White Paper

A Tour beyond BIOS Implementing the ACPI Platform Error Interface with the Unified Extensible Firmware Interface

Palsamy Sakthikumar
Intel Corporation

Vincent J. Zimmer
Intel Corporation

January 2013

Executive Summary

This paper presents a universal model to design the industry standard WHEA/APEI infrastructure in x86 platforms using UEFI firmware.

Table of Contents

Overview 4
Introduction to APEI 4
Goal and Motivation 5
Platform Error Infrastructure 6
Error Classifications 7
ACPI Platform Error Interface 9
Hardware Error Source Table (HEST) 9
ERST (Error Record Serialization table (ERST) 9
Error Injection Table (EINJ) 9
Boot Error Record Table (BERT) 10
Generic Error Status block 10
APEI Error handling models 10
Platform Error Signaling 11
The APEI Error source signaling 11
UEFI Overview 12
UEFI Overview 12
ACPI Table Installation 13
UEFI PI DXE SMM 14
APEI Design in UEFI PI-based host firmware 17
APEI Infrastructure in UEFI 17
UEFI Modular Approach to APEI 18
APEI Architecture Modules 20
Processor/Chipset Modules 20
Platform Specific Modules 21
Conclusion 23
Glossary 24
References 25

Overview

The ACPI Platform Error Interface (APEI)/WHEA (Windows Hardware Error Architecture) is standard defined by ACPI specification for PC and server platforms to report and handle platform errors in graceful way. It's important transfer hardware platform error knowledge to operating systems and management stacks in a meaningful way, in order to perform necessary corrective actions to recover the system or prevent system from sudden failure.

Introduction to APEI

As every platform design and implementation are different, OS needs universal way for error reporting and providing error log information. The APEI (ACPI Platform Error Interface) formally known as WHEA (Windows Hardware Error Architecture) is an ACPI standard. Using APEI and UEFI standards, we can easily implement the APEI in UEFI firmware in a modular fashion, so that most of the UEFI modules are common to all platforms. This commonality makes it easy to port these capabilities to every platform, including mobile, client and servers.

Summary

This section has provided an overview of error reporting and APEI.

Goal and Motivation

The ACPI Platform Error Interface (APEI) is an industry standard interface defined by the Advanced Configuration and Power Interface (ACPI) specification. The platform runtime interface to the platform firmware is largely based upon ACPI, but the Unified Extensible Firmware Interface (UEFI) is a complementary technology that entails a major platform standard in the industry for platform hardware initialization and OS enabling software. Marrying these to produce a universal model is key motivation and goal of this paper. Though platforms are different across different stock keeping units (SKUs) and original equipment manufacturers (OEMs), most of the APEI implementation can be implemented in an interoperable fashion by applying the modularity offered in UEFI.

Error reporting and recovery is a key requirement for the enterprise. In order to hit 5-9's of availability, a machine must have maximum uptime. Error reporting forms an important element of platform Reliability, Availability and Serviceability (RAS) which provides this up-time.

Summary
This section has provided a rationale for leveraging UEFI technology to deploy APEI.

Platform Error Infrastructure

All platforms have different designs and different components that may fail during the life of the system, such as DRAM failures [GOOGLE]. The error detection and handling is a key function to determine the failing components and taking corrective action in a timely fashion in order to prevent any data corruption and also to prolong the life of the system. The error handling is a cooperative activity between the platform hardware (HW), host firmware (UEFI or PC/AT BIOS), and the operating system (OS). The platform errors can be generally classified to Memory subsystem, IO subsystem, Processor, chipset and platform hardware. The role of the host firmware and low level out-of-band, management firmware includes configuring the platform hardware, chipset and processor to detect and report all errors at boot time. In addition to the pre-OS configuration, when the errors are singled at runtime, these firmware agents read error registers in order to analyze and report results to the OS promptly in a manner which is understandable to the OS software. At this point, the OS can initiate appropriate corrective action and/or inform the administrator to take the actions required. Figure 1 shows the typical platform error infrastructure and how the errors get propagated from hardware to the OS with the help of the host firmware.

The errors from all the subsystems get signaled to host firmware or the OS directly. When signaled to the host firmware, the host firmware will read the hardware registers, analyze the component that generated the error and assess the severity of the error. Host firmware will create detailed error log information for the OS and notify the OS of the error's occurrence. Host firmware may additionally generate a platform log and communicate this log to the management controllers out of band firmware for system management purposes. Such controllers include but are not limited to baseboard management controllers (BMC). When the error notification is conveyed to the OS either directly or from host firmware, the OS will inspect the hardware registers or host firmware created error log to further analyze the error and initiate corrective actions. The OS may also inform the administrator to take any corrective action as well.

