Open Compute Project

Project Olympus Intel® Xeon® Scalable Processor BIOS Specification

Author:

Mallik Bulusu, Principal Firmware Engineering Manager, Microsoft

Revision History

Date	Description
11/1/17	Version 1.0

Contents

1 Introduction

1.1 Purpose of the Document

The System BIOS is an essential platform ingredient which is responsible for platform initialization that must be completed before booting of an operating system. Thus, the BIOS execution phase of the boot process is often referred to as pre-boot phase. The purpose of this document is to provide guidance on BIOS development for Intel® Xeon® Scalable Platform that complies with WCS specifications.

1.2 Structure of the Document

Chapter 1 provides introduction to the document.

Chapter 2 provides hardware overview.

Chapter 3 provides BIOS design overview.

Chapter 4 provides processor support requirements.

Chapter 5 provides memory support requirements.

Chapter 6 provides Integrated IO (IIO) support requirements.

Chapter 7 provides PCH support requirements.

Chapter 8 provides manageability requirements.

Chapter 9 provides NVDIMM support requirements.

Chapter 10 provides networking requirements.

Chapter 11 provides security requirements.

Chapter 12 provides error handling requirements.

Chapter 13 provides firmware update requirements.

Chapter 14 provides OS boot support requirements.

Chapter 15 provides BIOS POST codes.

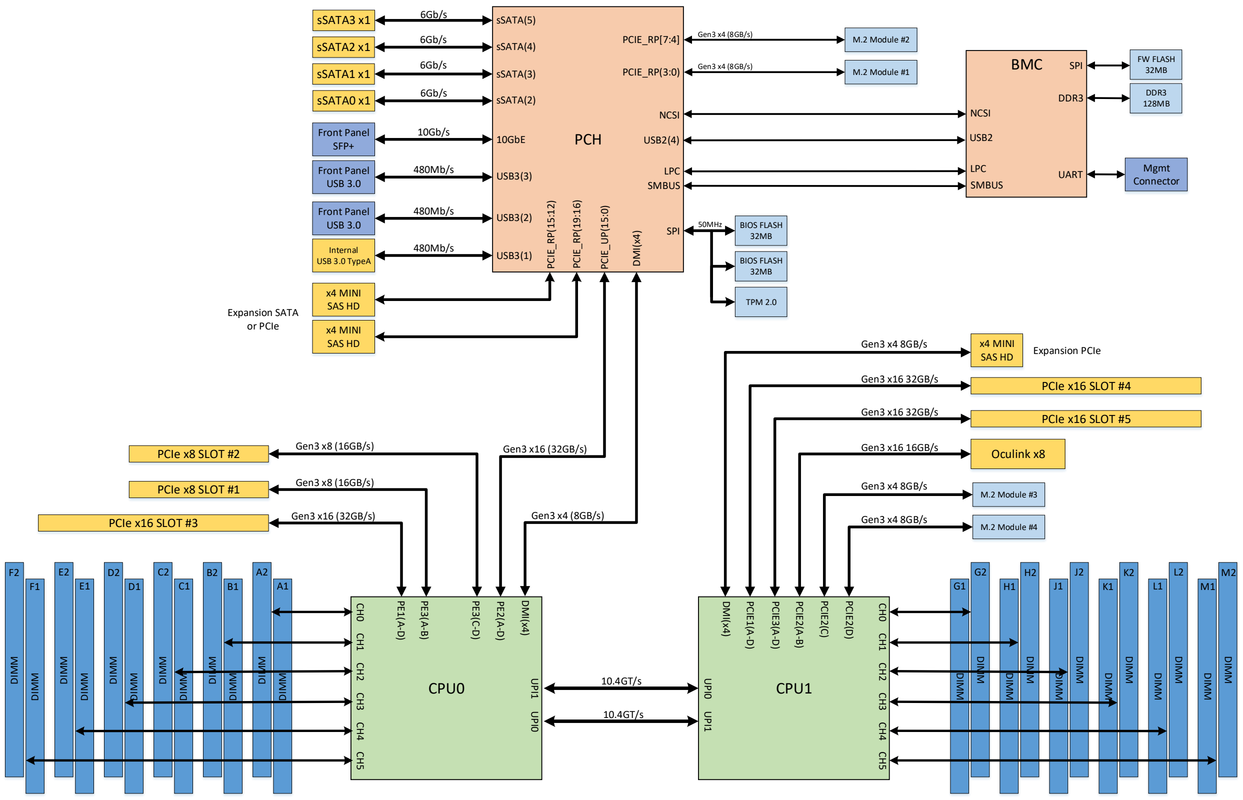
Chapter 16 provides FRU information.

