**# Title:**

RASF Gen2

**# Status:**

Draft

**# Document:**

ACPI Specification Version 6.next

**# License:**

SPDX-License-Identifier: CC-BY-4.0

**# Submitter:**

* Harb Abdulhamid (Ampere)
* Thanu Rangarajan (Arm)
* Samer El-Haj Mahmoud (Arm)
* TianoCore Community (<https://www.tianocore.org>)

**# Summary of the change**

This ECR is related to a proposal to upgrade the current RASF table. The upgraded RASF table offers a richer set of interfaces that enables the OS to perform improved PFA and scrub control of system memory. Additionally, the upgraded RASF table:

* includes provisions for extending to other types of RAS features that might apply to other components in the system such as caches.
* fixes issues that were present in RASF Gen 1.

The enhanced scrub controls are required to address RAS requirements of modern scalable platforms that have complex memory systems with a multitude of memory controllers that are in turn associated with NUMA domains. It is also common for RAS errors related to memory to be associated with NUMA domains, where the NUMA domain functions as a FRU identifier. As a result, PFA tools and error recovery algorithms that such tools might use could benefit from being able to control memory scrubbing at a NUMA domain granularity. The updated RASF table enables this capability.

Specifically, the ECR includes the following changes:

1. Updated signature of the shared memory of the subchannel in accordance with the definitions in Chapter 14 (Platform Communication Channel) of the ACPI specification, which dictate that the shared memory must have a signature of type “PCCx”, where x is the subchannel index in the PCCT. RASF Gen1 violates this rule.

2. The ability to scale RAS functions to multiple instances of the same underlying component, which is offered in the form of more than one PCC channel per RAS function, where a channel can be dedicated to a given component instance.

3. Independent memory scrubbing controls for each NUMA domain, based on #2.

4. Generalization to system components other than memory.

5. Provision for background (or patrol) scrubbing, distinct from on-demand scrubbing for a specific region.

The changes are done in a manner to ensure backward compatibility with RASF Gen1.

**# Benefits of the change**

The use of RASF Gen2 enables OSPM to perform improved PFA in a given system that supports it.

**# Impact of the change**

Platforms firmware will have to support the new format of RASF. OS’s require a driver to support RASF. These are both fundamentally new code.

**# Detailed description of the change [normative updates]**

Existing text

New text

Deleted Text

Edit Instructions

**1.7.1 Platform Implementations of ACPI-defined Interfaces**

…

• Maximum System Characteristics Table (MSCT)

• ACPI RAS Feature Table (RASF)

• ACPI RAS2 Feature Table (RAS2)

• Memory Power State Table (MPST)

• Platform Memory Topology Table

…

*Instructions: Add new section in ACPI 6.5 after 5.2.20 and before 5.2.21. Using the placeholder X to represent this new section.*

**5.2.6 System Description Table Header**

…

**Table 5.5: DESCRIPTION\_HEADER Signatures for tables defined by ACPI**

…

|  |  |  |
| --- | --- | --- |
| …. |  |  |
| “RASF” | ACPI RAS Feature Table | Section 5.2.20 |
| “RAS2” | ACPI RAS2 Feature Table | Section 5.2.X |
| …. |  |  |

**5.2.X ACPI RAS2 Feature Table (RAS2)**

The RAS2 table provides interfaces for platform RAS features. RAS2 offers the same services as RASF, but is more scalable than the latter. In particular, RAS2 supports independent RAS controls and capabilities for a given RAS feature for multiple instances of the same component in a given system.

The platform can support either RAS2 or RASF but not both.

**Table 5.a RAS2 Table Format**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| Header | | | |
| Signature | 4 | 0 | Signature is set to ‘RAS2’ for RAS Feature 2 Table. |
| Length | 4 | 4 | Length in bytes for entire RAS2. The length implies the number of PCC descriptors fields at the end of the table |
| Revision | 1 | 8 | 1 |
| Checksum | 1 | 9 | Entire table must sum to zero |
| OEMID | 6 | 10 | OEM ID |
| OEM Table ID | 8 | 16 | The table ID is the manufacturer model ID |
| OEM Revision | 4 | 24 | OEM revision of table for supplied OEM Table ID |
| Creator ID | 4 | 28 | Vendor ID of utility that created the table |
| Creater Revision | 4 | 32 | Revision of utility that created the table |
| **RAS2 Specific Enteries** |  |  |  |
| Reserved | 2 | 36 | Reserved, should be zero. |
| Number of PCC descriptors | 2 | 38 | Number of PCC descriptors. |
| RAS2 Platform Communication Channel (PCC) Descriptor List | N\*8 | 40 | List of PCC descriptors. |

**5.2.X.1 Common Definitions**

**5.2.X.1.1 RAS2 Platform Communication Channel Descriptor**

RAS2 supports multiple PCC channels, where a channel is dedicated to a given component instance. The RAS2 PCC descriptor specifies the PCC sub-space associated with a specific RAS feature. The RAS feature type specifies the RAS feature.