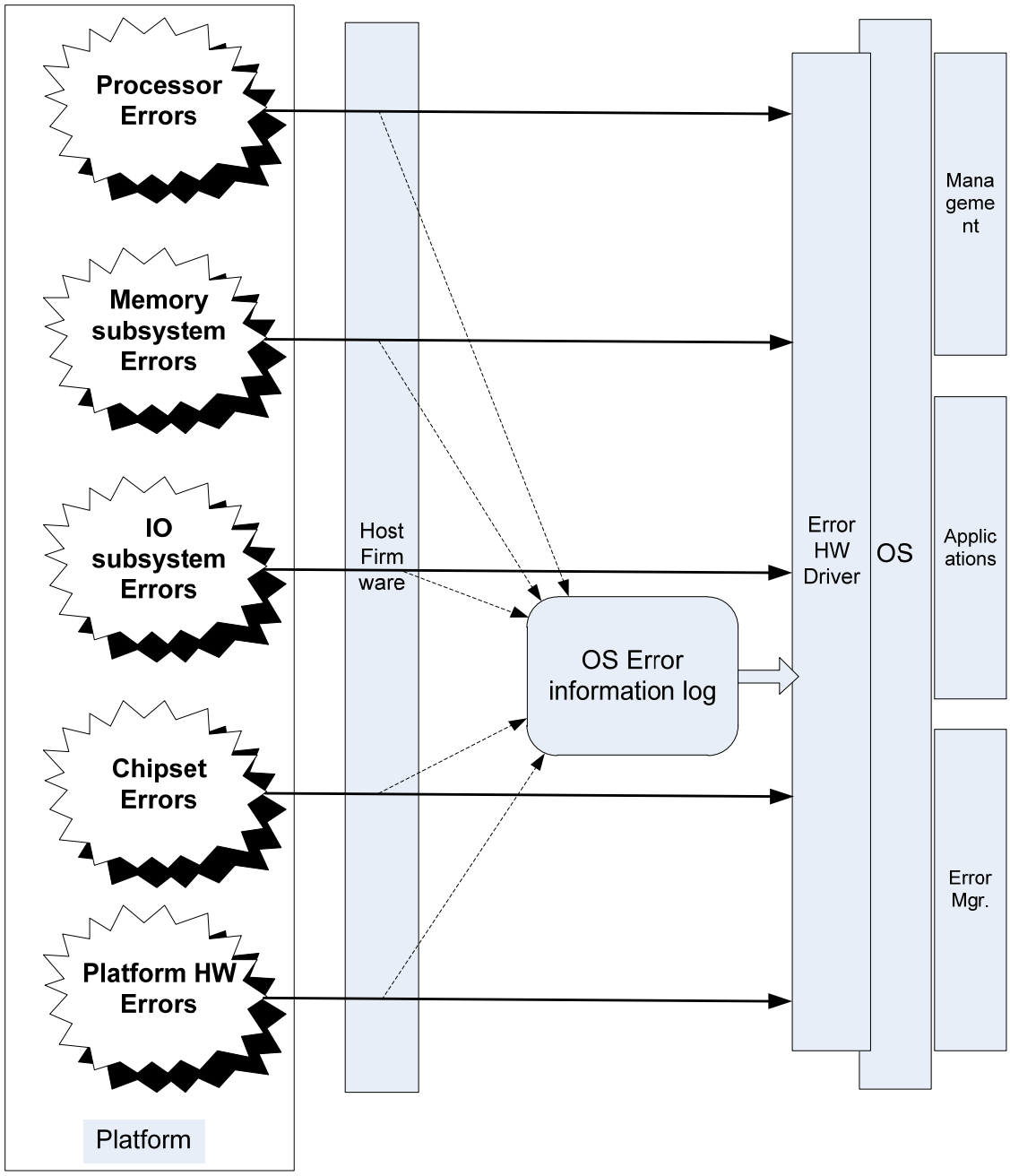


Figure 1 Platform Error Infrastructure

The platform hardware can also provide error injection functionality. For high availability systems such as enterprise servers, the OS needs a software method or API to inject hardware errors into platform for purposes of validating the functionality of the error reporting and logging mechanisms.

Error Classifications

Generally all platform errors can be classified to three categories.

- Corrected errors (CE): These are hardware corrected errors or recovered errors, such as single bit memory errors or memory failover.
- Uncorrected errors (UE): These are errors hardware could not correct or recover. There is a possibility for software or the OS to recover from this error.
- Fatal errors: These are uncorrectable errors from which neither software or hardware could recover. Continuing to run in the face of these errors could make system unreliable.

Some uncorrected errors and all fatal error conditions require immediate containment to prevent any data corruption. This may require a system shut down or reset to recover or replacing the failing component, also referred as FRU (Field replacement unit). As such, it is necessary for the error reporting mechanisms to isolate the error to respective FRU units.

Summary
This section introduced the overall concept of error initiation and reporting.

ACPI Platform Error Interface

As every platform design and implementation are different, OS needs universal way for error reporting and providing error log information. The APEI (ACPI Platform Error Interface) formally known as WHEA (Windows Hardware Error Architecture) is an ACPI standard defining error reporting interfaces to OS. Using APEI and UEFI standards, we can easily implement the APEI in UEFI host firmware modularly, so that most of the UEFI modules are common to all platforms and making it easy portable for every platform including mobile, client and servers. The APEI defines four ACPI tables for declaring platform error infrastructure and Error record container for communicating error information to OS.

Hardware Error Source Table (HEST)

The HEST table enables host firmware to declare all errors that platform component can generate and error signaling for those. The host firmware shall create Error source entries in HEST for each component (such as, processor, PCIe device, PCIe bridge, etc) and each type of error with corresponding error notification mechanism (singling) to OS. These error entries include x86 architectural errors, industry standard errors and generic hardware error source for platform errors. The x86 architectural errors, MCE and CMC, and standard errors PCIe AER, MSI and PCI INTx can be handled by OS natively. The generic hardware error source can be used for all firmware 1st errors and platform errors (such as memory, board logic) that do not have OS native signaling, so they have to use platform signaling SCI or NMI.

ERST (Error Record Serialization table (ERST)

The ERST table provides a generic interface for the OS to store and retrieve error records in the platform persistent storage, such as NVRAM (Non-volatile RAM). The error records stored through this interface shall persist across system resets till OS clears it. The OS will use this interface store error information during critical error handling for later extensive error analysis. Host firmware shall provide serialization instruction using ACPI specification defined actions to facilitate read, write and clear error records.

Error Injection Table (EINJ)

One of the important functions required in implementing the error is the ability to inject error conditions by the OS to evaluate the correct functionality of the entire error handling in the platform hardware, host firmware and the OS. The EINJ table interface facilitates error injection from the OS. The host firmware shall provide at minimum one error injection method per error type supported in the platform. The host firmware will provide the generic error serialization instructions to trigger the error in the hardware.

Boot Error Record Table (BERT)

The BERT table provides the interface to report errors that occurred system boot time. In addition BERT also can be used to report fatal errors that resulted in surprise system reset or shutdown before OS had the chance to see the error. The host firmware shall build this table with error record entries for each error that occurred.

Generic Error Status block

The Error status block is the standard error data container for communicating detailed error information log to the OS for generic error sources listed in HEST. The generic error source may include firmware 1st errors and non-standard platform errors. The error status block can log multiple errors until the OS reads them and handles these errors appropriately. The standard x86 MCE and CMC error log information are presented in Machine check bank registers in the processor. In the same fashion, the PCIe AER standard error information is presented in PCIe/PCI device's or bridge's configuration register space. Given these mechanisms, the OS can directly read and analyze the data. The error record structure and information are specified in UEFI 2.3.1 [UEFI] specification.

