ACPI for the Armv8-A RAS Extension and RAS System Architecture 2.0 BET1

Platform Design Document

Non-confidential

Notice
This document is a Beta version of a specification undergoing review by Arm partners. It is provided to give advanced information only.

Document number: DEN0085

Contents

Release information

Date	Version	Changes
2024/May/30	2.0 BET1	• Introduced CPER format for error nodes • Add timebase offset description
2023/May/30	2.0 BET0	• New AEST node type for proxy error nodes. • Introduced Error Mapping concept • (Section 2.1.5) Updated GIC structure to reference various structures using identifiers instead of offsets. • (Section 2.1.1.1) Updated the cache substructure to include reference to the Cache Identifier field in the PPTT Type 1 structures. • (Section 2.1.3.1) Updated the processor generic substructure to include a reference to any generic data that might apply to the error node being described. • (Section 2.2) Added field in the Node Interface structure to specify the addressing mode used for reporting addresses in ERR<n>_ADDR. • Added flag to indicate whether an error node can generate an FHI for UCs. • Fixed miscellaneous editorial issues. • Added new section on ACPI device object definition for error nodes. This is to support MSI programming requirements. • New AEST node type for PCIe root complex. • Updated language with correct terminology. • Added field for describing processor affinity of error nodes. • Added field for discovery of common error group-level registers, to support error nodes with > 4K blocks and to support the RASv2 RAS agent concept. • Introduced new interface type for single error record memory-mapped view. • Added clarifications on the base address field for various interface types. • Added an interface flag to indicate validity of the error group level register set base address field. • Updated CPER record format to include description of component type. • Add explicit support for large error groups and proxy nodes • Allow error injection and interrupt configuration registers to be disjoint from the main group register set.
2020/Sep/28	1.1	• External Release version 1.1. • (Section 2.1.5) Updated GIC structure to reference various structures using identifiers instead of offsets. • (Section 2.1.1.1) Updated the cache substructure to include reference to the Cache Identifier field in the PPTT Type 1 structures. • (Section 2.1.3.1) Updated the processor generic substructure to include a reference to any generic data that might apply to the error node being described. • (Section 2.2) Added field in the Node Interface structure to specify the addressing mode used for reporting addresses in ERR_ADDR.
2020/Aug/28	1	• External Release version 1.0.

Arm document reference: PRE-21585

version 5.0, March 2024

About this document

Terms and abbreviations

Term	Meaning
ACPI	Advanced Configuration and Power Interface
APIC	Advanced Programmable Interrupt Controller. This is a generic term used in ACPI for an interrupt controller.
ASL	ACPI Source Language
CPER	Common Platform Error Record. See [1].
GIC	Arm Generic Interrupt Controller
GSIV	Global System Interrupt Vector
IORT	I/O Remapping Table
MADT	Multiple APIC Description Table. The MADT describes an interrupt controller.
PE	Processing Element
PPTT	Processor Properties Topology Table
RAS	Reliability, Availability and Serviceability
SMMU	Arm System Memory Management Unit
SRAT	System Resource Affinity Table
TLB	Translation Lookaside Buffer
UE	Uncorrectable error
UID	Unique Identifier

References

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Unified Extensible Firmware Interface*. UEFI Forum.

[2] *Advanced Configuration and Power Interface Specification*. UEFI Forum.

[3] *Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile: DDI 0587C.c*. Arm Limited.

[4] *Arm® System Memory Management Unit Architecture Specification versions 3.0, 3.1 and 3.2: IHI 0070C.a*. Arm Limited.

[5] *Arm I/O Remapping Table: DEN0049*. Arm Limited.

[6] *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4: IHI 0069E (ID012119)*. Arm Limited.

[7] *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile: Arm DDI 0487E.a (ID070919)*. Arm Limited.

Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to errata@arm.com. Give:

- The title (ACPI for the Armv8-A RAS Extension and RAS System Architecture).
- The document ID and version (DEN0085 2.0 BET1).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Introduction

This specification provides a detailed description of ACPI [2] extensions required to enable kernel-first handling of errors in a system that supports the Armv8-A RAS extension and the RAS System Architecture defined by the *Arm Architecture Reference Manual Supplement Reliability, Availability and Serviceability (RAS) for A-profile Architecture* specification [3].

2 The Arm Error Source Table

The the Arm Architecture Reference Manual Supplement Reliability, Availability and Serviceability (RAS) for A-profile Architecture specification defines the term *node* to refer to a component that implements the RAS System Architecture defined in [3].

This document describes the Arm Error Source Table (AEST) in ACPI. The AEST provides an ACPI representation of nodes in a system that is based on [3].

The remainder of this document uses the term *error node* to refer to a node in [3], and the term *AEST node* to refer to the ACPI representation of the error node.

The AEST is described in Table 3.

Table 3: AEST format

Field	Byte length	Byte offset	Description
Header			
Signature	4	0	'AEST', Arm error source table
Length	4	4	Length of table in bytes
Revision	1	8	For this version of the specification, this field must be 2.
Checksum	1	9	The entire table must sum to zero.
OEM ID	6	10	OEM ID
OEM Table ID	8	16	For AEST, the table ID is the manufacturer model ID.
OEM Revision	4	24	OEM revision of the AEST table for the supplied OEM Table ID
Creator ID	4	28	The vendor ID of the utility that created the table.
Creator Revision	4	32	The revision of the utility that created the table.
Body			
Array of AEST node structures	–	36	Array of AEST node structures that are described in Table 4.

The format of the AEST node structure, or simply AEST node, is described in Table 4.

Table 4: AEST node structure

Field	Byte Length	Byte Offset	Description
Header			

Field	Byte Length	Byte Offset	Description
Type	1	0	AEST node type: 0x00 - Processor error node 0x01 - Memory error node 0x02 - SMMU error node 0x03 - Vendor-defined error node 0x04 - GIC error node 0x05 - PCIe error node All other values are reserved.
Length	2	1	Length of structure in bytes.
Reserved	1	3	Must be zero.
Offset to AEST node-specific data	4	4	Offset from the start of the AEST node to AEST node-specific data.
Offset to AEST node interface	4	8	Offset from the start of the AEST node to the AEST node interface structure.
Offset to AEST node interrupt array	4	12	Offset from the start of the AEST node to AEST node interrupt array.
AEST node interrupt array size	4	16	Number of entries in the interrupt array.
AEST node generic data			
Timestamp Rate	8	20	If the timestamp extension is implemented, and does not use the timebase of the system Generic Timer, as indicated by ERR<n>FR.TS== 0b10, this field indicates the timestamp frequency of the counter in Hertz. Else this field must be zero and the OS must ignore its contents.
Timebase offset	8	28	If the timestamp extension is implemented, and does not use the timebase of the Generic Timer, as indicated by ERR<n>FR.TS== 0b10, this field contains a signed 64-bit integer that represents the offset, in ns , between the timebase of the node timer and the timebase of the Generic Timer. The conversion between a reading in the node time axis into the Generic Timer time axis is achieved by the following equation: timestamp_generic_axis = node_counter_reading . (10^9 / timestamp_rate) + timebase_offset This field is ignored if ERR<n>FR.TS $\neq$ 0b10
Error injection countdown rate	8	36	If Common Fault Injection Model Extension is supported as indicated by ERR<n>FR.INJ != 0b00, this field provides the rate in Hertz at which the Error Generation Counter decrements. Otherwise this field must be zero and the OS must ignore its contents.

Field	Byte Length	Byte Offset	Description
AEST node-specific data	–	Offset for AEST node- specific data	
AEST node interface	–	Offset for AEST node interface	
AEST node interrupt array	–	Offset for AEST node interrupt array	

AEST nodes provide a description of error nodes that are based on the Arm RAS System Architecture described in [3]. The AEST node is composed of the following parts:

- A header that is described in Table 4.
- A set of common fields that is described in Table 4.
- A component-specific section that associates the AEST node to the component in the system that is associated with the error node.
- A section that describes the interfaces that are associated with the error node.
- A section describing interrupts that are associated with the error node.

2.1 Component types

Each error node in a system is associated with a component. The AEST node that represents an error node must provide information in its AEST node-specific data section to describe both the error node and the component that the error node is associated with. This information enables the OS to discover this association. The following components are currently supported:

- PE (Processor structure)
- Memory (Memory controller structure)
- SMMU (SMMU structure)
- Vendor-specific (Vendor-defined structure)
- GIC (GIC structure)
- PCIe root complex (PCIe Root Complex structure)
- Proxy error node (Proxy error structure)

The tables that are described in these sections provide the structure for the AEST node-specific data section of an AEST node.

2.1.1 Processor structure

The Processor structure describes error nodes that are related to the processor and its internal components. The Processor structure is described in Table 5. The ACPI Processor Properties Topology Table (PPTT) table is used to identify processors whose error nodes are described in the processor structures. This specification works with PPTT table revision 3 or later.

Table 5: Processor structure

Field	Byte Length	Byte offset	Description
ACPI processor ID	4	0	Processor ID of node. For this revision of this specification, this field represents the _UID of the processor. This field must be set to 0 and ignored if this is a global or shared error node, as specified by the Flags field.
Resource Type	1	4	Specifies which resource within the processor this error node pertains to. 0x0 - Cache 0x1 - TLB 0x2 - Generic Other values are reserved.
Reserved	1	5	Reserved, must be zero.
Flags	1	6	Flags associated with this structure. See Table 6.
Revision	1	7	For this version of the spec, this field must be set to 0.
Processor affinity level indicator	8	8	Processor affinity descriptor for the resource that this error node pertains to. This field is only valid if the interface type of this AEST node is set to SR (see Section 2.2). This field must be ignored if the shared resource flag is set to 0. See Table 6. This field must match the ERRDEVAFF register defined in [3]. This field has been deprecated in version 2.0 of this specification and must be set to 0 by the platform and ignored by the OS. The OS should instead use the Processor affinity field in the AEST node interface sub-structure described in Table 17 to determine the processor affinity of this error node.
Resource-specific data			
Resource substructure	–	16	Processor resource whose error node is being described by this AEST node.

Table 6: Processor structure flags

Field	Bit Length	Bit offset	Description
Global	1	0	This flag is an indication that this error node is global for this resource type. A global error node is a single representative of error nodes of this resource type for all processors in the system. 0b - This is a dedicated error node. 1b - This is a global error node.

Field	Bit Length	Bit offset	Description
Shared	1	1	This flag is an indication that this AEST node represents an error node present on a resource that is shared by multiple processors in the system. The identity of the processors that are sharing this resource is specified in the processor affinity level indicator field. 0b - This AEST node represents a resource that is private to the specified processor. 1b - This AEST node represents a resource that is shared by multiple processors.
Reserved	6	2	Reserved, must be zero.

2.1.1.1 Processor cache resource substructure

The cache resource substructure describes a cache within the processor whose error node is being defined in the parent processor structure. The cache substructure provides a reference to the Type 1 structure in the PPTT table that describes the cache.

Table 7: Processor Cache resource substructure

Field	Byte Length	Byte offset	Description
Cache reference	4	0	Reference to the cache structure in the PPTT table that describes this cache. This field must match the Cache ID field of the cache structure. The PPTT table must support valid Cache IDs for this substructure to be supported.
Reserved	4	4	Reserved, must be zero.

2.1.1.2 Processor TLB resource substructure

This substructure is intended for describing error nodes associated with TLBs.

Table 8: Processor TLB resource substructure

Field	Byte Length	Byte offset	Description
TLB reference	4	0	Reference to the TLB where this error node is present. This field must be set to the level of the TLB.
Reserved	4	4	Reserved, must be zero.