Chapter 17 provides descriptions of commonly used acronyms.

2 Hardware Overview

2.1 Hardware Block Diagram

The hardware block diagram for a WCS Intel® Xeon® Scalable Platform mother board is as shown below:



For additional details on WCS Intel® Xeon® Scalable Platform Hardware design, refer to WCS Hardware Intel® Xeon® Scalable Platform Motherboard Specification.

2.2 Key Hardware Features

Key features for WCS Intel® Xeon® Scalable Platform:

- Support for up to 3 standard x16 PCIe slots
  - 3 FH/HL in 1U
  - 2 GP-GPU + 1 PCIe card in 2U
- Support for non-standard PCIe x24 (using 1 standard x16 slots + Oculink x8 cable)
- Support for up to 2 LP PCIe x8 slots

  - Can support up to 4 M.2 Modules through riser cards
- Support for 4 onboard M.2 Modules
- Support for 4 SATA HDDs or SSDs
- Support for x8 SATA expansion or x8 PCIe expansion
- Support for x4 PCI expansion
- Support for 1x10 GbE from PCH
- Support for Blade Management (AST1250)
- Support for x16 bandwidth QAT
- Support for Intel® Omni-Path Architecture PCIe Adapter (Storm Lake)
- Support for Datasafe storage
  - NVDIMM, RAID, M.2, Intel® Optane™
- Support for WCS rack management

2.3 Chipset & Core Logic

Processor	Intel® Xeon® Scalable Processors
Core Chipset	Intel® C620 series chipset (PCH)
Memory	DDR4: 24 x DIMM 12 DIMMs per CPU 2 DIMMs per Channel
SAS Controller	Intel® C620 series chipset (PCH)
Slots	2 PCIe x8 slots – Supports PCIe M.2 riser cards 3 PCIe x16 slots – Supports standard PCIe x16 cards
BIOS	16-MB SPI flash chip
BMC	BMC-lite BMC Aspeed AST1250 Intel® Innovation Engine
LOM	1x10GbE on QSFP+ Connector

2.4 Bus Configuration

Bus Type	# of slots/Ports
☐ PCI	
☐ PCI 64-bit	
☐ PCI-X	

Bus Type	# of slots/Ports
☒ PCI Express	7
☒ USB 1.1/2.0/3.0	2 x 2.0 & 1 x 2.0
☐ Other	

2.5 Memory Configuration

Memory module type	DIMM

System Memory Standard	DDR4
Max. System Memory (GB)	1536 GB
No. of Memory Slots	24
BIOS Flash ROM Size (MB)	16MB Minimum.
BIOS ROM Flash Device(s)	32 MB, two parts for backup
BIOS ROM Flash Interface	Serial Peripheral Interface (SPI)

2.6 System Ports

USB	4 x 2.0
UART	BMC 3 x UART 16550
SATA	10 x SATA 3 Ports

2.7 PCI Routing Information

ODM to specify how the INT/PIRQ pins from the chipset are connected to each slot/device

Slot Number (or Onboard Device)	IDSEL or DEV. #	Bus #	PIRQ 0 (INT A)	PIRQ 1 (INT B)	PIRQ 2 (INT C)	PIRQ 3 (INT D)	APIC routing

2.8 PCH GPIO Configuration

GPIO pin mappings are listed in the WCS-Software-GPIO appendix. This document will detail the ASPEED 1250 and Intel® C620 series chipset (PCH) pin assignments for WCS compliance.

2.9 System Clock Configuration

ODM to define if the System Clock Initialization will need to be supplied unless the System Clock Initialization is identical to an AMI supported CRB or no programming is required.

Are the system clocks run by a clock controller? (Example: CK505)	
☐ Yes (if Yes, Continue)	☐ No (if No, Done with table)
System clock programming should be identical to the programming used on the CRB.	