APEI Error handling models

APEI offers two error models, Firmware first model and OS Native model.
Firmware 1st is used when the host firmware needs to initially examine the error and attempt recovery or corrective action in an OS transparent way. This model is also used when certain OEMs want more control over error handling before the OS takes control, such as for purposes of executing some management functions. In the Firmware 1st model, all errors are initially signaled to the host firmware via SMI or other General Purpose Input (GPI) events. Then host firmware analyzes and decides what to do, and at the end of the flow creates a detailed APEI error log with FRU information to OS. Finally, the host firmware will then signal the OS about the existence of the error via SCI, NMI, or other interrupts.

The OS native model, on the other hand, provides handling of the error directly by the OS or OS level software by directly accessing the hardware registers and analyzing the error. This requires standard architecture in the hardware for providing error information in the hardware and signaling, for e.g. industry standard PCIe AER and x86 MCA architectures. This model takes the burden off of host firmware.

Platforms can use combined model also, where some errors are handled firmware 1st, some natively and some both (a.k.a. parallel model). Many of the servers employ this combined model for better handling and managing the server better via remotely.

Note: For the Firmware 1st model, the host firmware has to program the processor and chipset to trigger SMI upon hardware error pins or upon processor/chipset error messages. After host firmware handling the errors 1st and building error records for OS, it will generate SCI or NMI to OS.

Platform Error Signaling

Whenever an error detected in the platform hardware, the hardware error will be signaled to host firmware or OS or both. In x86 architecture, the error signals to host firmware are hardware error pins, hardware error registers or error messages. These error signals will be typically routed to trigger System management interrupt (SMI) so that host firmware can handle the errors in a secluded environment in System management mode (SMM). The error signaling to the OS are interrupts. There interrupts are x86 architecture interrupts, such as MCE (Machine check architecture error), Corrected Machine Check (CMC) interrupts or standard IO interrupts such as, PCE AER (Advanced error reporting) MSI (Message signaling interrupt), PCI interrupt (INTx), or platform NMI (non-maskable interrupt) and ACPI SCI (System control interrupt). Upon these interrupts being activated the OS will handle errors in a processor driver, ACPI driver or in an APEI driver.

The APEI Error source signaling

The following table shows how the various subsystem errors are signaled to APEI domain to host firmware and OS in different error reporting model.

	Firmware 1st Model		Native reporting model
Errors	Signal to host firmware	Signal to OS	Signal to OS
Processor CE	SMI	SCI	CMC
Processor UE & Fatal	SMI	NMI	MCE
Memory CE	SMI	SCI	CMC
Memory UE & Fatal	SMI	NMI	MCE
IO CE	SMI	SCI	PCIe AER - MSI/INTx
IO UE/Fatal	SMI	NMI	PCIe AER - MSI/INTx
Chipset CE	SMI	SCI	CMC or None
Chipset UE/Fatal	SMI	NMI	MCE or None
Hardware CE	SMI	SCI	None
Hardware UE/Fatal	SMI	NMI	None

Summary

This section introduced APEI and its constituent elements.

UEFI Overview

The implementation of the APEI elements described herein is based upon host firmware based upon the Unified Extensible Firmware Interface (UEFI) Platform Initialization (PI) specification sets. The UEFI specifications are purposely silent on construction intent and policy. Instead, the UEFI specification is a pure interface specification that admits to conformance testing of the API's.