2.1.1.3 Processor generic resource substructure

This substructure is intended for implementations that have a generic, processor-wide error node that caters to multiple resources in the processor. The exact interpretation of the error record information is left to the OS drivers that are specific to the processor. Software must consult the MIDR register of the current processor implementation to understand which resources are being handled by the error node that is being described.

Table 9: Processor generic resource substructure

Field	Byte Length	Byte offset	Description
Data	4	0	Vendor-defined supplementary data. Set to zero if not used.

2.1.2 Memory Controller structure

The Memory controller structure is described in Table 10.

Table 10: Memory Controller structure

Field	Byte Length	Byte Offset	Description
Proximity domain	4	0	SRAT proximity domain. Must be set to 0 and ignored if the SRAT table is not present.

2.1.3 SMMU structure

The SMMU [4] structure is described in Table 11. An SMMU implementation can have one or more error nodes associated with it. The error node might be associated with a discrete internal component or unit within the SMMU, for example the central control unit or a TLB. Some of these units, for example a TLB, might also be associated with the Stream IDs that define a device or a set of devices that interface with the SMMU, for example Root Complexes. Therefore, the ACPI description of the SMMU error node must include references to the SMMU, and the device or devices that interface with the internal unit of the SMMU that is being described in this AEST node. The reference to the device or devices serves as a proxy for the SMMU internal unit or component.

This structure only works with IORT tables with nonzero revision numbers.

Table 11: SMMU structure format

Field	Byte Length	Byte Offset	Description
IORT node reference	4	0	Reference to the IORT [5] table node that describes this SMMU. The reference must match the Identifier field of the SMMU node.
SMMU-specific data			
Subcomponent reference	4	4	Reference to the IORT table node that is associated with the sub-component within this SMMU. This reference must point to a Root Complex or Named Component node that is associated with this SMMU subcomponent. If this subcomponent is a part of the SMMU itself, then this field is a reference to the IORT table node that describes the SMMU. The reference must match the Identifier field of the IORT node.

2.1.4 Vendor-defined structure

The Vendor-defined structure is described in Table 12. An OSPM might log this structure as raw data, or offer them to vendor-specific drivers where appropriate.

Table 12: Vendor-defined structure format

Field	Byte Length	Byte Offset	Description
Hardware ID	8	0	ACPI Hardware ID of the component.
Unique ID	4	8	The ACPI Unique identifier of the component. If there are multiple instances of this component, this field helps to identify a specific instance.
Vendor- specific data	16	12	Vendor-specific data, for example to identify this error source.

2.1.5 GIC structure

The GIC [6] structure is described in Table 13. A GIC implementation might have one or more internal error nodes on one or more internal interfaces.

Table 13: GIC structure format

Field	Byte Length	Byte Offset	Description
Interface type	4	0	Type of GIC interface that associated with this error node. 0x0 - GIC CPU (GICC) 0x1 - GIC Distributor (GICD) 0x2 - GIC Redistributor (GICR) 0x3 - GIC ITS (GITS) Other values are reserved.
Instance Identifier	4	4	Identifier for the interface instance. If the interface type is GICC, then this field represents the ACPI UID of the processor that this GICC interface is associated with. If the interface type is GICD, then this field represents the GIC ID of the GICD structure in the ACPI MADT table that describes this GIC Distributor. If the interface type is GITS, then this field represents the GIC ITS ID field of the GIC ITS structure in the ACPI MADT table that describes this GIC ITS interface. If the interface type is GICR, then this field represents the ACPI UID of the processor that this GIC Redistributor is associated with.

2.1.6 PCIe Root Complex structure

The PCIe Root Complex structure is described in Table 14, and is used to describe error nodes on PCIe Root Complexes in the system. A PCIe Root Complex is defined as any logical unit that operates as a bridge between the system and the PCIe sub-system. This specification assumes that PCIe Root Complexes are always associated with an SMMU and thus described in the ACPI IORT table [5].

Table 14: PCIe Root Complex structure format

Field	Byte Length	Byte Offset	Description
IORT node reference	4	0	Reference to the IORT [5] table node that describes this PCIe Root Complex. For IORT tables implementations based on IORT specification issues E and later, this field must match the Identifier field of the RC node in the IORT. For IORT tables implementations based on earlier issues of the IORT specification, this field must be set to the offset of the RC node from the start of the IORT table.

2.1.7 Proxy error structure

In the RASv2 architecture [3], the status of the records within a Group G1 can be subsumed as a node N2 with a single record in a group from an upstream RAS Agent. The node N2 is a proxy of G1. The status of N2 is equal to the logical AND of all records in G1. The status of N2 is represented as a single bit in the ERRGSR (with index equal to *start error record index*) of the group where N2 is present

The proxy node is described in the AEST via Table 15. The proxy node entry contains the required information for the AEST entry of the proxied node to be located.

Note that an AEST node entry is uniquely identified by the {Base address, Start error record index} tuple.

Table 15: Proxy node structure format

Field	Byte Length	Byte Offset	Description
node address	8	0	The address of the first error record in the group that the proxied node belongs to.

2.2 Interface

The AEST Interface structure describes the interface that the error node supports and the properties of that interface. The interface structure is present at the interface offset field that is defined in the AEST node header. The AEST node header is described in Table 4.

Error node interfaces can have the System register (SR), the memory mapped (MMIO) , or the single record memory-mapped view, as described in [3].

Error nodes with the memory-mapped view can be of two types, depending on the location of the error group level registers:

1. Legacy type, where the error group level registers are located in the same 4KiB block as the error records.
2. Detached register type, where sections of the register interface may be absent, or at a location separate from the error group. These can represent error nodes from large error groups (of 16KiB or 64KiB group format).

The RASv2 Architecture [3] mandates the fault injection registers to be separate from the error group. For the memory-mapped view, the RASv2 Architecture mandates the Group status register and interrupt configuration register to be at a particular offset (which is a function of the Group format) from the start of the memory-mapped view. For convenience, the Table 17 optionally lists the addresses of the Group Status register set and Interrupt Configuration register set. Note that some IP may deviate from the RASv2 mandate and present the Group Status register set and Interrupt Configuration register set at offsets that differ from the RASv2 specification, in that case the base address fields in Table 17 are mandatory.

The AEST interface node describes all of these properties to the OS.

Table 17 describes the format of the interface entries.

The RASv2 architecture [3] allows for Error Group with a 4KiB, 16KiB, or 64KiB format. The size of the interface structure depends on the Error Group format. The interface structure (Table 17) has 3 fields with size dependant on the Error Group format, these are:

- Error record implemented,
- Error group-based status reporting supported,
- Addressing mode.

Bit[n] of these bit-fields represents the error record *n* in the error group that this node is part of. Only error records that are part of this node are representable in the bit-fields, all other bits must be zero. The size of the bit-fields depends on the Group format:

Group format	bit-field size	maximum number of error records
4KiB	64bit (8B)	56
16KiB	256bits (32B)	224
64KiB	896bits (112B)	896

As a consequence, the size of Table 17 will vary with the Group format. Note the *gf* parameter, defined in the Group format field description, that parametrizes the size of Table 17.

Table 17: AEST node interface structure

Field	Byte Length	Byte Offset	Description
Interface type	1	0	Interface type: 0x0 – SR view 0x1 – Memory-mapped view 0x2 – Single record Memory-mapped view All other values are reserved.
Group format	1	1	Group size: 0x0 – 4KiB: gf = 1 0x1 – 16KiB: gf = 4 0x2 – 64KiB: gf = 14 All other values are reserved. This field must be set to 0x0 for interface type 0x2.
Reserved	2	2	Must be zero.
Flags	4	4	Flags associated with this AEST node interface. See Table 18.

