# # Title:

EINJ Updates for CXL

# # Status:

Draft

# # Document:

ACPI Specification Version 6.*next*

# # License:

SPDX-License-Identifier: CC-BY-4.0

# # Submitter:

* Mahesh Natu (Intel)
* Shubhada Pugaonkar (Intel)
* Samer El-Haj Mahmoud (Arm)
* Thanu Rangarajan (Arm)
* Leo Duran (AMD)
* TianoCore Community ([https://www.tianocore.org](https://www.tianocore.org/))

# # Summary of the change

This ECR is related to a proposal to update the current EINJ table to support error injection on CXL. Specifically, the ECR includes the following changes:

1. Updates to section 18.6.1 (Error Injection Table (EINJ)) to support error injection on CXL.

# # Benefits of the change

Enables support for error injection on systems that support CXL.

# # Impact of the change

Platform firmware will have to understand and support the new format of the EINJ structure, and OS’s will also require a driver to support the new format. These are both fundamentally new code.

# # Detailed description of the change [normative updates]

Existing text

New text

~~Deleted text~~

**18.6 Error Injection**

…

**Table 18.29 Error Type Definition**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | | | **Description** |
| 0 | | | Processor Correctable |
| 1 | | | Processor Uncorrectable non-fatal |
| 2 | | | Processor Uncorrectable fatal |
| 3 | | | Memory Correctable |
| 4 | | | Memory Uncorrectable non-fatal |
| 5 | | | Memory Uncorrectable fatal |
| 6 | | | PCI Express Correctable |
| 7 | | | PCI Express Uncorrectable non-fatal |
| 8 | | | PCI Express Uncorrectable fatal |
| 9 | | | Platform Correctable |
| 10 | | | Platform Uncorrectable non-fatal |
| 11 | | | Platform Uncorrectable fatal |
|  | 12 |  | CXL.cache Protocol Correctable |
|  | 13 |  | CXL.cache Protocol Uncorrectable non-fatal |
|  | 14 |  | CXL.cache Protocol Uncorrectable fatal |
|  | 15 |  | CXL.mem Protocol Correctable |
|  | 16 |  | CXL.mem Protocol Uncorrectable non-fatal |
|  | 17 |  | CXL.mem Protocol Uncorrectable fatal |
| ~~12~~18:30 | | | RESERVED |
| 31 | | | Vendor Defined Error Type. If this bit is set, then the Error types and related data structures are defined by the Vendor, as shown in the Vendor  Error Type Extension Structure |

**NOTE:** CXL errors (Bits 17:12) are intended to target the CXL port (e.g., via Link or Protocol errors, and not actual Component errors).

**Table 18-30 SET\_ERROR\_TYPE\_WITH\_ADDRESS Data Structure**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| Error Type | 4 | 0x00 | Bit map Bitmap of error types to inject. Refer to *Error Type Definition*. This field is cleared by the platform once it is consumed. |
| Vendor Error Type Extension Structure Offset | 4 | 4 | Specifies the offset from the beginning of the table to the vendor error type extension structure. If no vendor error type  extension is present, bit31 in error type must be clear and this field must be set to 0. |
| Flags | 4 | 0x88 | Bit [0] – Processor Identification Field Valid  Bit [1]– Memory Address and Memory address Mask Field Valid.  **NOTE:** For CXL errors the Memory Address points to a CXL 1.1 compliant memory-mapped Downstream port.  Bit [2] – PCIe SBDF field valid.  **NOTE:** For CXL errors the SBDF points to a CXL 2.0 compliant Root port.  Bit [31:3] – RESERVED  This field is cleared by the platform once it is consumed. |
| Processor Error | | | |
| Processor ID | 4 | 0x0C12 | Optional field: on non-ARM architectures, this is the physical APIC ID or the X2APIC ID of the processor which is a target  for the injection; on ARM systems, this is the ACPI Processor UID value as used in the MADT. |
| Memory Error | | | |
| Memory Address | 8 | 0x1016 | Optional field specifying the physical address of the memory that is the target for the injection. Valid if Bit [1] of the Flags  field is set. |
| Memory Address Range | 8 | 0x1824 | If non-zero, provides a range mask for the Memory Address field. Valid if Bit [1] of the Flags field is set. |
| PCIe Error | | | |
| PCIe SBDF | 4 | 0x2032 | Byte 3 – PCIe Segment  Byte 2 – Bus Number  Byte 1:  Bits [7:3] Device Number  Bits [2:0] Function Number  Byte 0 – RESERVED |