TF-RMM ID reg management

Design Review

Soby Mathew & Sona Rebecca

10/30/2025

Agenda:

- Overview - Introduction and motivation behind approach
- Previous approach and issues seen
- Design flow
- Future enhancements and benefits
- References

Overview – Motivation Behind the New ID reg management in RMM

Primary problem statement:

Feature discovery:

- RMM needs to configure features for its own use and functionality and must determine which features are enabled in Realm world and take the appropriate actions when Realms are scheduled.

Version mismatch:

- RMM and TF-A evolve at different paces and are often maintained by different teams.
- Testing environments often combine newer RMM builds with older TF-A versions, leading to traps, panics, or UNDEF abort due to mismatched expectations.

Secondary problem:

ID Register Emulation:

- RMM emulates ID registers for Realms using ad-hoc field masking.
- This approach is fragile as newly introduced ID register fields may be exposed to Realms if RMM isn’t updated to mask them.

Previous ID Register Access in RMM – Risks

How Feature Discovery Was Previously Handled

- RMM directly reads hardware ID registers at runtime or boot such as ID_AA64PFR0_EL1, ID_AA64ISAR1_EL1, etc. to discover CPU feature support.
- This approach assumed that EL3 (TF-A) had already enabled the relevant features and configured trap controls to allow access from EL2. As a result, RMM implicitly depended on a tight revision lockstep with TF-A to avoid access violations.

Scenario for primary problem: Disabling BRBE from R-EL2

- Previously, EL3 disabled BRBE for the Realm world by writing MDCR_EL3.SBRBE = 0x00
- During the world switch at EL3, the value of MDCR_EL3.SBRBE is not changed— remaining 0x01 instead of resetting to 0x00 — ensuring that BRBE stays enabled for the Realm world.
- RMM now disables BRBE by writing to BRBCR_EL2.
- RMM also emulates FEAT_BRBE as being absent to Realms at R-EL1.
- So, the latest TF-RMM now requires the latest TF-A for successful boot.

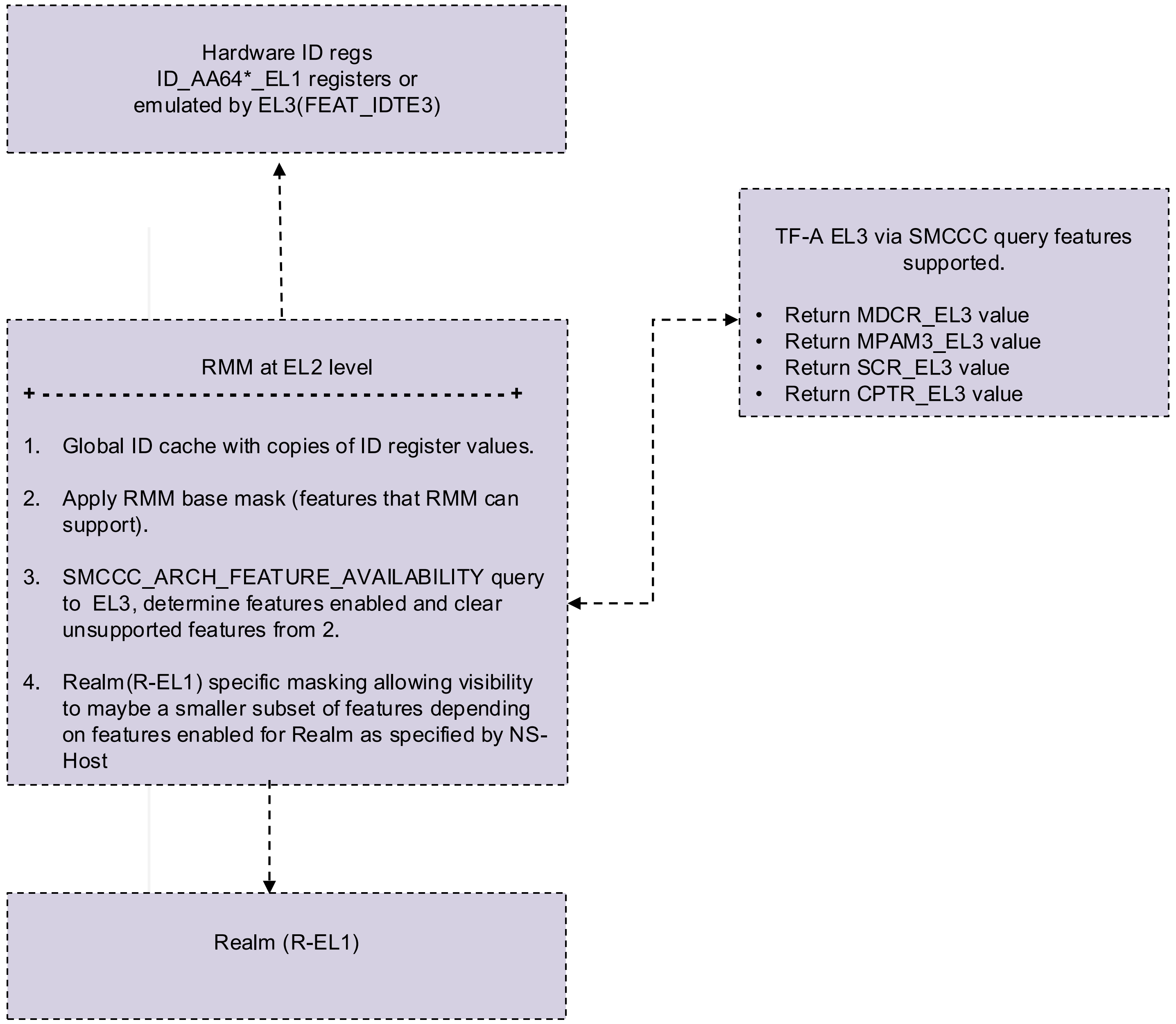
Example for the secondary problem: ID register emulation