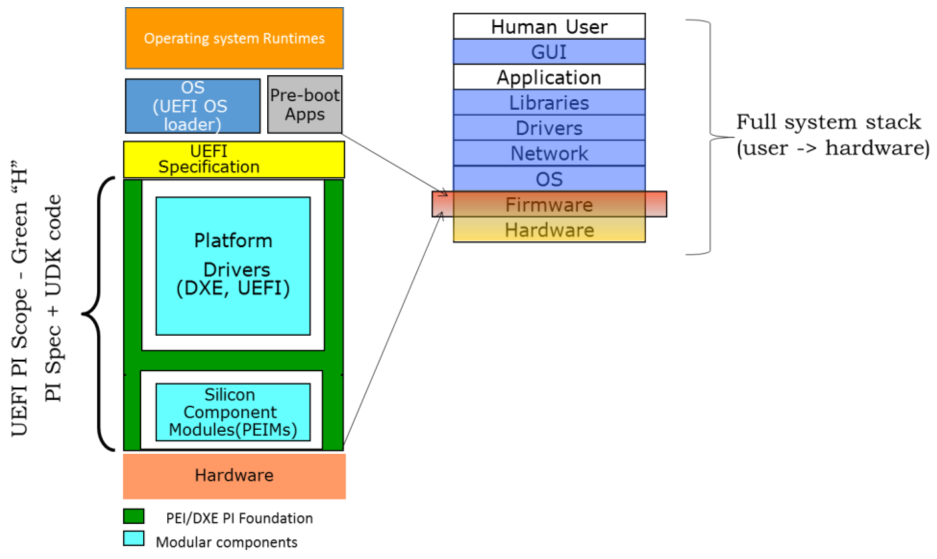
2.10 ACPI Configuration

Sleep State Support (list requirements)	S0, S5

3 BIOS Design Overview

3.1 BIOS Core Internals

The WCS BIOS for the Intel® Xeon® Scalable Platform generation must be compliant with UEFI 2.5 and PI 1.4 specifications. The WCS Intel® Xeon® Scalable Platform BIOS must implement SEC, PEI, DXE and BDS phases of Tiano model (see Figure below).



The WCS Intel® Xeon® Scalable Platform BIOS must implement UEFI boot and runtime services as well as UEFI defined boot flows (see below) must be must be supported. The BIOS must support both legacy (through CSM) and UEFI boot modes. The BIOS must also have support core support for SMM handling.

3.2 Aptio 5.x Support Utilities

These utilities are required during manufacturing, in debug, deployment, and field update phases.

3.2.1 Aptio 5.x Support Utilities: ROM Maintenance & Modification

- AMISCE
- ChangeLogo
- MMTool

3.2.2 Aptio 4.x Support Utilities: Manufacturing and Field Deployment

Utility Name	Utility Description	Supported OS
AMI Firmware Update (AFU)	Reprogram platform flash memory with a new ROM image. Update main BIOS image, boot block, or "ROM holes" based on user input	MS-DOS Microsoft Windows (32-bit) Microsoft Windows (64-bit) Microsoft WinPE
AMISCE	AMISCE is a command line tool used to import and export NVRAM setup data, allowing users to modify BIOS setup values without entering the setup interface. AMISCE uses NVRAM and the HII database (setup questions and related information). Data exported from the current system BIOS is stored in a text based script file. The script file can be modified and imported to update the NVRAM setup variables.	MS-DOS Microsoft Windows (32-bit) Microsoft Windows (64-bit) Microsoft WinPE

3.3 System Management BIOS (SMBIOS)

The BIOS should provide support for the System Management BIOS Reference Specification, Version 3.0. For detailed information on the System Management BIOS requirements refer to the DMTF web page at http://www.dmtf.org/.

The BIOS must implement the following SMBIOS tables:

Type	Structure
Type 0	BIOS Information
Type 1	System Information
Type 2	Base board Information
Type 3	System Enclosure or Chassis
Type 4	Processor Information
Type 7	Cache Information
Type 8	Port Connector Information
Type 9	System Slots
Type 11	OEM Strings
Type 13	BIOS Language Information
Type 16	Physical Memory Array
Type 17	Memory Device
Type 19	Memory Array Mapped Address
Type 38	IPMI Device Information
Type 41	Onboard Devices Extended Information
Type 127	End-of-Table

3.4 Advanced Configuration and Power Interface (ACPI) Overview

The purpose of the ACPI BIOS is to supply the ACPI tables. POST creates the ACPI tables and locates them in extended memory (above 1 MB). The location of these tables is conveyed to the ACPI-aware operating system through a series of tables located throughout memory. The format and location of these tables is documented in the Advanced Configuration and Power Interface Specification, Revision 6.0.

The BIOS supports ACPI 6.0. To prevent conflicts with a non-ACPI-aware operating system, the memory used for the ACPI tables should be marked as “reserved” in INT 15h, function E820h.

As described in the ACPI specifications, an ACPI-aware operating system generates an SMI to request that the system be switched into ACPI mode. The BIOS responds by setting up all system- (chipset) specific configurations required to support ACPI and sets the SCI_EN bit as defined by the ACPI specification. The system automatically returns to legacy mode on hard reset or power-on reset.

There are three runtime components to ACPI:

- **ACPI Tables:**
  These tables describe the interfaces to the hardware. ACPI tables can make use of ACPI Machine Language (AML), the interpretation of which is performed by the operating system. The operating system contains and uses an AML interpreter that executes procedures encoded in AML and is stored in the ACPI tables. AML is a compact, tokenized, abstract machine language. The tables contain information about power management capabilities of the system, APICs, and bus structure. The tables also describe control methods that the operating system uses to change PCI interrupt routing, control legacy devices in the Super I/O, find out the cause of a wake event, and handle PCI hot plug, if applicable.
- **ACPI Registers:**
  The constrained part of the hardware interface, described (at least in location) by the ACPI tables.
- **ACPI BIOS:**
  This is the code that boots the machine and implements interfaces for sleep, wake, and some restart operations. The ACPI Description Tables are also provided by the ACPI BIOS.

The ACPI specification requires the system to support at least one sleep state. The BIOS supports S0 and S5 states.

The S5 state is equivalent to operating system shutdown. No system context is saved when entering S5.

The BIOS should meet all WHCK ACPI 6.0 requirements. Please refer to: Windows Hardware Certification Kit (HCK) and ACPI Logo Certification on: http://msdn.microsoft.com/en-us/library/windows/hardware

3.4.1 ACPI Tables Supported

The BIOS supports the following ACPI Tables. For additional information, refer to Advanced Configuration and Power Interface Specification, Revision 6.0.

Table ID	Description	Signature
MADT	Multiple APIC Description Table	“APIC”
BERT	Boot Error Record Table	“BERT”
DSDT	Differentiated System Description Table	“DSDT”
EINJ	Error Injection Table	“EINJ”
ERST	Error Record Serialization Table	“ERST”
FADT	Fixed ACPI Description Table	”FACP”
FACS	Firmware ACPI Control Structure	“FACS”
MSCT	Maximum System Characteristics Table	“MSCT”
HEST	Hardware Error Source Table	“HEST”
RSDP	Root System Description Pointer	"RSDP"
RSDT	Root System Description Table	“RSDT”
SLIT	System Locality Distance Information Table	“SLIT”
SRAT	System Resource Affinity Table	“SRAT”
SSDT	Secondary System Description Table	“SSDT”
XSDT	Extended System Description Table	“XSDT”
BOOT	Simple Boot Flag Table	“BOOT”
DMAR	DMA Remapping Table	“DMAR”
HPET	IA-PC High Precision Event Timer Table	“HPET”
UEFI	UEFI ACPI Data Table	“UEFI”
SPCR	Serial Port Console Redirection Table	“SPCR”
SPMI	Server Platform Management Interface Table	“SPMI”

MCFG	PCI Express memory mapped configuration space base address	“MCFG”
WDDT	Watchdog Timer Description table	“WDDT”
SLIC	Software Licensing	“SLIC”
PMCT		“PMCT”
FPDT	Firmware Performance Data Table	“FPDT”

3.4.2 System Sleep States

The platform supports the following ACPI system sleep states:

- ACPI S0 (working) state
- ACPI S5 (soft-off) state

3.4.3 Wake Events / SCI Sources

The server board supports the following wake-up sources in the ACPI environment. The operating system controls enabling and disabling these wake sources:

- As required by ACPI specification, the power button can wake the system from all sleep states (S5).

3.4.4 EFI Shell

The EFI shell can be used to execute EFI applications, perform diagnostics or boot to EFI supported operating systems.

The BIOS supports EFI Shell version 2.4. The BIOS also supports Microsoft UEFI requirements for UEFI and secure boot.

As part of the Microsoft requirements for secure boot, UEFI Shell will be disabled when the secure boot is enabled.

4 Processor Support

4.1 Reference Code Integration

Latest validated Intel® Xeon® Scalable Platform CPU reference code drops released by Intel must be integrated into the BIOS.

4.2 Processor Initialization

The BIOS must implement processor initialization flows as specified in Intel® Xeon® Scalable Platform Server BIOS Writer's guide published by Intel.

The BSP is responsible for executing the BIOS POST and preparing the server to boot the operating system. At boot time, the server is in virtual wire mode and the BSP alone is programmed to accept local interrupts (INTR driven by programmable interrupt controller (PIC) and non-maskable interrupt (NMI)).

As a part of the boot process, the BSP wakes each AP. When awakened, an AP programs its memory type range registers (MTRRs) to be identical to those of the BSP. All APs execute a halt instruction with their local interrupts disabled. The system management mode (SMM) handler expects all processors to respond to an SMI.