**Table 5.b RAS2 Platform Communication Channel Descriptor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| PCC Identifier | 1 | 0 | Identifier of the RAS2 Platform Communication Channel. OSPM should use this value as an index into the subspace array within the PCCT table. |
| Reserved | 2 | 1 | Reserved, must be zero. |
| Feature Type | 1 | 3 | RAS feature type. RAS feature types are defined in **5.c**. |
| Instance | 4 | 4 | Identifier for the system component instance that this RAS feature is associated with. |

**Table 5.c RAS Feature Types**

|  |  |
| --- | --- |
| **RAS Feature Type** | **Description** |
| 0x00 | RAS features related to memory. |
| 0x01-0x7F | Reserved for future standard RAS feature types defined by this specification. |
| 0x80-0xFF | Vendor-defined RAS feature types. |

**5.2.X.1.2 Using PCC registers**

OSPM will write PCC registers by filling in the register value in PCC sub channel space and issuing a PCC Execute command. See Table 5.d. To minimize the cost of PCC transactions, OSPM should read or write all registers in the same PCC subspace via a single read or write command.

**Table 5-d PCC Command Codes used by RAS2 Platform Communication Channel**

|  |  |
| --- | --- |
| **Command** | **Description** |
| 0x00 | Reserved |
| 0x01 | Execute RAS2 Command. |
| 0x02-0xFF | All other values are reserved. |

**5.2.X.1.3 RAS2 Platform Communication Channel**

RAS2 platform communication channel format is defined in Table5.e as below.

**Table 5.e RAS2 Platform Communication Channel Shared Memory Region**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| Signature | 4 | 0 | The PCC signature. The signature of a subspace is computed by a bitwise-or of the value 0x50434300 with the subspace ID. For example, subspace 3 has the signature 0x50434303. |
| Command | 2 | 4 | PCC command field; see PCC Command Codes used by RAS2. See Table 5.d and Section 14. |
| Status | 2 | 6 | PCC status field. See Section 14. |
| Communication Space: |  |  |  |
| Version | 2 | 8 | Byte 0 - Minor Version | Byte 1 - Major Version. For this revision, this field must be set to 0x0000. |
| RAS Features | 16 | 10 | Bitmap describing the platform RAS features as  shown in Table 5.f. The definition of the bits is system component specific. For example, Table 5.h shows the bitmap definitions for Memory RAS features. The Platform populates this field to indicate which RAS features for the given feature type are supported for this system component instance. The OSPM uses this field for RAS feature discovery. |
| Set RAS Capabilities | 16 | 26 | Bit Map of the RAS features for which the OSPM is  invoking the command. The Bit Map is described in  Section 5.2.X.1.4. OSPM sets the bit corresponding to a  RAS capability to invoke a command on that capability.  The bitmap implementation allows OSPM to invoke a  command on each RAS feature supported by the platform  at the same time. |
| Number of RAS2 Parameter  blocks | 2 | 42 | The Number of parameter blocks will depend on how  many RAS Capabilities the Platform Supports. Typically,  there will be one Parameter Block per RAS Feature, using which that feature can be managed by  OSPM. |
| • Set RAS Capabilities Status | 4 | 44 | Status:  0000b = Success  0001b = Not Valid  0010b = Not Supported  0011b = Busy  0100b = FailedF  0101b = Aborted  0110b = Invalid Data |
| • Parameter Blocks | Varies  (N  Bytes) | 48 | Start of the parameter blocks, the structure of which  is shown in the Parameter Block Structure for PATROL\_  SCRUB. These parameter blocks are used as  communication mailbox between the OSPM and the  platform, and there is 1 parameter block for each RAS  feature. NOTE: There can be only on parameter block  per type. |

**5.2.X.1.4 RAS2 Platform RAS Feature Bitmap (generic)**

The following table shows a generic definition of the Platform RAS features supported for a given system component type. The exact definitions are specific for each system component type. For example, table 5.h show the bitmap definitions for Memory RAS features.