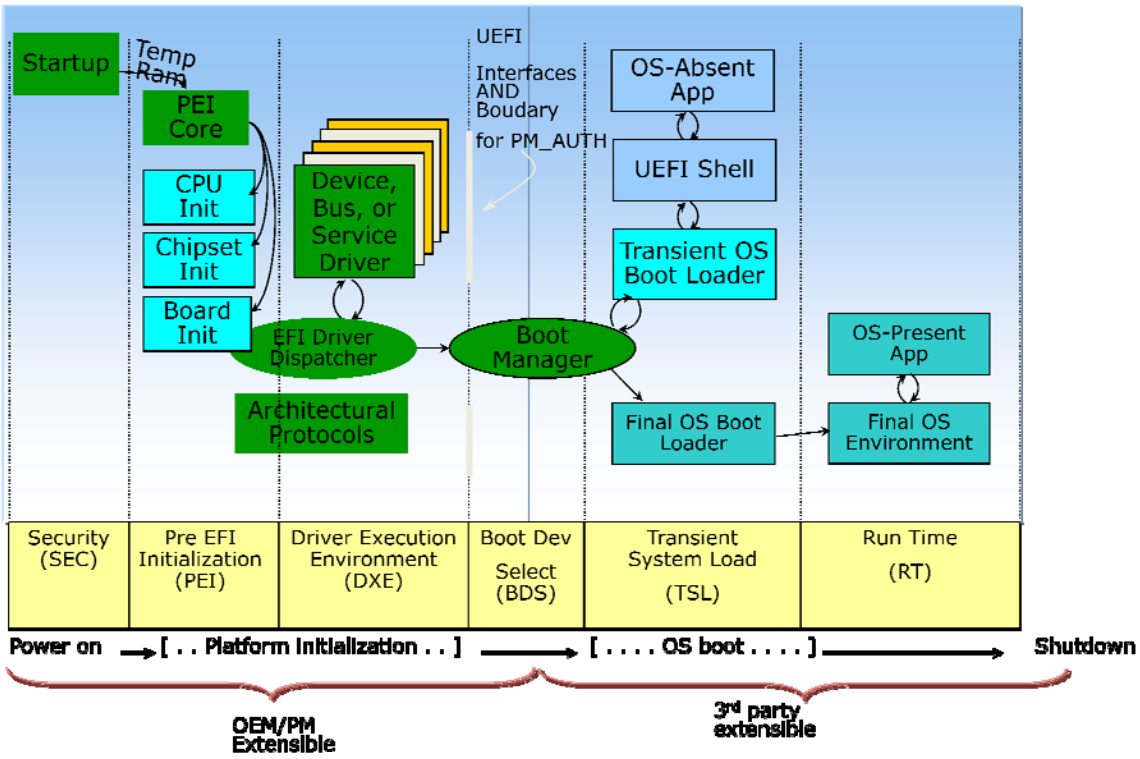


Figure 2 UEFI PI Boot Flow

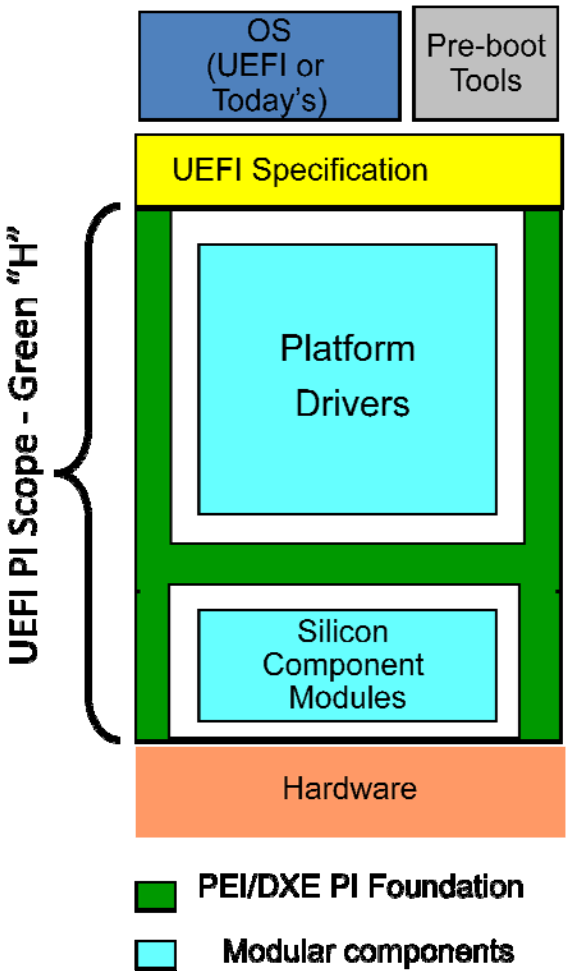


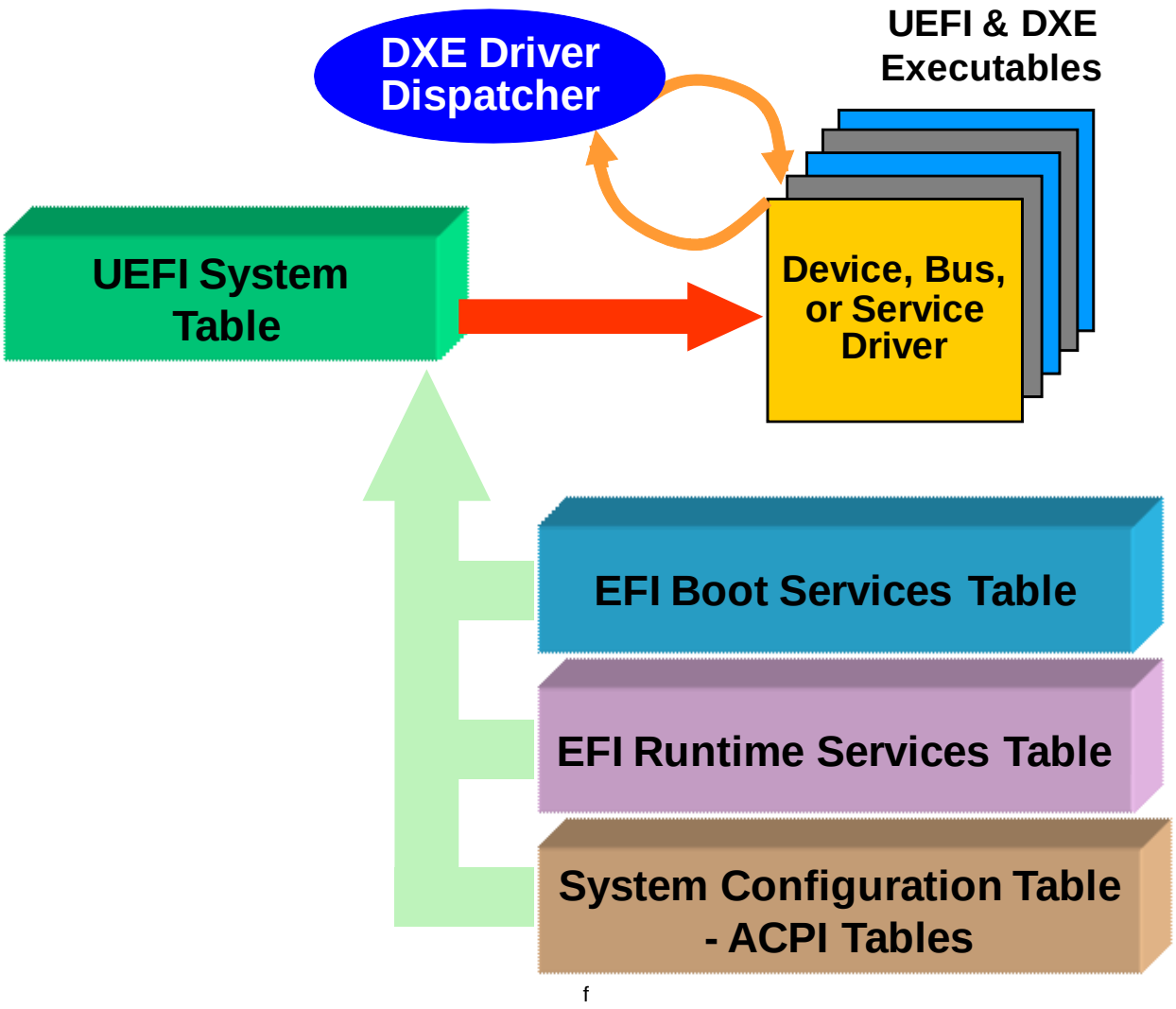
Figure 3 UEFI PI Software Layering Diagram

As you can see from the boot flow in Figure 2 above, the SEC, PEI, and DXE all run prior to having the UEFI services available. The UEFI specification describes a set of interfaces to the platform, and the UEFI PI DXE phase acts as the UEFI core. In fact, the DXE core is the preferred embodiment of the UEFI interfaces. The SEC, PEI, and DXE components are provided by the platform manufacturer. These PI elements are also referred to as the 'Green H', as shown in Figure 3 above.