WCS Intel® Xeon® Scalable Platform is a dual socket system. From the CPU socket population perspective, BIOS must support boot from fully populated (CPU_0 & CPU_1) as well as single socket (CPU_0) configurations.

4.3 CPU Steppings

Typically, the processor configurations are symmetric. In other words, both CPUs are usually of the same stepping level. However, if mixed processor steppings are permitted by Intel, BIOS must support booting the system from such configurations.

4.4 Microcode Update

The BIOS must load microcode into each processor during POST.

4.5 Intel® Hyper-Threading Technology

Hyper threading technology must be supported and enabled via platform BIOS policy variable.

4.6 Enhanced Intel® SpeedStep® Technology

BIOS must detect and implement EIST via BIOS policy variable.

4.7 Direct Cache Access (DCA)

BIOS shall enable DCA by default as per the Intel® Xeon® Scalable Platform System BIOS Writer's Guide.

4.8 Intel® Virtualization Technology

Intel® Virtualization Technology (Intel® VT) must be supported via platform BIOS policy variable.

4.9 Processor Cache

The L1 and L2 caches are enabled by the BIOS during POST. The BIOS should enable all levels of processor cache as early as possible during POST. The L3 is enabled by the processor hardware and cannot be disabled by BIOS. It is recommended that the L1 and L2 caches be left enabled.

All detected cache sizes are reported in the SMBIOS Type 7 structures. If the BIOS implements a cache disable feature, it must also implement a cache enable algorithm.

4.10 Cache Control

It is the responsibility of the BIOS to configure the memory map and cache ability identically on all processors to ensure cache coherency.

4.10.1MTRR Initialization and Cache Enable Requirements

The BIOS must program the Memory Type Range Registers (MTRRs) as well as each logical processor's CR0.CD bit to enable caching.

4.10.2Cache Prefetcher Controls

BIOS shall support Cache Prefetcher Controls via platform BIOS policy.

4.10.3Adjacent Cache Line Prefetcher

BIOS shall support enable/disable of Adjacent Cache Line Prefetcher via platform BIOS policy.

4.11 PROCHOT_RESPONSE Policy

BIOS shall configure PROCHOT response frequency to the maximum efficiency ratio Pn (1200 MHz).

4.12 Built-In Self-Test (BIST)

The BIST_ENABLE can be controlled by a BMC, GPO, strap or other mechanism. BIOS shall implement a platform policy to control BIST execution.

4.13 Disabling Logical Processors

The Intel® Xeon® Scalable processor is implemented with 1 or more cores with each core capable of supporting Intel® HT Technology. The result is multiple logical processors in a physical package. For various reasons, the user may want to disable logical processors in the physical packages (for example, 1. software licenses per logical processor per package and 2. BIST Failures). Akin to previous processor architectures it is possible for BIOS to control the number of enabled logical processors

The table below shows a subset of the possible combinations of active logical processors that can be achieved with BIOS managed logical processor control:

Active Logical Processor Configurations (per package)

Active Cores	Intel® Hyper-Threading Technology	Total Enabled Logical Processors
1	Disabled	1
1	Enabled	2
2	Disabled	2
2	Enabled	4
3	Disabled	3
3	Enabled	6
4	Disabled	4
4	Enabled	8
5	Disabled	5
5	Enabled	10
6	Disabled	6
6	Enabled	12
7	Disabled	7
7	Enabled	14
8	Disabled	8
8	Enabled	16

Note: At least one core needs to be left active in each processor package.

4.14 Processor Power Management Supported

The BIOS shall support following processor controls:

- Processor power states (C-state)
- Processor clock throttling (T-state)
- Processor performance states (P-state)

C0 is an active power state where the CPU executes instructions, and other C states are processor sleeping states where the processor consumes less power and dissipates less heat than leaving the processor in the C0 state. The deeper C states supported includes

- C1, C1E, C3, C6

Clock throttling is a technique which can stop granting cycles to emulate a divided processor clock frequency by controlling the clock logic.

BIOS support 8 stages of T-State.

Since P-state saves power by restricting the CPU core frequency. The deeper P states the OS selects the more power the platform saves.

- BIOS support the following P states:
  P1, P2, P3 P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15.

4.15 Enhanced Intel SpeedStep® Technology

The BIOS shall support EIST via a platform BIOS policy variable.

4.16 Hardware Power Management (HWPM)

The BIOS shall support four HWPM modes via platform BIOS policy, viz. Disable, Native Mode, Native Mode with Legacy Support, and Out of Band Mode

- .

5 Memory Support

5.1 Memory Initialization - Cold Boot Flow

This section describes the high level cold-boot flow of the IMC initialization. These are the basic steps that the Memory Reference Code does to initialize the memory subsystem. For specific register functionality and bit definition, please refer to the appropriate processor EDS published by Intel.

1. Initialize Uncore, Platform Controller Hub (PCH), Platform specific configuration
2. Detect Reset State (cold boot, warm reset) - input to MRC
3. Get CPU data - input to MRC
4. Early Initialize Throttling
5. Detect DIMM presence and initial configuration
6. Disable TSOD polling
7. Initialize SMBus controllers
8. Read pre-selected SPD data for each DIMM
9. Read SPD device (type 0x0B) to detect DIMM presence
10. Keep track of highest common DIMM frequency per socket
11. Read min TCK, # of banks, # of ranks, raw card ID, Vdd support.
12. LVDDR4 detection, voltage adjustment for LVDIMM
13. Qualify memory configuration (check population rules, take appropriate actions)
14. Disable unpopulated channels
15. Map out DIMMs that do not confirm with DIMM population rules
16. Check DIMM geometry against JEDEC specification
17. Map out DIMMs that are not POR
18. Check DIMM rank structure
19. Initialize rank structures based on requested RAS mode
20. Evaluate requested RAS mode to ensure supported configuration
21. Configure the DDR frequency
22. Determine the highest common frequency across all sockets
23. Request a system reset if adjusting frequency is necessary
24. Configure the desired voltage level
25. Set Vdd voltage via processor or platform VRs (DDR4/DDR4L)
26. Gather detailed SPD data (DRAM timing info etc)
27. Initialize DIMM memory technology (Global early configuration)
28. Enable ECC if required
29. Channel Early Config
30. Verify if selected CAS latency is supported by each DIMM on each channel
31. Program memory timings for each channel
32. Pre DDR Training

33. Disable scrambling
34. Program ODT timing parameters
35. Set platform Vref
36. Start JEDEC initialization sequence
37. Perform DDR training
38. Post DDR Training
39. Perform DRAM memory test
40. Channel Late Configuration
41. Enable ECC if requested
42. Enable DDR scrambling
43. Perform DRAM ECC initialization
44. Finalize throttling initialization
45. Switch to normal mode
46. Initialize memory map
47. Set up RAS mode, Patrol/Demand scrubbing
48. Construct BDAT data structure if feature is enabled
49. Report DIMM information if Serial Debug Console is enabled
50. Exit

5.2 Memory Test and ECC init Using Hardware Engine

Intel® Xeon® Scalable processor has a built-in hardware memory test and init engine. Use of this engine for memory testing is strongly recommended since it provides significantly better test coverage per time ratio as compared to any software based memory tests. The Memtest engine tests all memory locations using pseudo-random data patterns.

The test engine is ECC-agnostic and can flag any data mismatches and identify the failed DIMM, but it is not capable of differentiating between single bit errors and a multi bit errors. Additional software tests can be performed if such differentiation is desired. At the end of the memory test, the ECC will not be corrected. Hence, the memory test should be followed by memory init. The init engine initializes all memory locations to validate ECC. Intel® Xeon® Scalable Platforms support a feature called “data scrambling”. This feature improves detection of DDR address bit errors and may potentially reduce power consumption. If data scrambling is desired, the BIOS should enable this feature before ECC initialization.

Memory test and init may be executed on multiple channels simultaneously. This is expected to further speed up memory test during system boot. The degree of parallelization can be limited by amount of power delivery, which is a function of the system design.

5.3 System Memory layout

The BIOS provides the total amount of memory in the system by supporting the INT 15h, E820h function. For additional information, refer to Advanced Configuration and Power Interface (ACPI) Specification, Revision 2.0 for details.

5.3.1 Memory Reservation for Memory-mapped Functions

A region of size 0.25 GB of memory below 4 GB is always reserved for mapping chipset, processor and BIOS (flash) spaces as memory-mapped I/O regions. This region will appear as a loss of memory to the operating system. In addition to this loss, the BIOS creates another reserved region for memory-mapped PCI Express* functions, including a standard 0.25 GB of standard PC Express configuration space. This memory is reclaimed by the operating system if PAE is turned on in the operating system

5.3.2 High-Memory Reclaim

When 4 GB or more of physical memory is installed (physical memory is the memory installed as DIMMs), the reserved memory is lost. However, the chipset provides a feature called High memory reclaim, which allows the BIOS and the operating system to remap the lost physical memory into system memory above 4 GB (the system memory is the memory that can be seen by the processor).

The BIOS will always enable high-memory reclaim if it discovers installed physical memory equal to or greater than 4 GB. For the operating system, the reclaimed memory is recoverable only when it supports and enables the PAE feature in the processor. Most operating systems support this feature. For details, see the relevant operating system manuals.

5.4 Memory Thermal Throttling

Thermal Throttling is a mechanism employed by the memory controller to reduce the bandwidth of memory traffic to lower the thermal temperature of the DIMMs to protect them from overheating. BIOS shall support two possible thermal throttling mechanisms, viz. Closed Loop Thermal Throttling (CLTT) and Open Loop Thermal Throttling (OLTT). The specific throttling mechanism enabled shall be controlled via a platform BIOS policy option.

5.5 Memory Power and Thermal Management

Each mode in which the Integrated Memory Controller (iMC) module will reduce performance for power savings will be at the command of the Uncore power manager (PCU). The Uncore power manager will be aware of collective CPU power states, Platform power states, and isochronous requirements. It will request entry into a particular mode and the iMC module will acknowledge entry. The DDR4 power states can be summarized as the following:

- Normal operation (highest power consumption)
    - CKE Power-Down Active Power Down

  - Precharge Power Down with Fast Exit (DLL-ON)
  - Precharge Power Down with Slow Exit (DLL-OFF)
- Self-refresh
  - IO-MDLL off
  - PLL-off
- Memory Power Saving (Optional)

5.5.1 Self Refresh

The Uncore power manager may request the iMC module to place the DRAMs in Self Refresh (SR) State. Self Refresh will not be a per channel state, all channels will either be in Self Refresh or active. The possible changes in configuration registers are:

- SREF_enable timer
- Exit conditions

5.5.2 Memory Power Saving (Optional)

The idea of memory power saving is to manually set the memory frequency in the BIOS setup menu. There are three options: Auto, 1066MHz, and 1333MHz which could be set. If a lower frequency is set, the system will consume less power automatically. This feature must go through reset mechanism to activate.

6 Integrated IO (IIO) Support

6.1 Intel® Xeon® Scalable Processor IIO

Intel® Xeon® Scalable Processor IIO supports:

- PCI Express* Interface: Supports Gen1 (2.5Gb/s), Gen2 (5Gb/s) and Gen3 (8Gb/s) speed. When the DMI3 port is operating as PCIe, it will operate as Gen3 x4 port-0. Refer to RS - Grantley Platform Design Guide (PDG) for how to strap DMI2 lanes to operate as PCIe.
- DMI3 Interface to the PCH: Chip-to-chip connection between the Intel® C620 series chipset (PCH).
- Integrated IOAPICs: used to convert legacy interrupts from IO devices into messages to the CPU's Local APIC. Intel® Xeon® Scalable processor supports a unique IOAPIC per PCIe port. Unlike previous platforms where only one IOAPIC is required per CPU, Intel® Xeon® Scalable Platform requires more than one IOAPIC per CPU depending on the number of supported PCIe ports. Intel RC introduces the name called IIO stacks where each stack has its corresponding PCIe port and a unique IOAPIC. Please refer to Intel RC for more info about IIO stacks.
- Intel® UPI Technology: used for efficient, high bandwidth data movement interrupt between two locations in memory or from memory to IO.
- I/O Virtualization Logic (VT-d2).

6.1.1 PCI Express General Purpose Ports

Intel® Xeon® Scalable Processor IIO module supports 48 lanes of PCIe support that can be configured as up to 12 independent PCIe ports:

- Port-0 is either DMI3 (legacy CPU) or PCIe Gen3 x4 Port (if not connected to PCH for 52 lanes of PCIe support).
- IOU0 is synonymous with Port-2, 1 x16 PCIe Gen3 port but can be bifurcated down to 2 x8 or 4 x4.
- IOU1 is synonymous with Port-3, 1 x16 PCIe Gen3 port but can be bifurcated down to 2 x8 or 4 x4.
- IOU2 is synonymous with Port-1, 1 x16 PCIe Gen3 port but can be bifurcated down to 2 x8 or 4 x4.

6.1.2 PCIe Slots and Bifurcation

There are five PCIe slots configured by the BIOS. The table below shows the width and possible bifurcations for the five slots.