**Table 5.f Platform RAS Feature Bitmap**

|  |  |  |
| --- | --- | --- |
| **Bit** | **RAS Feature** | **Description** |
| 0 | Feature 1 | RAS Feature 1 |
| 1 | Feature 2 | RAS Feature 2 |
| … | … | … |
| 127 | Feature 128 | RAS Feature 128 |

**5.2.X.2 Memory RAS Features – Feature Type 0**

Memory RAS features apply to RAS capabilities, controls and operations that are specific to memory. These features might be provided through one or more PCC sub-spaces. RAS2 sub-spaces for memory-specific RAS features have a Feature Type of 0x00 (Memory).

**Table 5.g RAS2 Platform Communication Channel Descriptor for Memory RAS Features**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| PCC Identifier | 1 | 0 | Identifier of the RAS2 Platform Communication Channel. OSPM should use this value as an index into the subspace array within the PCCT table. |
| Reserved | 2 | 1 | 0 |
| Feature Type | 1 | 3 | 0x00: Memory. See **5.c**. |
| Instance | 4 | 4 | Proximity domain that this RAS feature is associated with. This field must match the ACPI SRAT table definitions. See **5.2.16**. |

**Table 5.h Platform RAS Feature Bitmap for Memory RAS**

|  |  |  |
| --- | --- | --- |
| **Bit** | **RAS Feature** | **Description** |
| 0 | Hardware-based memory scrub feature | Indicates that the platform supports hardware-based memory scrubbing. |
| 1-127 | *Reserved* | *Reserved for future use* |

**5.2.X.2.1 Hardware-based Memory Scrubbing**

The platform can use this feature to expose controls and capabilities associated with hardware-based memory scrub engines. Modern scalable platforms have complex memory systems with a multitude of memory controllers that are in turn associated with NUMA domains. It is also common for RAS errors related to memory to be associated with NUMA domains, where the NUMA domain functions as a FRU identifier. This feature thus provides memory scrubbing at a NUMA domain granularity.

The following are supported:

* Independent memory scrubbing controls for each NUMA domain, identified using its proximity domain.
* Provision for background (patrol) scrubbing of the entire memory system, as well as on-demand scrubbing for a specific region of memory.

**Table 5.j Parameter Block Structure for PATROL\_SCRUB**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| Type | 2 | 0 | 0x0000 – Hardware-based memory scrub RAS feature. |
| Version | 2 | 2 | Byte 0 – Minor Version  Byte 1 – Major Version  For this format of the parameter block, this field should be set to 0x0001. |
| Length | 2 | 4 | Length, in bytes of the entire parameter block structure |
| Patrol Scrub Command  (INPUT) | 2 | 6 | 0x01 - GET\_PATROL\_PARAMETERS  0x02 - START\_PATROL\_SCRUBBER  0x03 – STOP\_PATROL\_SCRUBBER |
| Requested Address Range  (INPUT) | 16 | 8 | OSPM Specifies the BASE (Bytes 7-0) and SIZE (Bytes 15-8) of the address range to be patrol scrubbed. If OSPM requests default scrubbing through Bit 0 of the Configure patrol scrubbing field, then this field must be ignored by the platform.  OSPM sets this parameter for the following commands:  GET\_PATROL\_PARAMETERS  START\_PATROL\_SCRUBBER |
| Actual Address Range  (OUTPUT) | 16 | 24 | The platform returns this value in response to GET\_PATROL\_PARAMETERS. The platform calculates the nearest patrol scrub boundary address from where it can start. This range should be a superset of the Requested Address Range.  This field must be ignored by the OSPM if it is being returned in response to a request to enable default scrubbing through Bit 0 of the Configure patrol scrubbing field.  BASE (Bytes 7-0) and SIZE (Bytes 15-8) of the address |
| Flags (OUTPUT) | 4 | 40 | The platform returns this value in response to GET\_PATROL\_PARAMETERS  Bit [0]: Will be set if memory scrubber is already running for address range specified in “Actual Address Range”.  Bits [31:1]: Reserved, must be zero. |
| Scrub Parameters (OUTPUT) | 4 | 44 | The platform returns this value in response to GET\_PATROL\_PARAMETERS  Bits [7:0]: Current scrub rate that is in effect on the memory region specified in “Actual Address Range”. If OSPM requested background scrubbing, then this field will reflect the current background patrol scrubbing rate.  Bits [15:8]: Minimum scrub rate supported.  Bits [23:16]: Maximum scrub rate supported  Bits [31:24]: Reserved, must be zero. |
| Configure Scrub Parameters (INPUT) | 4 | 48 | The OSPM sets this field as follows, for the START\_PATROL\_SCRUBBER command:  Bit[0]: Request background patrol scrubbing.  Bits [7:1]: Reserved, must be zero.  Bits [15:8]: Requested scrub rate, must be in the range (minimum scrub rate, maximum scrub rate).  Bits [31:16]: Reserved, must be zero. |