



# Realm Management Extension (RME) and Memory Encryption Contexts (MEC) example software flows

Version 1.0

**Non-Confidential**

Copyright © 2024 Arm Limited (or its affiliates).  
All rights reserved.

**Issue 01**

109839\_0100\_01\_en



# Realm Management Extension (RME) and Memory Encryption Contexts (MEC) example software flows

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.

## Release information

### Document history

Issue	Date	Confidentiality	Change
0100-01	11 June 2024	Non-Confidential	First release

## Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm’s view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS 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. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

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

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

## Product Status

The information in this document is Final, that is for a developed product.

## Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email [terms@arm.com](mailto:terms@arm.com).

# Contents

<b>1. Overview.....</b>	<b>6</b>
<b>2. RME Examples.....</b>	<b>7</b>
<b>3. Granule Transition Flow.....</b>	<b>8</b>
<b>4. Procedures for changing the size of a GPT contiguous region.....</b>	<b>10</b>
<b>5. MEC Examples.....</b>	<b>12</b>
<b>6. Mismatched MECID avoidance.....</b>	<b>13</b>
<b>7. Primary and Alternate MECID use models.....</b>	<b>14</b>
7.1 Realm EL1&0 translation regimes.....	14
7.1.1 Primary MECID.....	14
7.1.2 Alternate MECID.....	14
7.2 Realm EL2&0 translation regimes.....	15
7.2.1 Primary 0 MECID.....	15
7.2.2 Alternate 0 MECID.....	15
7.2.3 Primary 1 MECID.....	15
7.2.4 Alternate 1 MECID.....	16
7.3 EL3 translation regime.....	16
<b>8. Related Information.....</b>	<b>17</b>

# 1. Overview

In this guide, we look at several different programming examples of software flows. These flows are important in the understanding of the Realm Management Extension (RME) Arm Architecture.

This guide assumes a familiarity with the Arm A-profile architecture and the Realm Management Extension.

The examples included in this guide are:

- Granule Transition Flow
- Procedures for changing the size of a GPT contiguous region
- Mismatched MECID avoidance
- Granule MECID Association Flows
- Primary and Alternate MECID use models

## 2. RME Examples

The following provides example software sequences for using the new features introduced within the The Realm Management Extension (RME), for Armv9-A.

## 3. Granule Transition Flow

The properties of the Granule Transition Flow (GTF) are:

1. The GTF changes the Physical Address Space (PAS) association of a physical granule from a previous physical address space to a new physical address space.
2. The GTF completes once the association of the physical granule with the new physical address space is observable.
3. When the GTF completes the following outcomes are guaranteed:
  - Writes to the previous physical address space will not become observable.
  - If the previous physical address space is Realm or Secure then no accesses, including Speculative read accesses, to the previous physical address space can observe unscrubbed values from before the GTF.
  - If the previous physical address space is Realm or Secure then no instructions, including execution under Speculation, can observe unscrubbed values from before the GTF.
  - If the previous physical address space is Realm or Secure then no accesses to the granule in the new physical address space observe unscrubbed values.
4. GTF outcomes are guaranteed by EL3 without relying on cooperative behavior of SW that has access to the previous physical address space (for example, software running at EL2), other than performing scrubbing operations where appropriate.

### Delegate

This sequence transitions the physical granule at address `addr` from Non-secure to Secure or Realm physical address space.

On implementations with FEAT\_MTE2, Root firmware must issue `DC_CIGDPAPA` instead of `DC_CIPAPA`, in order to additionally clean and invalidate Allocation Tags associated with the affected locations.

```
Delegate(phys_addr* addr, PAS target_pas) {  
    // In order to maintain mutual distrust between Realm and Secure  
    // states, remove any data speculatively fetched into the target  
    // physical address space.  
    for (i = 0; i < granule_size; i += cache_line_size)  
        DC_CIPAPA((addr+i), target_p);  
  
    DSB(OSH);  
  
    write_gpt(addr, target_pas)  
    DSB(OSHST);  
  
    TLBI_RPALOS(addr, granule_size);  
    DSB(OSH)  
  
    for (i = 0; i < granule_size; i += cache_line_size)  
        DC_CIPAPA((addr+i), PAS_Ns);  
  
    DSB(OSH);  
}
```



```
}
```

## Undelegate

This sequence transitions the physical granule at address `addr` from Secure or Realm physical address space to Non-secure.

The sequence assumes that the EL2 software for Secure or Realm state has already scrubbed the appropriate locations.

On implementations with FEAT\_MTE2, Root firmware must issue `DC_CIGDPAPA` instead of `DC_CIPAPA`, in order to additionally clean and invalidate Allocation Tags associated with the affected locations.

```
Undelegate(phys_addr* addr, PAS current_pas)

    // In order to maintain mutual distrust between Realm and Secure
    // states, remove access now, in order to guarantee that writes
    // to the currently-accessible physical address space will not
    // later become observable.
    write_gpt(addr, No_access);
    DSB(ØSHST);
    TLBI RPALOS(addr, granule_size);
    DSB(ØSH);

    // Ensure that the scrubbed data has made it past the PoPA
    for (i = 0; i < granule_size; i += cache_line_size)
        DC_CIPAPA((addr+i), current_pas);

    DSB(ØSH);

    // Ensure that GPC completes, non-coherent caches are properly flushed
    TLBI RPALOS(addr, granule_size);
    DSB(ØSH);

    // Remove any data loaded speculatively in NS space from before the scrubbing
    for (i = 0; i < granule_size; i += cache_line_size)
        DC_CIPAPA((addr+i), PAS_NS);

    DSB(ØSH);

    write_gpt(addr, PAS_NS);
    DSB(ØSHST);

    // Ensure that all agents observe the new NS configuration
    TLBI RPALOS(addr, granule_size);
    DSB(ØSH);
```

## 4. Procedures for changing the size of a GPT contiguous region

Example procedure to increase contiguity from 4KB to 2MB, assuming PGS is 4KB. This example procedure does not consider mutual exclusion.

```
// Parameters:
// base = base address of desired contig region
// expected_gpi = value of all GPIs in the region

assert IS_ALIGNED(base, 2MB);
assert IS_VALID_GPI (expected_gpi)

// Required GPTE is 16 GPI values, all the same
uint64_t required_gpте;
required_gpте = expected_gpi;
required_gpте |= required_gpте << 4;
required_gpте |= required_gpте << 8;
required_gpте |= required_gpте << 16;
required_gpте |= required_gpте << 32;

for(gpте_addr=base, gpте_addr < base+2MB, gpте_addr += 64KB) {
    actual_gpте = gpt_entry(gpте_addr);
    // All entries must be consistent before the change
    if (actual_gpте !=required_gpте)
        return false;
}

uint64_t new_gpте = 0x1; // Contiguous descriptor
new_gpте |= expected_gpi<<4; // GPI field
new_gpте |= 01<<8; // Contig field

for(gpте_addr=base, gpте_addr < base+2MB, gpте_addr += 64KB) {
    set_gpt_entry(gpте_addr, new_gpте);
}

// No TLB maintenance required
return true;
```

Example procedure to decrease contig from 2MB to 4KB, assuming PGS is 4KB.

This example procedure does not consider mutual exclusion.

```
// Parameters
// base = = base address of desired contig region
// expected_gpi = value of all GPIs in the region

assert IS_ALIGNED(base, 2MB);
assert IS_VALID_GPI(expected_gpi);

// Required GPTE value
uint64_t required_gpте = 0x1; // Contiguous descriptor
required_gpте |= expected_gpi<<4; // GPI field
required_gpте |= 01<<8; // Contig field

for(gpте_addr=base, gpте_addr < base+2MB, gpте_addr += 64KB) {
    // All entries must be consistent before the change
    if (actual_gpте !=required_gpте)
        return false;
}
```

```
// New GPTE is 16 GPI values, all the same
uint64_t new_gppte;
new_gppte = expected_gpi;
new_gppte |= new_gppte << 4;
new_gppte |= new_gppte << 8;
new_gppte |= new_gppte << 16;
new_gppte |= new_gppte << 32;

for(gppte_addr=base, gppte_addr < base+2MB, gppte_addr += 64KB) {
    set_gpt_entry(gppte_addr, new_gppte);
}

// It is assumed that the entries are being cracked so that they can
// be changed, for whatever reason. In which case it required to
// perform TLB maintenance now.
DSB();
TLBI_RPALOS(base, 2MB);
DSB();
return true;
```

## 5. MEC Examples

The following provides example software sequences for using the new features introduced by Memory Encryption Contexts (MEC), for Armv9-A.

It refers to generic “software” if the example is appropriate to more than one exception level, and specifically to conceptual “Realm Management software” when describing procedures for Realm EL2.

## 6. Mismatched MECID avoidance

Software must ensure that mismatched Memory Encryption Context ID (MECID) memory accesses do not occur, as they will corrupt memory within the Realm PA space.

Software achieves this by ensuring:

- All active translations to a physical granule use the same associated MECID value.
  - If multiple translations exist to a physical granule with different MECID values, then mismatched MECIDs can occur due to architected or speculative accesses.
- Granules are not accessible through an active translation when updating the associated MECID value in a system register.
- All cache lines are cleaned and invalidated to the PoE, before any change of associated MECID or MECID state.

## 7. Primary and Alternate MECID use models

The following are examples of Primary and Alternate MECID use models.

### 7.1 Realm EL1&O translation regimes

This regime includes two translation tables. This translation regime is subject to stage 2 translation:

#### 7.1.1 Primary MECID

The VMECID\_P\_EL2 is the primary MECID, for the stage 2 Realm EL1&O translation regime. VMECID\_P\_EL2 will be used implicitly by the EL1&O Realm VM to protect the Realm VM state and provide consistent security guarantees, the Realm state includes:

- Private code and data pages
- EL1&O stage 1 translation tables
- EL1&O stage 2 translation tables

#### 7.1.2 Alternate MECID

The VMECID\_A\_EL2 is the alternate MECID, for the stage 2 Realm EL1&O translation regime. VMECID\_A\_EL2 is designed to support the secondary use cases of:

- Shared data between two or more Realm VMs
- Shared code between two or more Realm VMs

Note, a shared data model between Realm VMs has significant challenges with mutual authentication, that exceed the current scope of the CCA Security Model.

MEC provides only one alternate context per EL1 Realm. This greatly limits the creation of 1:1 encrypted shared memory channels, between multiple Realms.

## 7.2 Realm EL2&0 translation regimes

When `HCR_EL2.E2H == 1` the Realm EL2&0 translation regime has two sets of translation tables, referenced by `TTBR0_EL2` and `TTBR1_EL2`, each with their own primary and alternate MECIDs.

A common usage model is for supervisory software to use `TTBR1_EL2` translation tables for its own code and data, and dynamically use the `TTBR0_EL2` translation tables for the software it is currently supervising. The following explanations of EL2 MECIDs assumes this model.

### 7.2.1 Primary 0 MECID

The EL2&0 primary 0 MECID, when set to be equal `VMECID_P_EL2`, can enable the Realm management software to access the Realm VM state and stage 2 translation tables.

And optionally, Realm management software may protect additional Realm state with this MECID:

- Meta-data
- vCPU contexts

This MECID is configured in `MECID_P0_EL2` for the `TTBR0_EL2` translation table.

The primary Realm VM MECID value will be identical between `VMECID_P_EL2` and `MECID_P0_EL2`

### 7.2.2 Alternate 0 MECID

The alternative 0 MECID can be used by the Realm management software to implement use cases such as code or data sharing.

Realm management software having simultaneous access to primary 0 MECID and alternate 0 MECID of a Realm VM, can allow for efficient implementation of data sharing schemes that require data copying between Realm contexts.

This MECID is configured in `MECID_A0_EL2` for the `TTBR0_EL2` translation tables. For Realm management software to accesses memory shared with a Realm VM, the alternate Realm VM MECID value, `VMECID_A_EL2`, would be identical to that in `MECID_A0_EL2`.

### 7.2.3 Primary 1 MECID

The primary 1 MECID can be used by the Realm management software for its own code, private data and the `TTBR1_EL2` and `TTBR0_EL2` Realm EL2 translation tables.

This MECID is configured in `MECID_P1_EL2` for the `TTBR1_EL2` translation tables.

## 7.2.4 Alternate 1 MECID

The alternate 1 MECID can be used by the Realm management software for private data it wishes to isolate from the primary 1 MECID context, if the Realm management software security model requires such isolation. This MECID is configured in MECID\_A1\_EL2 for the TTBR1\_EL2 translation tables.

## 7.3 EL3 translation regime

All accesses to the Secure, Non-Secure and Root PA space from EL3, use the default MECID zero.

Memory accesses to the Realm PAS, when translation table descriptor {NSE,NS} = {1,1}, are associated with the MECID configured in MECID\_RL\_A\_EL3.

MECID\_RL\_A\_EL3 is most likely to contain the MECID associated with the Realm management software.

EL3 Monitor can use MECID\_RL\_A\_EL3 when accessing the Realm EL2 memory space to:

- Populate Realm PA space memory during Realm EL2 boot
- Write attestation reports for Realm management software

If non-delegated attestation signing is in use, the Monitor could directly write into the requesting Realm's memory, provided it is supplied with the granule PA and Realm MECID



## 8. Related Information

The following resources are related to material in this guide:

- [Arm Architecture Reference Manual for A-profile architecture](#)
- [Arm System Memory Management Unit Architecture \(SMMU\) Specification](#)
- [Learn the Architecture - SMMU Software Guide](#)
- [Arm Realm Management Extension \(RME\) System Architecture](#)
- [Learn the architecture - Realm Management Extension](#)
- [Learn the architecture - Introducing Arm Confidential Compute Architecture](#)