- In the ID register emulation logic, the previous version of RMM had masks to hide features from Realms.
- These masks had to be updated based on if a feature needed to be hidden or exposed to Realms.
- For e.g. ID_AA64DFR1_EL1 was emulated such that only ICNTR and EBEP fields were cleared by RMM. All the other fields are exposed to Realms even if the feature will trap to RMM from R-EL1.

Cached ID reg framework

How the New Approach Works:

- Cache the ID register values once in RMM memory in a global cache.
- We need a framework in RMM to discover features enabled in EL3; keep it cached at RMM and use this info to further filter features exposed via ID reg to Realms at R-EL1
  - Create a single global structure in RMM memory that stores all ID register values once at cold-boot.
  - First stage of filtering by R-EL2 with a predefined base mask
    - This is an "allow" mask that filters only features R-EL2 will support or need to be exposed to R-EL1.
  - Second stage of filtering is via SMC to EL3 to query the features supported by EL3.
- This way R-EL2 now has created ID reg cache based on features it can use .
- Realms only see a safe subset of features explicitly enabled for its use. Future features exposed in new fields will not be visible to Realms.



Hardware ID regs
ID_AA64*_EL1 registers or
emulated by EL3(FEAT_IDTE3)
TF-A EL3 via SMCCC query features supported.
• Return MDCR_EL3 value
• Return MPAM3_EL3 value
• Return SCR_EL3 value
• Return CPTR_EL3 value
RMM at EL2 level
1. Global ID cache with copies of ID register values.
2. Apply RMM base mask (features that RMM can support).
3. SMCCC_ARCH_FEATURE_AVAILABILITY query to EL3, determine features enabled and clear unsupported features from 2.
4. Realm(R-EL1) specific masking allowing visibility to maybe a smaller subset of features depending on features enabled for Realm as specified by NS-Host
Realm (R-EL1)

Conclusion: Future enhancements and benefits

- Discover feature enablement status of EL3 for realm world.

- Prevent ID reg fields which may get used in future Arch revisions from being visible to realms.
  - Some RES0 fields — which are currently reserved and expected to read as zero — could change meaning in future revisions exposing features to Realms that RMM might not support.

- With a global cached ID reg scheme created at boot time, we avoid the need to call the SMCCC_ARCH_FEATURE call every instance an ID register is accessed.

- With FEAT_IDTE3, having a global cached copy of the ID registers is essential to avoid traps to EL3.

- In future this mechanism can be:
  - Used to detect feature mismatches in secondary warm boot and RMM can then downgrade global capability to a lower subset if no Realm has started yet, or panic.
  - When Realm migration is supported in future, RMM can compare feature set in this global structure and deny migration requests early if the feature set has a mismatch.

References

- Documentation for SMC Calling Convention (SMCCC)/SMCCC_ARCH_FEATURE_AVAILABILITY
  - https://developer.arm.com/documentation/den0028/latest

- Supporting TF-A patches:
  - https://review.trustedfirmware.org/q/topic:%22bk/smccc_feature%22

- RMM patches:
  - https://review.trustedfirmware.org/q/topic:%22sm/revlock%22

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm