YVR18-107 - Platform Error Injection and Error Handling on Arm64 based platforms for RAS

Sughosh Ganu <sughosh.ganu@arm.com>

Thomas Abraham <thomas.abraham@arm.com>

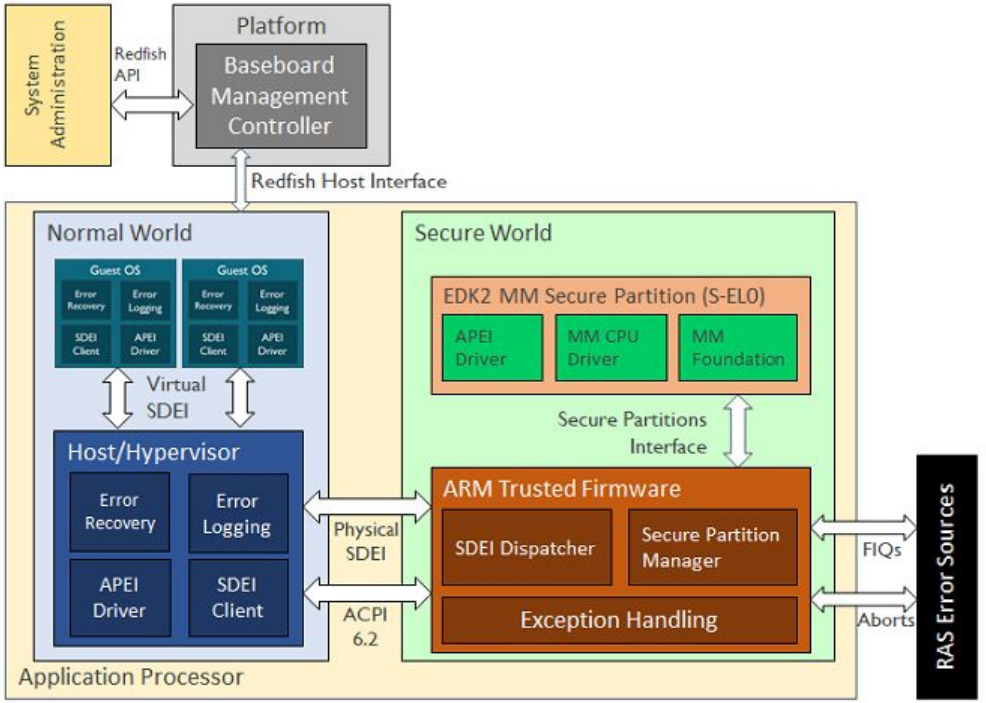
Development Status of RAS support for Arm64

- Multiple software components are under development to enable platform error injection and error handling support for Arm64 based server platforms
  - SPM, SP, APEI Dxe Driver, SDEI dispatcher, SDEI client
  - Integration of these components to demonstrate platform error injection and error handling flow is complete

- Significant progress has been made towards integrating these various software components on Arm's SGI-575 FVP platform to demonstrate platform error injection and error handling mechanism
  - Integration issues identified and resolved
  - Upstreaming of platform specific RAS support patches for SGI-575 platform is in progress
    - Can act as a reference solution for other Arm64 based platforms.

Arm64 specific System Components

- Secure Partition Manager (TF-A)
  - Executes with EL3 privileges and routes platform error event interrupts to appropriate components for further processing
- StandaloneMM (EDK2)
  - Implemented as a Secure Partition (SP) and executing at S-EL0, handles error injection and error handling requests from SPM
- SDEI Dispatcher (TF-A)
  - Dispatch system events (including platform error events) to SDEI client
- SDEI Client (Linux)
  - Registers with SDEI dispatcher to receive notifications about system events (platform error events) and forwards it to appropriate sub-system within linux for further processing
- APEI Dxe Driver (StandaloneMM)
  - Queries supported platform error sources from StandaloneMM and builds HEST



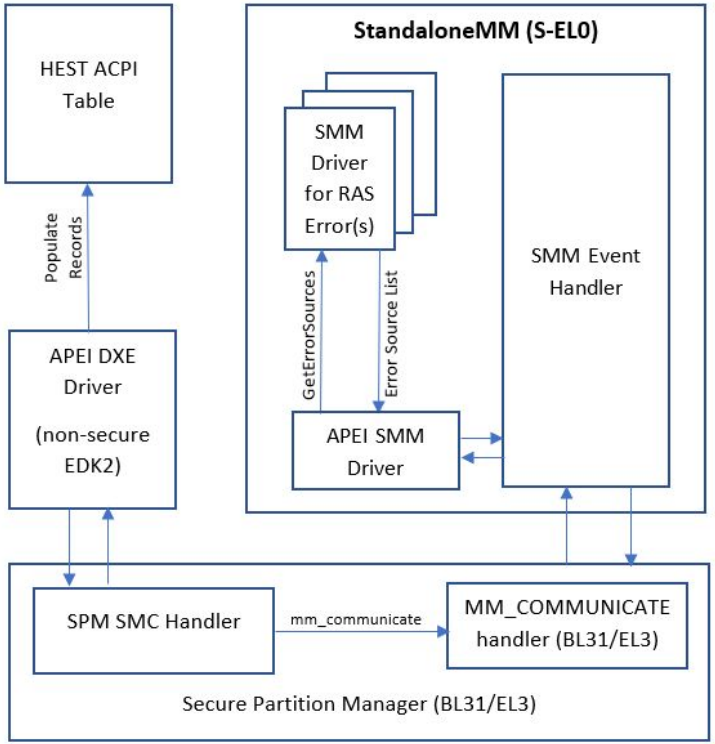
Putting them together for Error Injection and Handling

- Arm64 specific system wide components required to support RAS functionality can be put together to implement the platform error injection and error handling features for Arm64 based platforms

- Three primary steps towards enabling RAS functionality
  - During boot, populating HEST table with RAS error event sources
  - Error Injection (inducing platform error to validate correctness of platform error handling)
  - Error handling (from error interrupt event generation to appropriate handling of the error)

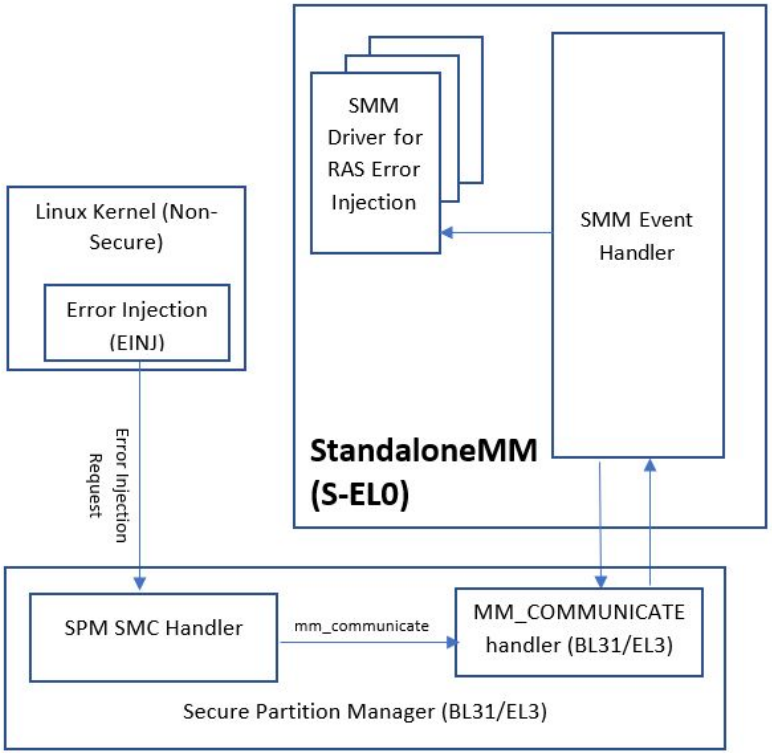
Building HEST ACPI Table During Boot

- HEST ACPI table lists the different platform error event sources supported on the platform
  - Memory Errors, CPU Errors, etc.
  - Kernel learns about supported platform error events using HEST table
- APEI Dxe driver in edk2 is tasked with populating the HEST table with platform error sources.
  - Available error sources are known only to the secure code (StandaloneMM drivers in this case)
  - APEI Dxe Driver uses MM_COMMUNICATE protocol to retrieve the list of supported platform error sources and populates the HEST table
- Platform Specific Considerations
  - Develop StandaloneMM driver(s) for all platform error sources
    - Should implement EFI_MM_RAS_ERROR_SOURCE_INFO_PROTOCOL

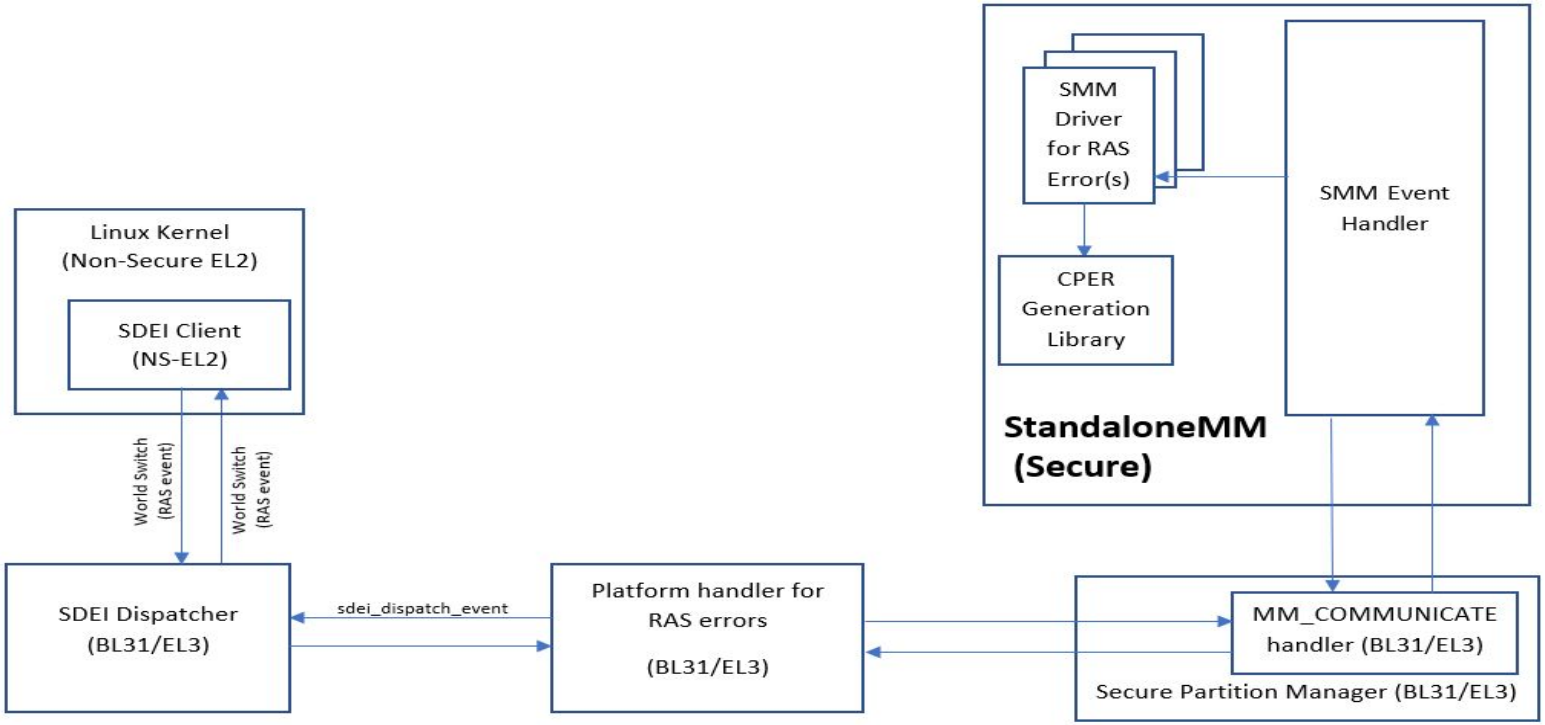


Error Injection

- Helps to validate and demonstrate platform error handling functionality
  - Error injection is initiated from the non-secure code, e.g linux, but platforms allow only secure access to error injection mechanisms.
- Uses MM_COMMUNICATE SMC call to enter StandaloneMM image
  - SPM component services the SMC call and facilitates entry into StandaloneMM image executing at S-EL0
- Control is passed to the Error Injection module in StandaloneMM package
  - Error injection module injects the error in the hardware component, thus resulting in the error interrupt

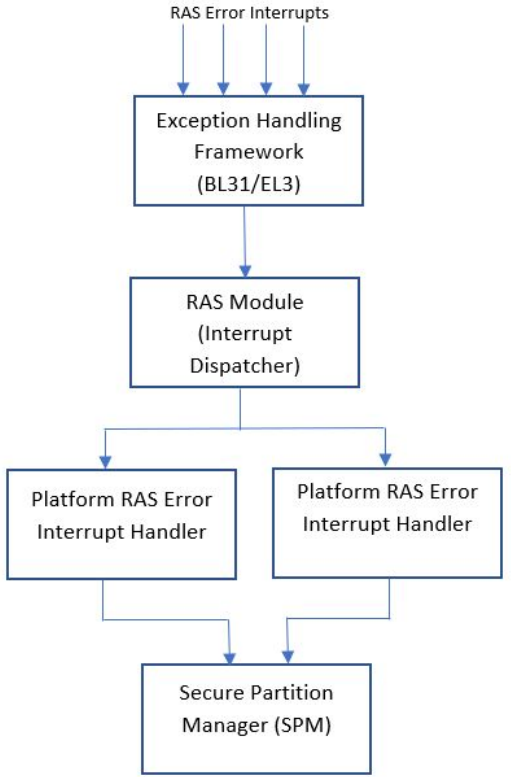


Error Interrupt Event Processing – Execution Flow



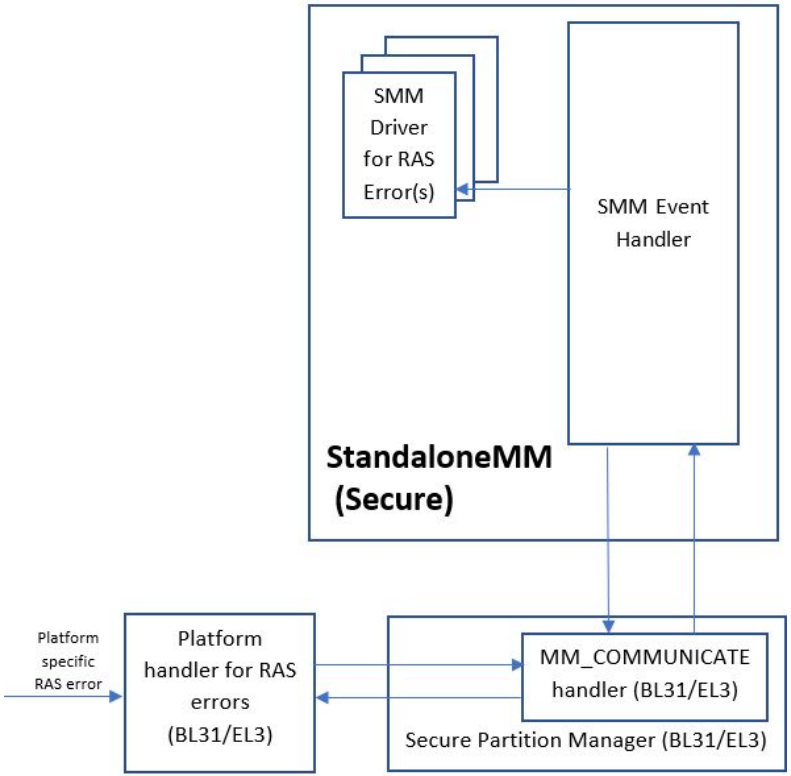
Error Interrupt Event Processing – Dispatching interrupts

- AArch64 allows for firmware-first handling of the error event interrupt
  - Interrupts have to be routed to EL3
- RAS module in BL31 registers with EHF for handling all platform error interrupts
  - Also, provides an interface for platforms to register their respective error handlers for platform specific errors
- Platform's interrupt handler in EL3 is responsible for delegating further processing of the interrupt via the SPM or SDEI interfaces
- Platform Considerations
  - For firmware first handling model, all error interrupts should be routed through Group-0 of GIC interrupt controller
  - Platform code should register with error interrupt handlers with the RAS module during initialization phase of BL31



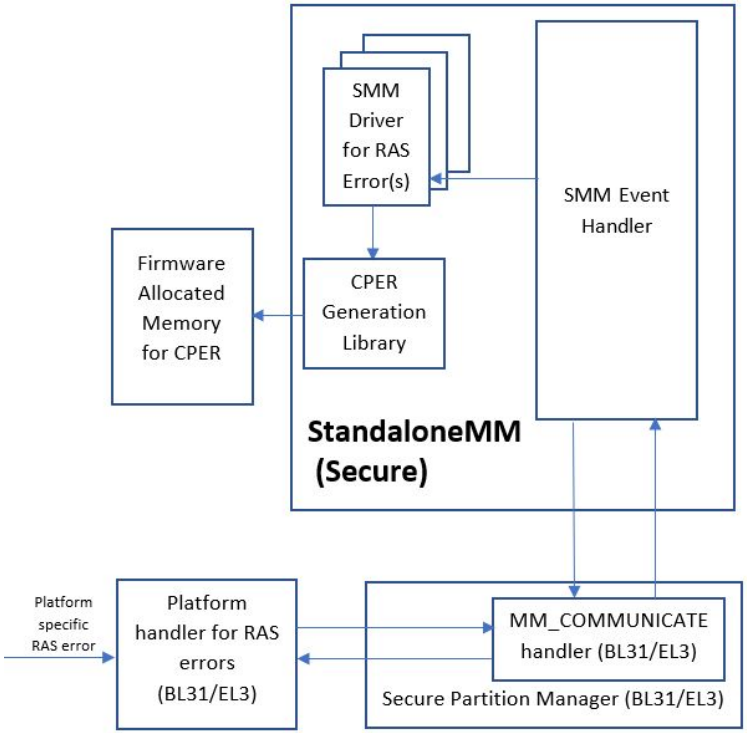
Error Interrupt Event Processing – Path through SPM

- Secure Partition Manager (SPM) is an interface to Secure Partitions implemented at S-EL0
  - Example of a Secure Partition is the StandaloneMM image for handling platform errors
- Platform's error interrupt handler calls the mm_communicate API implemented by SPM to delegate handling of platform errors
- mm_communicate, is a synchronous api, which calls StandaloneMM and on return passes control back to the platform's error interrupt handler for further processing
- Platform Considerations
  - mm_communicate being synchronous facilitates the complete servicing of the error interrupt along with clearing of the interrupt and population of the CPER record before the platform's error interrupt handling proceeds



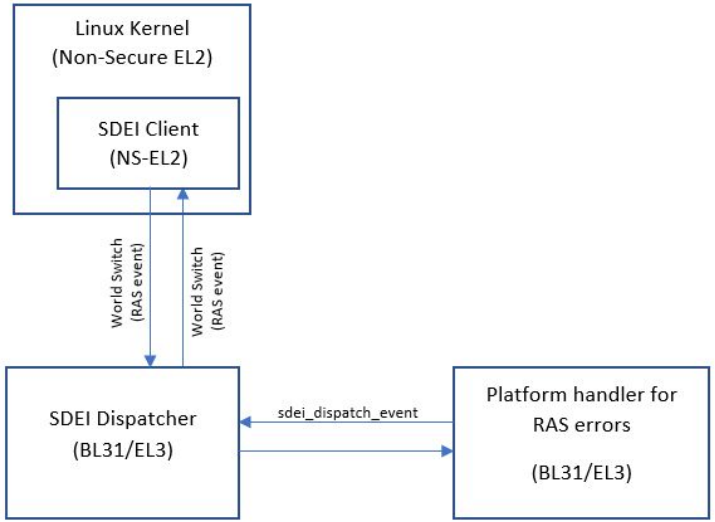
Error Interrupt Event Processing – Role of StandaloneMM

- StandaloneMM package runs as a secure partition in S-EL0
  - Contains drivers for handling platform errors
  - Contains CPER library which is used by the individual error handling drivers for generation of the CPER record
- Platform specific error interrupt handler invokes the corresponding driver module in StandaloneMM image for handling the platform error
- Platform error handling driver in StandaloneMM populates the CPER record with relevant information for subsequent processing
- Platform Considerations
  - Platform error handling driver should clear the error interrupt before returning control back
  - Ensure firmware pre-allocates buffers for CPER for every error source supported



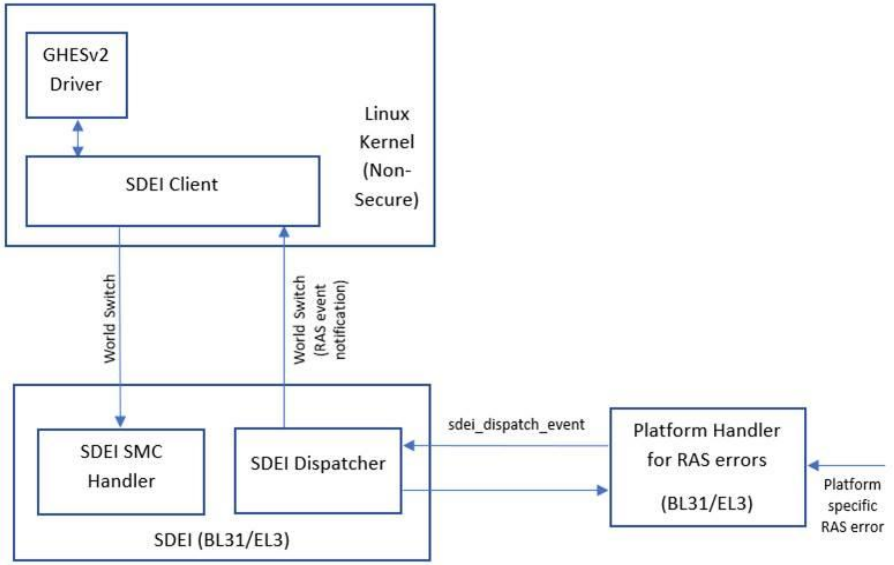
Error Interrupt Event Processing – SDEI Dispatcher

- SDEI dispatcher is responsible for dispatching SDEI events to the SDEI client for processing
  - Platform error interrupt registered as an SDEI event
  - Platform specific error handler calls the SDEI dispatcher api sdei_dispatch_event
- sdei_dispatch_event is a synchronous call to the sdei client, returning back to the platform's error handler
  - saves the current cpu context onto a stack and jumps to the SDEI client, which is part of the linux kernel



Error Interrupt Event Processing – SDEI Client

- Part of Linux kernel in Non Secure world
  - During kernel boot, SDEI client registers its callback with SDEI dispatcher for receiving SDEI event notifications
- Invokes GHES driver in Linux kernel for processing of CPER records
- SDEI client returns back to the SDEI dispatcher, which restores the earlier saved context, returning back to the platform's error interrupt handler. This completes the processing of the error interrupt



Arm64 specific RAS components – Upstream Status

- Secure Partition Manager and SDEI dispatcher upstreamed in TF-A
- Platform RAS error handling code of SGI-575 upstreamed in TF-A
- Upstreaming of StandaloneMM package code for edk2 currently in progress
- SDEI client driver along with GHES driver bits for AARCH64 upstreamed in Linux kernel