Field	Byte Length	Byte Offset	Description
Base Address	8	8	Base address of the first error record of the error group that this node belongs to, if the interface type is 0x01. If the interface type is 0x02, this field represents the base address of the only error record of the error node. This field is not valid if interface type is 0x0.
Start error record index	4	16	Zero-based index of first standard error record that belongs to this error node. This value must lie in the range 0-(N-1). If the interface type is 0x0, N is the value read from the ERRIDR_EL1 register. If the interface type is 0x1, then N is the value read from the ERRDEVID register. If interface type is 0x02, this field must be set to 0, and the OS should ignore it.
Number of error records	4	20	Number of error records in this error node. This includes both implemented and unimplemented error records. This field must be set to 1 for the single record memory-mapped view.
Error record implemented	8.gf (see Group format)	24	This bitmap indicates which of the error records within this error node are implemented and must be polled for error status. Bit[n] of this field pertains to error record corresponding to index n in this error group. Bit[n] = 0b: Error record at index n is implemented. Bit[n] = 1b: Error record at index n is not implemented. If interface type is 0x02, this field must be set to 0, and the OS should ignore it.
Error group-based status reporting supported	8.gf (see Group format)	24 + 8.gf (see Group format)	This bitmap indicates which error records within this error node support error status reporting using ERRGSR register. Bit[n] of this field pertains to error record corresponding to index n in this error group. Bit[n] = 0b: Error record at index n supports error status reporting through ERRGSR.S. Bit[n] = 1b: Error record at index n does not support error reporting through the ERRGSR.S bit If this error record is implemented, then it must be polled explicitly for error events. This field only applies to interface type 0x0 when the PE implements FEAT_RASv2. The field is present for interface types 0x1 and 0x2 ,with its size defined by gf, the OS must ignore its contents

Field	Byte Length	Byte Offset	Description
Addressing mode	8.gf (see Group format)	24 + 16.gf (see Group format)	This bitmap specifies the addressing mode used by each error record within this error node to populate the ERR<n>_ADDR register. Bit[n] of this field pertains to error record corresponding to index n in the error group. Bit[n] = 0b: Error record at index n reports System Physical Addresses (SPA) in the ERR<n>_ADDR register. Bit[n] = 1b: Error record at index n reports error node-specific Logical Addresses (LA) in the ERR<n>_ADD register. OS must use other means to translate the reported LA into SPA.
ACPI Arm error node device	4	24 + 24.gf (see Group format)	This field must be set to the ACPI _UID field of the Arm error node device in DSDT that describes this error node. This field must be ignored if the ACPI device valid flag is set to 0.
Processor affinity	4	28 + 24.gf (see Group format)	Indicates the processor or group of processors that this error node is associated with. This field must be set to the ACPI _UID of the processor or processor container that this error node has affinity with. The affinity type flag indicates whether the _UID value belongs to a processor or group of processors.
Base address of Error group level register set	8	32 + 24.gf (see Group format)	Address of the group status registers . The group status registers are defined in [3]. This address points to the ERRGSR register. This field must be ignored if: • flags[4] is set to 0b, or • interface type is 0x0 or 0x2.
Fault injection registers base	8	40 + 24.gf (see Group format)	Address of the fault injection registers. These registers are defined in [3]. This address points to the base of the Fault injection group, which starts with the register ERR0PFGF, see [3] . This field must be ignored if: • flags[5] is set to 0b, or • interface type is 0x0.
Interrupt configuration registers base	8	48 + 24.gf (see Group format)	Address of the interrupt registers for this node. This address points to the ERRFHICR0 register, see [3]. This field must be ignored if: • flags[6] is set to 0b, or • interface type is 0x0 or 0x2.

Table 18: AEST node interface flags

Field	Bit Length	Bit Offset	Description
Shared interface	1	0	0b - This AEST node interface is private to an error node. 1b - This AEST node interface is shared. For error nodes on processor caches, the sharing is restricted to the processors that share the indicated cache. This flag only applies when interface type = 0x0.
Clear MISCx after logging	1	1	Indicates whether the MISCx registers must be cleared after their contents have been logged. 0b - Do not clear MISCx after logging. 1b - Clear MISCx after logging.
Arm error node device valid	1	2	This fields indicates whether there is an ACPI Arm error node device object in DSDT for this error node. Section Section 3 provides details on the Arm error node device. 0b - This error node does not have an ACPI Arm error node device in DSDT. 1b - This error node has an ACPI Arm error node device in DSDT.
Affinity type	1	3	Specifies whether the _UID value set in the Processor affinity field identifies a processor or a group of processors. 0b: This node is associated with a single processor whose _UID is specified in the Processor affinity field. 1b: This node is associated with a group of processors, and the Processor affinity field points to a processor container.
Error group address field valid	1	4	Indicates whether the base address of the error group level register set is valid. 0b: The base address is invalid. 1b: The base address is valid. If the base address is invalid, the OS must consider the offset of the group status registers to be in the standard 4KiB error group format of the memory-mapped view [3].
Fault injection address valid	1	5	Indicates whether the base address of the fault injection register set is valid. 0b: The base address is invalid. 1b: The base address is valid. If the base address is invalid, the fault injection group is nonexistent.
Interrupt configuration address valid	1	6	Indicate whether the interrupt configuration registers base is valid. 0b: The base address is invalid. 1b: The base address is valid. If the base address is invalid, the OS must consider the offset of the Interrupt configuration registers to be in the standard 4KiB error group format of the memory-mapped view [3].
Reserved	25	7	Reserved, must be zero.

2.3 Interrupts

Error nodes can generate three kinds of interrupts:

- Error Recovery Interrupt (ERI)
- Fault Handling Interrupt (FHI)
- Critical Error Interrupt (CEI)

Only the first two types, ERI and FHI, are represented in ACPI. This is because critical error interrupts are intended for notifying system controllers of the error event, instead of application processors under the control of the OS.

If ERI and FHI are combined into the same interrupt, then the firmware must present one interrupt structure each, and these structures must have identical GSIV values.

Table 19 describes the interrupt structures that are used to represent error node interrupts to the OS. These structures form the entries of the AEST node interrupt array. The array can be found through the AEST interrupt array offset that is defined in the AEST node header. The AEST node header is described in Table 4.

Table 19: AEST Node Interrupt structure

Field	Byte Length	Byte Offset	Description
Interrupt type	1	0	Interrupt type: 0x0 – Fault Handling Interrupt 0x1 – Error Recovery Interrupt All other values are reserved.
Reserved	2	1	Must be zero.
Interrupt Flags	1	3	Bits [31:2]: Must be zero. Bit 0: Trigger type • 0b – Interrupt is edge-triggered • 1b – Interrupt is level-triggered This field is valid only for wire-based interrupts. Bit 1: FHI on UE • 0b – This error node supports FHI for uncorrectable errors (UEs). • 1b – This node does not support FHI for uncorrectable errors (UEs).
Interrupt GSIV	4	4	GSIV of interrupt, if interrupt is an SPI or a PPI. Must be zero if the interrupt is not wire-based. If GSIV is 0, then MSI must be used instead.
Reserved	4	8	Reserved, must be zero.

3 Representation of Error Nodes in DSDT

The AEST describes generic properties of error nodes based on [3]. Error nodes might additionally be described as ACPI device objects in the DSDT. The error node can be viewed as a pseudo-device as it has a memory region and interrupts associated with it.

The ACPI device representation is useful for describing additional attributes and properties of the error node.

Note that the ACPI Arm error node device is an optinal extension to the mandatory description in the AEST. It cannot be used for overriding the properties described in the AEST.

3.1 ACPI Hardware Identifier for the ACPI Arm Error Node Device

The ACPI Arm error node device is assigned an ACPI _HID value of “ARMHE000”.

```
Device(ERR0) { // Arm error node device instance

      Name(_HID, "ARMHE000")
      Name(_UID, 0)
      Name(_STR, Unicode("Arm error node 0"))
}
```

3.2 MSI Support

The ACPI Arm error device object is useful for describing the MSI routing for error nodes. The Arm error device object must also be described in the IO topology of the system, from where the MSI routing can be obtained. The ACPI IORT table describes the IO topology of the system, where the error device object must appear as a Named Component. See [5] for more details on the IORT table and the Named Component.

4 CPER Format for Arm RAS System Architecture

This section describes how error nodes based on [3] are represented in ACPI BERT and CPER records.

4.1 Representing RAS error nodes in the BERT

Representing error node contents in BERT will require expressing them in the CPER first.

4.2 Representing Arm error nodes in CPER

CPER is described in the UEFI specification. (see Appendix N of [1]). CPER records are composed of a header, and a set of sections. Each section describes information relevant to an error event. There is at least one section, but there can be more. The CPER specification includes sections that are generic, such as the generic processor error section, and sections that are architecture-specific, such as the Arm or IA64 error sections.

CPER records are used by the system firmware at boot-time for specifying boot-time errors. When used in this manner, the CPER records are presented using the ACPI BERT table. CPER records are also populated and presented to the OS at runtime, for errors that are handled in firmware-first mode.

The CPER specification describes non-standard records, whose format is described in vendor-specific documents.

4.2.1 CPER section for error nodes

The CPER section for error nodes provides the means for presenting error node information to the OS in the form of a CPER record [1]. The section encapsulates the contents of an error node's error records.

The error node section is described by the following Section Type:

```
Section Type: {0xBF32D4D5, 0xB427, 0x4025, {0x84, 0x95, 0x8A, 0x9E, 0x5D, 0x40,
    ↪ 0x30, 0xE4}}
```

The contents of the section are described in Table 20.

Table 20: Arm RAS System Architecture node CPER

Field	Byte Length	Byte offset	Description
Revision	4	0	This field must be set to 0 for this version of the specification.
Component type	1	4	This field must be set to the value of the Type field in the AEST node that describes this error node. The Type field is defined in Table 4.
Reserved	3	5	This field must be set to zero.
Number of error record descriptors	4	8	Number of error records captured in the section.
Error record descriptor array	–	12	Array of error record descriptors, where each descriptor describes an error record based on [3]. The format of these entries is specified in Table 21.

Field	Byte Length	Byte offset	Description
Error node type-specific data	Varies	12 + sizeof(Error record descriptor array)	This field describes the component that the error node is associated with. The format of this field is based on the value of the Component type field and follows the structure outlined in Section 2.1.

Table 21: Arm RAS System Architecture node CPER error record descriptor

Field	Byte Length	Byte offset	Description
Error record number	4	0	RAS error record number within the error node.
RAS extension revision	1	4	Describes the revision of the Arm Architecture Reference Manual RAS Supplement [3] specification that this error node is based on. This field takes the following format: • bits[7:4] REVISION field of the ERRDEVARCH register defined in [3]. • bits[3:0] ARCHVER field of the ERRDEVARCH register defined in [3]. Note that registers ERR<n>MISC2 and ERR<n>MISC3 are only valid if this value is non zero.
Reserved	3	5	Reserved must be zero
ERR<n>FR	8	8	Contents of the Error Record Feature Register.
ERR<n>CTLR	8	16	Contents of the Error Record Control Register
ERR<n>STATUS	8	24	Contents of the Error Record Primary Status Register.
ERR<n>ADDR	8	32	Contents of the Error Record Address Register.
ERR<n>MISC0	8	40	Contents of the Error Record Miscellaneous Register 0.
ERR<n>MISC1	8	48	Contents of the Error Record Miscellaneous Register 1.
ERR<n>MISC2	8	56	Content of the Error Record Miscellaneous Register 2.
ERR<n>MISC3	8	64	Content of the Error Record Miscellaneous Register 3.