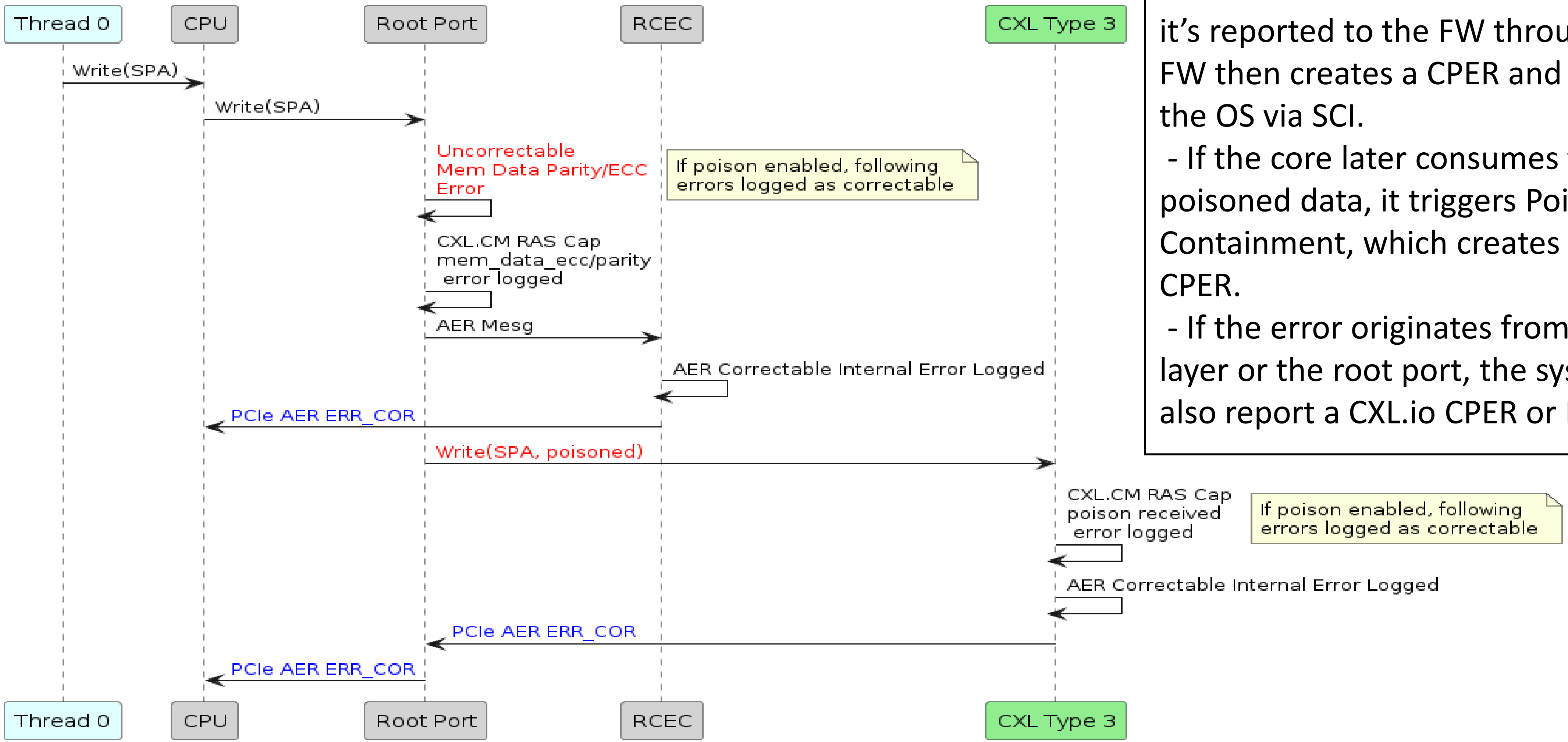
Error pollution with CXL Errors

- Firmware-First Error Flow -			
	Device-side / Memory Error	Root Port / Host-side Error	**Link / Protocol-level Error**
Error Trigger & Propagation	➢CXL memory error occurs. ➢Device Issue **MEFN** ➢Firmware logs **CPER** ➢If core consume > create another **Poison Containment** + **CPER**	➢Error occurrence like : - Data misrouting - Cache coherency violations ➢Root Port issue **SMI** ➢Firmware logs **CPER** ➢OS sees it as a host-initiated error.	➢Error occurs at link layer : - Parity error, CRC failure etc. ➢These can simultaneously trigger: - **.mem .cache** (DVSEC-RAS capability structure) - **.io** ➢Firmware logs **CPER** ➢Device May Issue **MEFN** ➢If the corrupted data propagates, it may also trigger **Poison Containment + CPER**
Error Impact & Behavior	👉 A **single device error** may results **2 reports. (MEFN + Poison Containment)**	👉 Device is innocent, but **host logic reports an error**	👉 This is the **most chaotic scenario**, because transit-level corruption pollutes multiple layers.

the Future of Memory and Storage

P.S. The same flows would work in the Flat2lm mode.

Poison Error chart



Thread 0 | CPU | Root Port | RCEC | CXL Type 3

Write(SPA)

Write(SPA)

Uncorrectable Mem Data Parity/ECC Error

If poison enabled, following errors logged as correctable

CXL.CM RAS Cap mem_data_ecc/parity error logged

AER Mesg

AER Correctable Internal Error Logged

PCIe AER ERR_COR

Write(SPA, poisoned)

CXL.CM RAS Cap poison received error logged

If poison enabled, following errors logged as correctable

AER Correctable Internal Error Logged

PCIe AER ERR_COR

PCIe AER ERR_COR

Thread 0 | CPU | Root Port | RCEC | CXL Type 3

Summary:

- If the error comes from a CXL device, it’s reported to the FW through MEFN. FW then creates a CPER and sends it to the OS via SCI.
- If the core later consumes the poisoned data, it triggers Poison Containment, which creates another CPER.
- If the error originates from the link layer or the root port, the system may also report a CXL.io CPER or PCIe AER.

the Future of Memory and Storage