ACPI Table Installation

One set of DXE drivers have knowledge of the APEI infrastructure and the platform topology. These drivers populate the respective ACPI tables mentioned earlier. A pointer to the ACPI tables can be found via a reference in the UEFI System Table's set of system configuration

tables. ACPI tables have a GUID definition that can be found in the UEFI specification [UEFI]. See Figure 4 below



f

Figure 4 UEFI PI flow and ACPI tables

UEFI PI DXE SMM

Of the DXE components, there is a class of DXE Drivers called DXE SMM. The DXE SMM drivers support the OS runtime interactions with the platform. This can include SMI invocations from ASL or via pin activations from the CPU and chipset.

The UEFI PI boot flow augmented with the PI SMM driver loading is shown in Figure 5 below.

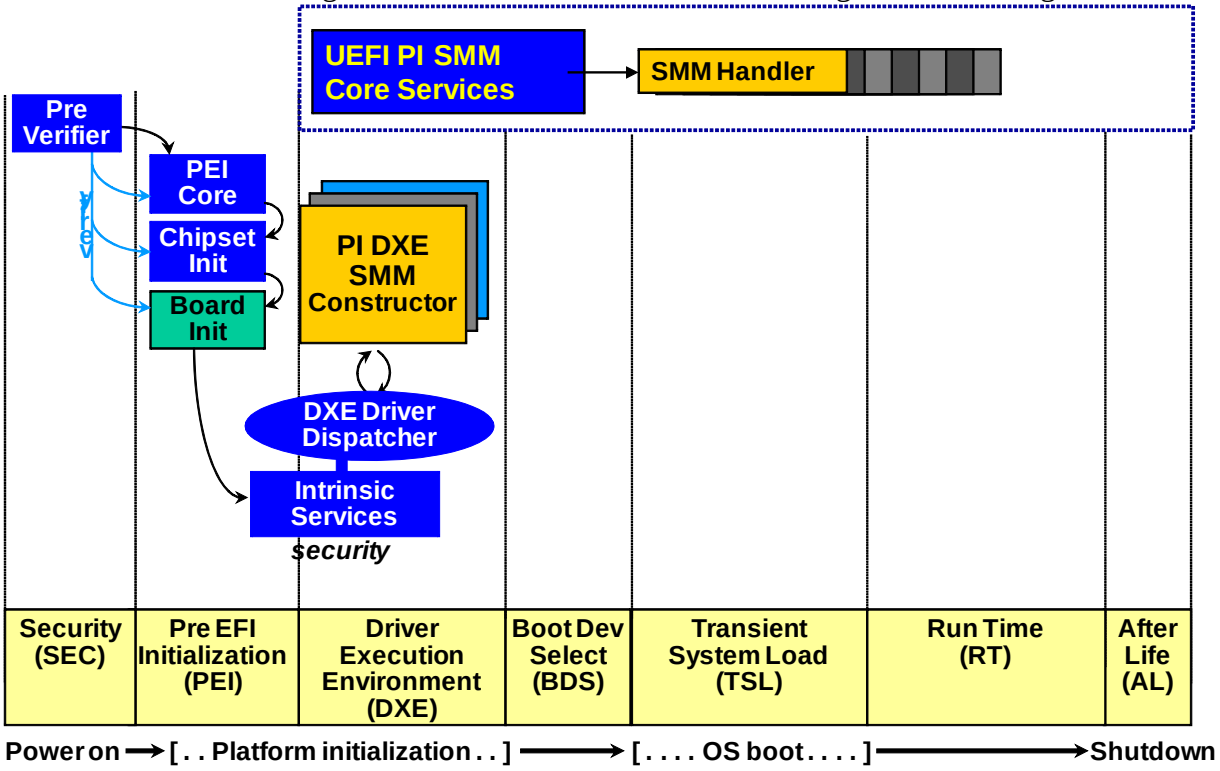


Figure 5 UEFI PI SMM Driver load

There can be a plurality of DXE SMM drivers, including but not limited to the APEI support modules. The relationship of a DXE SMM driver to the PI DXE SMM core is shown in Figure 6

below.

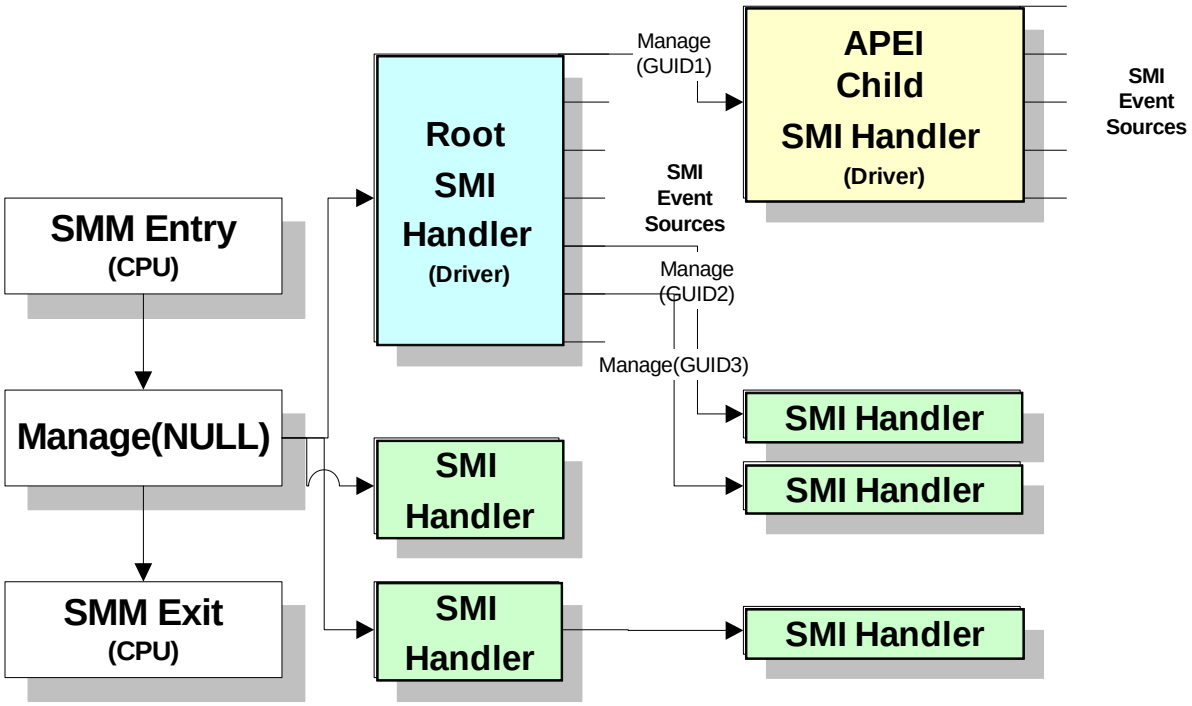


Figure 6 SMM driver topology

The PI SMM DXE drivers are PE/COFF executables like other DXE and UEFI drivers. As such, the functionality for different capabilities can be delivered as separate UEFI PI packaging-specification based source modules, or more likely going forward, as separate binaries. These source packages or binaries can be factored such that they are unique to pure software elements like generic APEI or error logging processing, versus other drivers which are specific to a given chipset family error reporting hardware.

Summary

This section has provided an overview of UEFI, PI, and the mechanisms by which ACPI tables and PI SMM drivers are deployed in the platform.

APEI Design in UEFI PI-based host firmware

Implementing a modular APEI error infrastructure in a system involves support in platform hardware, UEFI host firmware and the OS. Each entity has multiple components working cooperatively as part of supporting APEI.

APEI Infrastructure in UEFI

Now that the generic overview of UEFI and the PI infrastructure has been provided, Figure 7 below shows all components and their interactions to implement the APEI error interfaces. This modular infrastructure facilitates graceful and collective way of handling errors and recovering from errors. In addition provides portable design to any platform independent of individual design harnessing UEFI and UEFI PI host firmware standards. This section will focus on the UEFI PI host firmware components and how they can be designed in a common and platform independent model.

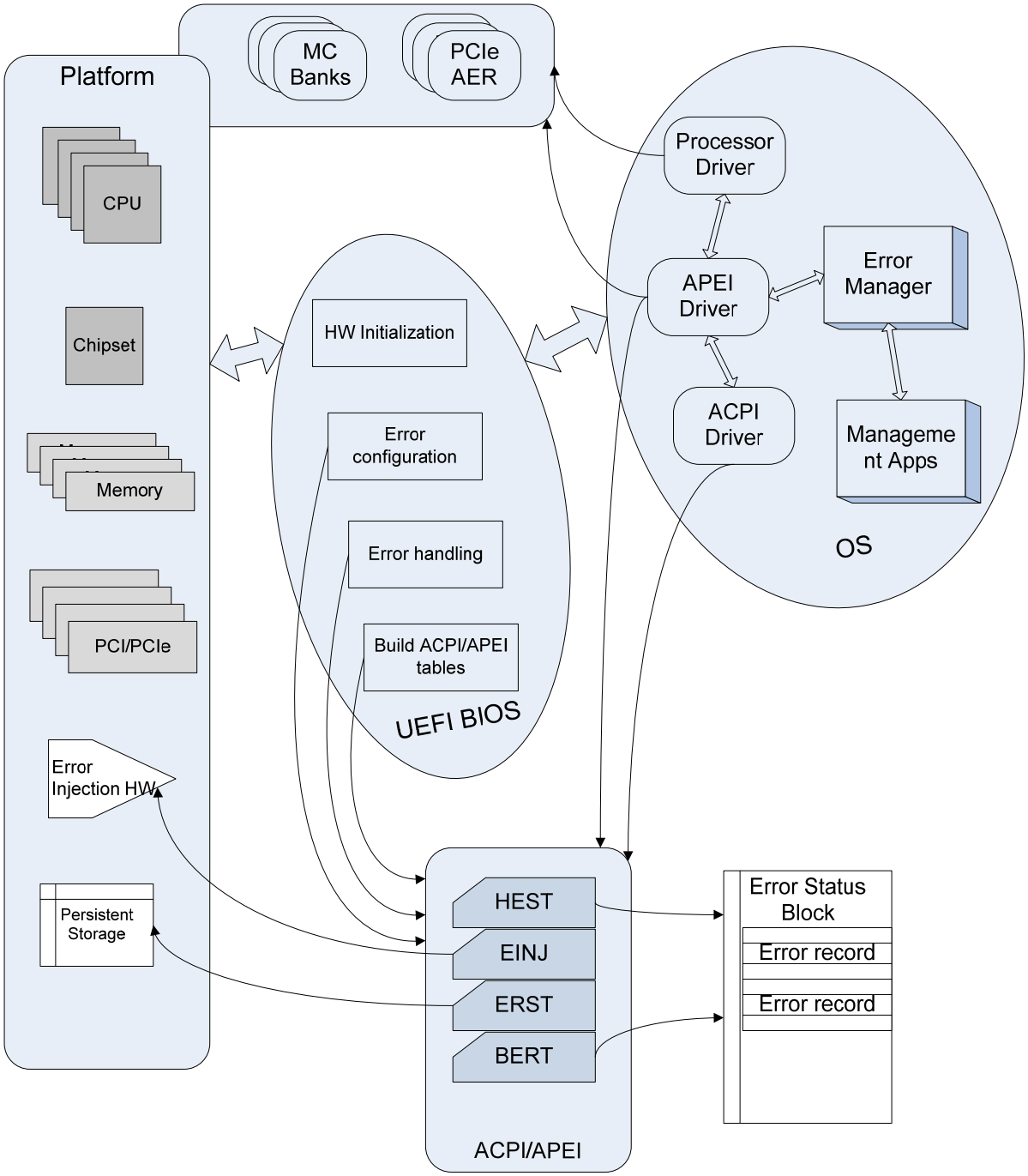


Figure 7 APEI Infrastructures in a UEFI host firmware-based System

UEFI Modular Approach to APEI

This section describes about different UEFI modules and design that segregates platform agnostic modules from platform design dependant modules. By following this UEFI implementation design, only the platform dependent modules will have to be ported from

platform to platform and retain most of the APEI implementation same for all implementations. Figure 8 below shows the different modules that APEI implementation can follow using UEFI PI standards. The following section will describe each module in details.

