

# ACPI for the Armv8 RAS Extensions 1.0

## Platform Design Document

Non-confidential

The logo consists of the lowercase letters 'arm' in a bold, sans-serif font.

## Contents

Release information	3
<b>Arm Non-Confidential Document Licence (“Licence”)</b>	<b>4</b>
<b>1 About this document</b>	<b>6</b>
1.1 Terms and abbreviations	6
1.2 References	6
1.3 Feedback	6
<b>2 Introduction</b>	<b>8</b>
<b>3 The Arm Error Source Table</b>	<b>9</b>
3.1 Component types	10
3.1.1 Processor structures	10
3.1.1.1 Processor cache resource substructure	12
3.1.1.2 Processor TLB resource substructure	12
3.1.1.3 Processor generic resource substructure	12
3.1.2 Memory Controller structures	13
3.1.3 SMMU structures	13
3.1.4 Vendor-defined structures	13
3.1.5 GIC structures	14
3.2 Interface	14
3.3 Interrupts	15

Copyright © 2019, 2020 Arm Limited. All rights reserved.

## Release information

Date	Version	Changes
2020/Jul/28	1.0	• External Release

## Arm Non-Confidential Document Licence (“Licence”)

This Licence is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“**Licensee**”) is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

- (i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;
- (ii) manufacture and have manufactured products which have been created under the licence granted in (i) above; and
- (iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

**Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.**

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PERMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights

granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This Licence may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this Licence and any translation, the terms of the English version of this Licence shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. No licence, express, implied or otherwise, is granted to Licensee under this Licence, to use the Arm trade marks in connection with the Document or any products based thereon. Visit Arm's website at <https://www.arm.com/company/policies/trademarks> for more information about Arm's trademarks.

The validity, construction and performance of this Licence shall be governed by English Law.

Copyright © [2020] Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.  
110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-21585 version 4.0

# 1 About this document

## 1.1 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
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
UID	Unique Identifier

## 1.2 References

This section lists publications by Arm and by third parties.

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

[1] *Advanced Configuration and Power Interface Specification 6.3*. UEFI Forum.

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

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

## 1.3 Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an email to [errata@arm.com](mailto:errata@arm.com). Give:

- The title (ACPI for the Armv8 RAS Extensions).
- The document ID and version (DEN0085 1.0).
- The page numbers to which your comments apply.

- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

## 2 Introduction

This specification provides details on ACPI [1] extensions that enable kernel-first handling of errors in a system that support the Arm RAS extensions [2]. The specification covers Armv8.2+ RAS extensions [3] for PEs. The specification also covers the RAS system architecture for non-PE system components.



### 3 The Arm Error Source Table

Error nodes based on Armv8.2 and above are represented in the Arm Error Source Table (AEST), which is described in Table 3.

**Table 3: AEST format**

Field	Byte length	Byte offset	Description
<b>Header</b>			
Signature	4	0	'AEST', Arm error source table
Length	4	4	Length of table in bytes
Revision	1	8	For this revision this must be 0.
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.
<b>Body</b>			
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
<b>Header</b>			
Type	1	0	Node type: <ul style="list-style-type: none"> <li>• 0x00 - Processor error node</li> <li>• 0x01 - Memory error node</li> <li>• 0x02 - SMMU error node</li> <li>• 0x03 - Vendor-defined error node</li> <li>• 0x04 - GIC error node</li> </ul> All other values are reserved.
Length	2	1	Length of structure in bytes.
Reserved	1	3	Must be zero.
Offset to Node-specific data	4	4	Offset from the start of the node to node-specific data.
Offset to Node Interface	4	8	Offset from the start of the node to the node interface structure.
Offset to Node Interrupt Array	4	12	Offset from the start of the node to node interrupt array.
Node Interrupt Array size	4	16	Number of entries in the interrupt array.
<b>Node generic data</b>			

Field	Byte Length	Byte Offset	Description
Timestamp Rate	8	20	If the timestamp extension is implemented, and does not use the timebase of the system Generic Timer, as indicated by ERRFR.TS == 0b10, this field indicates the timestamp frequency of the counter in Hz. Otherwise this field must be zero and the OS must ignore its contents.
Reserved1	8	28	Reserved, must be zero.
Error injection countdown rate	8	36	If Common Fault Injection Model Extension is supported as indicated by ERRFR.INJ != 0b00, this field provides the rate in Hz at which the Error Generation Counter decrements. Otherwise this field must be zero and the OS must ignore its contents.
<b>Node-specific data</b>	–	Offset for node-specific data	
<b>Node interface</b>	–	Offset for node interface	
<b>Node Interrupt array</b>	–	Offset for node interrupt array	

AEST nodes provide a description of error nodes that are based on Arm v8.2+ RAS Architecture [2]. 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.

## 3.1 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 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:

- [Processor structures](#)
- [Memory controller structures](#)
- [SMMU structures](#)
- [Vendor-defined structures](#)
- [GIC structures](#)

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

### 3.1.1 Processor structures

Processor structures describe error nodes that are related to the processor and its internal components. Processor structures are 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.

**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 node, as specified by the Flags field.
Resource Type	1	4	Specifies which resource within the processor this node pertains to. <ul style="list-style-type: none"> <li>• 0x0 - Cache</li> <li>• 0x1 - TLB</li> <li>• 0x2 - Generic</li> </ul>
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 3.2). 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 [2].
<b>Resource-specific data</b>			
Resource substructure	–	16	Processor resource whose error node is being described by this node.

**Table 6: Processor structure flags**

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

Field	Bit Length	Bit offset	Description
Shared	1	1	<p>This flag is an indication that this node represents an error node that is on a resource that is shared by multiple processors in the system.</p> <p>The identity of the processors that are sharing this resource is specified in the processor affinity level indicator field.</p> <p>0b - This AEST node represents a resource that is private to the specified processor.</p> <p>1b - This AEST node represents a resource that is shared by multiple processors.</p>
Reserved	7	2	Reserved, must be zero.

### 3.1.1.1 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 information for identification of the cache, that cross-refers the OSPM to corresponding Type 1 (cache) structures in the PPTT.

**Table 7: Processor Cache resource substructure**

Field	Byte Length	Byte offset	Description
Cache level	4	0	Level of cache from perspective of current processor.
Cache type	4	4	<p>Cache type:</p> <ul style="list-style-type: none"> <li>• 0x0 Data</li> <li>• 0x1 Instruction</li> <li>• 0x2 Unified</li> </ul> <p>Other values are reserved.</p>

### 3.1.1.2 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 level	4	0	TLB level from perspective of current processor.

### 3.1.1.3 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.

### 3.1.2 Memory Controller structures

Memory controller structures are described in Table 9.

**Table 9: Memory Controller structure format**

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.

### 3.1.3 SMMU structures

SMMU [4] structures are described in Table 10. 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 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 10: 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.
<b>SMMU-specific data</b>			
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.

### 3.1.4 Vendor-defined structures

Vendor-defined structures are described in Table 11. An OSPM might log these structures as raw data or offer them to vendor-specific drivers where appropriate.

**Table 11: Vendor-defined structure format**

Field	Byte Length	Byte Offset	Description
Hardware ID	4	0	ACPI HID of the component.
Unique ID	4	4	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	8	Vendor-specific data, for example to identify this error source.

### 3.1.5 GIC structures

GIC [6] error structures are described in Table 12. A GIC implementation might have one or more internal error nodes on one or more internal interfaces.

**Table 12: GIC structure format**

Field	Byte Length	Byte Offset	Description
Interface	4	0	Index of the GIC interface structure in the ACPI MADT table that describes the GIC instance and GIC that are associated with this error node.

## 3.2 Interface

Nodes can have a System register (SR) or a memory mapped (MMIO) interface. The Interface structure describes the interface that the 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. Table 13 describes the format of the interface entries.

**Table 13: AEST node interface structure**

Field	Byte Length	Byte Offset	Description
Interface type	1	0	Interface type: <ul style="list-style-type: none"> <li>• 0x0 – SR</li> <li>• 0x1 – MMIO</li> </ul> All other values are reserved.
Reserved	3	1	Must be zero.
Flags	4	4	Flags associated with this node interface. See Table 14.
Base Address	8	8	Base address of error group that contains the error node. This address is only valid if interface type is == 0x1.
Start error record index	4	16	Zero-based index of the first standard error record that belongs to this 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.

Field	Byte Length	Byte Offset	Description
Number of error records	4	20	Number of error records in this node. Includes both implemented and unimplemented records.
Error record implemented	8	24	This bitmap indicates which of the error records within this node are implemented in the current system, and must be polled for error status. Bit[n] of this field pertains to error record corresponding to index n in the parent error group. Bit[n] = 0b: Error record at index n is implemented. Bit[n] = 1b: Error record at index n is not implemented.
Error record -based status reporting supported	8	32	This bitmap indicates which of the error records within this node support error status reporting through the ERRGSR register. Bit[n] of this field pertains to error record corresponding to index n in the parent error group. Bit[n] = 0: Error record at index n supports error status reporting through ERRGSR.S. Bit[n] = 1: 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.

**Table 14: AEST node interface flags**

Field	Bit Length	Bit Offset	Description
Shared interface	1	0	0b - Node interface is private to the node. 1b - Node interface is shared. For processor cache nodes, 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.
Reserved	30	2	Reserved, must be zero.

### 3.3 Interrupts

RAS architecture [2] nodes can generate three kinds of interrupts:

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

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



**Table 15: AEST node interrupt structure**

Field	Byte Length	Byte Offset	Description
Interrupt type	1	0	Interrupt type: <ul style="list-style-type: none"> <li>• 0x0 – Fault Handling Interrupt</li> <li>• 0x1 – Error Recovery Interrupt</li> </ul> All other values are reserved.
Reserved	2	1	Must be zero.
interrupt Flags	1	3	Bits [31:1]: Must be zero. Bit 0: <ul style="list-style-type: none"> <li>• 0b – Interrupt is edge-triggered</li> <li>• 1b – Interrupt is level-triggered</li> </ul>
Interrupt GSIV	4	4	GSIV of interrupt, if interrupt is an SPI or a PPI. Must be zero if the interrupt is no wire-based. If GSIV is 0, then MSI must be used instead.
ID	1	8	If MSI is supported, then this field must be set to the Identifier field of the IORT ITS Group node. Must be zero if interrupt is wire-based.
Reserved1	3	9	Must be zero.