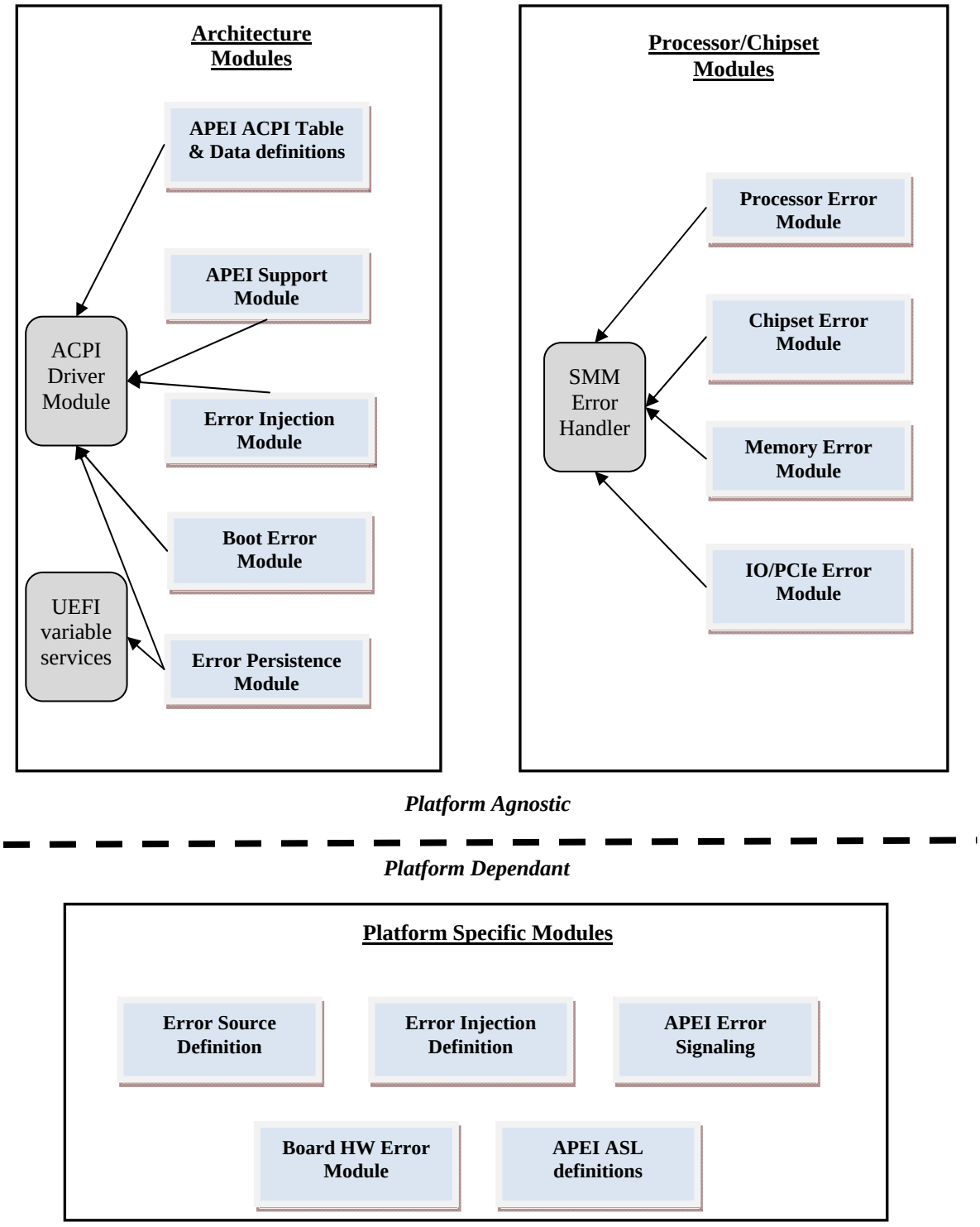


Figure 8 APEI Modules in UEFI host firmware

The APEI module in UEFI host firmware can classified to three types of modules as shown in figure 8. The Architecture modules are part of UEFI PI infrastructure and common in all implementations. The Processor/Chipset modules also can be part of UEFI PI infrastructure and common to one processor architecture, for e.g. Intel x86 architecture. This may need little porting for various processor/chipset generations. The Platform specific modules are platform design dependant and modified for each platform. This modular design reduces the modification greatly for platform to platform.

APEI Architecture Modules

The AEPI architecture modules will use exiting UEFI infrastructure such as ACPI driver and variable services to support APEI standard interface and communication to the OS.

APEI Tables and Data definition
This module contains the new ACPI table definitions for APEI and data structure for Error record structures, Error injection and Error persistence interfaces.

APEI Support Module
APEI support module is the collector all AEPI support in the platform and also provider of creating APEI table, data structure and interfaces for other modules. In addition this module also will create necessary runtime interfaces for OS APEI driver. This module will also enquire platform modules and create all Error sources that are supported in HEST table. The APEI support module is also responsible to create and publish EINJ, ERST and BERT tables on behalf of other modules by using UEFI ACPI standard protocols. This module also will provide runtime and boot interfaces to create Error records for Os in Error status block container.

Error Injection Module
The Error injection module will use APEI support module interfaces and install the support error injection entries and error injection support calls using the platform specific module interfaces.

Boot Error Module
The Boot error module will use processor/chipset modules to scan for boot time errors and last boot errors that were not reported to OS. Once it determines and collects the error, it will use APEI module interface to create error record BERT for each errors.

Error Persistence Module
The Error persistence module provide runtime interface to OS for storing/retrieving error records. This module can use UEFI variable services for persistent storage for portability. In addition this module will also provide proper information to APEI module to create ERST table.

Processor/Chipset Modules

Processor chipset modules will provide the supporting infrastructure to read hardware registers for determining error and collecting error information from various components in the system and also build detailed error information log for the OS using APEI Support module interfaces.

The top level error handling is done in runtime by SMM error handler in x86 architecture platforms. This main error handler will use the other subsystem error handlers to detect and process respective errors in addition to providing the error log record. The error record shall be created for all CE, UE and Fatal errors.

Processor Error module
The Processor error module is responsible to detect and analyze processor data errors and internal errors and perform any recover action that can be done. This module also will create UEFI specified processor error record log and pass it to main error handler.

Chipset Error Module
The Chipset error module is responsible to detect and analyze errors in chipset devices such legacy bridged and integrated devices and if possible perform any recovery actions. This module also will create UEFI specified processor/chipset error record log and pass it to main error handler.

Memory Error Module
The memory error module is responsible to detect and analyze memory subsystem errors including data errors, memory device errors, memory channel errors and redundancy failures. This module will also initiate any recovery action, such as sparing and mirror failover. This module also will create UEFI specified processor error record log for all memory errors and pass it to main error handler.

IO/PCIe Error Module
The IO/PCIe error module is responsible to detect and process errors in the IO sub system including data errors, device errors and PCIe link errors. It will determine and perform if any recovery action possible such as reset PCIe device. This module also will create UEFI specified processor error record log and pass it to main error handler.

Platform Specific Modules

The platform specific modules are the modules that will require modification from platform to platform. These define platform specific policies and support component errors and error injection capabilities and hardware error signaling to OS.

Error Source definition
The module will define the support error sources in the platform. These include both Os native errors and firmware 1st errors. The non-standard APEI errors can be always reported as generic hardware errors. In addition this module will list out error notification methods for each error. The APEI support module will use this platform specific information to create necessary APEI tables and interfaces to OS.

Error Injection definition
The Error injection definition will list out the types of error injection support in the platform and platform specific error injection mechanism. In general implementation most of the error

injection mechanisms can be moved to Processor/Chipset modules, so that they can be retained part of common modules.

APEI Error signaling
This module will implement the platform hardware specific interfaces to trigger AEPI hardware error notification, for e.g. SCI, NMI, to OS.

Board Hardware Error module
This module is for any additional platform board specific error reporting and handling. The errors handled by this module can be reported as generic hardware error sources and logged same way.

APEI ASL definitions
APEI ASL module will be platform specific module as it will contain the ASL code and device definition for support APEI. Depending on the platform implementation and policy overrides this module has to be ported from platform to platform.

Summary
This section provided an overview of the various APEI UEFI PI modules.

Conclusion

Error reporting on modern platforms entails several elements whose activities must be coordinated. These include platform design, host firmware construction, and error-reporting aware facilities in the OS and OS software. To deliver this class of functionality in the platform and host firmware, the interface definitions of ACPI and UEFI are leveraged so that an OS can be designed against these standards and not a specific vendor's implementation. Underneath the abstractions afforded by ACPI and UEFI, though, the modularity of the UEFI PI standards and rich open source implementations like the UEFI Development Kit [EDK2] can be used to support a rich set of vendor platforms and achieve high code re-use. This re-use helps with time-to-market by not having to re-engineer all of the firmware elements and also accrue the validation efforts on stable binaries of drivers that do not change across SKU's and generations of platforms.

Glossary

ACPI – Advanced Configuration and Power Interface. Static tables and ACPI Machine Language (AML) interpreted byte code. Preferred OS runtime interface to the platform.

APEI – ACPI Platform Error Interface. Industry standard platform error interface to OS.

AML – ACPI Machine Language

APEI – ACPI Platform Error Interface.

ASL – ACPI Source Language.

BIOS – Basic Input Output System. Firmware that executes on the host CPUs. Can be PC/AT BIOS or UEFI PI-based.

BMC – Baseboard Management Controller.

CMC – Corrected Machine Check.

MCA – Machine Check Architecture. Error signaling mechanism for x86 CPU's.

MSI – Message Signaled Interrupt.

PI – Platform Initialization. Volume 1-5 of the UEFI PI specifications. Volume 3 includes an execution mode for SMM.

SMM – System Management Mode. x86 CPU operational mode that is isolated from and transparent to the operating system runtime

UEFI – Unified Extensible Firmware Interface. Firmware interface between the platform and the operating system. Predominate interfaces are in the boot services (BS) or pre-OS. Few runtime (RT) services.

WHEA – Windows Hardware Error Architecture. Various error management technologies in Windows.

References

[ACPI] ACPI Specification Revision 5.0 www.acpi.info

[EDK2] UEFI Developer Kit www.tianocore.org

[FRAMEWORK] Intel Framework Specifications www.intel.com/technology/framework

[GOOGLE] Bianca Schroeder and Eduardo Pinheiro and Wolf-Dietrich Weber
"DRAM Errors in the Wild: A Large-Scale Field Study," Sigmetrics 2009
http://research.google.com/pubs/pub35162.html

[UEFI] Unified Extensible Firmware Interface (UEFI) Specification, Version 2.3.1c
www.uefi.org

[UEFI Book] Zimmer,, et al, "Beyond BIOS: Developing with the Unified Extensible Firmware Interface," 2nd edition, Intel Press, January 2011

[UEFI Overview] Zimmer, Rothman, Hale, "UEFI: From Reset Vector to Operating System," Chapter 3 of *Hardware-Dependent Software*, Springer, February 2009

[UEFI PI Specification] UEFI Platform Initialization (PI) Specifications, volumes 1-5, Version 1.2 www.uefi.org

[WHEA] Windows Hardware Error Architecture.
http://msdn.microsoft.com/en-us/library/windows/hardware/gg463286.aspx

Authors

Palsamy Sakthikumar (psakthik@gmail.com) is a platform architect with the Data Center Group at Intel Corporation.

Vincent J. Zimmer (vincent.zimmer@intel.com) is a Principal Engineer with the Software and Services Group at Intel Corporation.