Slots	CPU	Port#	Width	Bifurcation

1	0	Port 3A	X8	1x8 or 2x4
2	0	Port 3C	X8	1x8 or 2x4
3	0	Port 1A	X16	1x16, 2x8, or 4x4
4	1	Port 1A	X16	1x16, 2x8, or 4x4
5	1	Port 3A	X16	1x16, 2x8, or 4x4

6.1.3 PCIe Non-Slot Bifurcation

There are nine PCIe non-slot bifurcations required for WCS Intel® Xeon® Scalable Platform to support below key features:

- Support for non-standard PCIe x24 (using 1 standard x16 slots + Oculink x8 cable)
- Support for 4 onboard M.2 Modules
- Support for 3 x4 PCI expansion
- Support for x16 bandwidth QAT

Component	Port#	Width	Bifurcation
CPU 0	Port 2A	X16	1x16
CPU 1	Port 2A	X8	1x8
CPU 1	Port 2C	X4	1x4
CPU 1	Port 2D	X4	1x4
CPU 1	DMI as x4 PCIe	X4	1x4
PCH	Port 0	X4	1x4
PCH	Port 4	X4	1x4
PCH	Port 12	X4	1x4
PCH	Port 16	X4	1x4

6.1.4 PCI Express Port Initialization Algorithm

The BIOS should be aware of the number of logical PCIe ports present on the system. Of the set of ports present, to determine which ports are operational, BIOS must read and interpret how the PCIe interface is strapped. Handshake signaling during hardware training will detect lane designations to determine whether attached devices are communicative, and the link width in which they trained. The steps required to initialize the PCIe ports are

- For each IOUx (x=1/2/3) grouping, program bits[2:0] (bits[1:0] for IOU2) in the PCIE_IOUx_BIF_CTRL (Dev 1/2/3:Func 0:Offset_190h Word) register to the desired configuration.
- Set bit 3 in the same PCIE_IOUx_BIF_CTRL register to initiate the port training.
- Check the LNKSTS (Offset A2h) register bit[13] to see if training has completed. If it has not completed within 50ms of initiating training, then set LNKCON (OffsetA0h) register bit[4] to disable the link.

- Determine the widths of any links that have successfully trained. Read LNKSTS register bits[9:4] to determine the widths.
- Where the platform connects a slot to the port, set SLTCAP (Offset A4h) register bits[31:19] with a unique slot number.
- Repeat steps 2-5 for each configured PCI Express port.

By the end of this platform initialization step, BIOS will have a mapping of which ports came up in their full native configuration, which came up in a failed-down operating mode, and which failed completely to initialize. Support will be required in the boot ROM/FLASH and/or in NVRAM to store what a "full configuration" should look like for the platform in order to enable BIOS to distinguish "partial" configurations after link-up and after future hard reset sequences.

Upon completion of this sequence, the IIO PCI Express hierarchies are capable of running a standard PCI enumeration cycle

6.1.5 Scan Order

The BIOS assigns PCI bus numbers in a depth-first hierarchy, in accordance with the PCI Local Bus Specification, Revision 3.0. The bus number is incremented when the BIOS encounters a PCI-PCI bridge device. Scanning continues on the secondary side of the bridge until all subordinate buses are assigned numbers. PCI bus number assignments may vary from boot to boot with varying presence of PCI devices with PCI-PCI bridges. If a device with a bridge with a single bus behind it is inserted into a PCI bus, all subsequent PCI bus numbers below the current bus are increased by one.

The bus assignments occur once, early in the BIOS boot process, and never change during the pre-boot phase.

6.1.6 Resource Assignment

The BIOS resource manager assigns the PIC-mode interrupt for the devices that are accessed by the legacy code. The BIOS will ensure the PCI BAR registers and the command registers for all devices are correctly set up to match the behavior of the legacy BIOS after booting to a legacy operating system. Any legacy code cannot make any assumption about the scan order of devices or the order in which resources are allocated to them. The BIOS supports the INT 1Ah PCI BIOS interface calls.

6.1.6.1 IIO Resource Allocation

Due to the split IIO design implemented in Intel® Xeon® Scalable Processors, resources such as MMCFG, MMIOL, MMIOH, Legacy IO, IOAPIC are assigned individually to each IIO/MCP stack in each socket. WCS platform uses two CPU sockets with no MCP. Therefore, there will be 8 stacks with each IIO having 4 stacks.

By default, all resources are evenly distributed across the enabled stacks. For MMIOL, the evenly distributed values will be too low for add-in cards that need more resources. One case is for the graphics cards. When the prefetchable BAR is forced to below 4GB for legacy boot mode (for some card even for UEFI boot mode) for VGA/VESA using 32 bits address, the out of resource situation will cause the system to reboot in order to adjust the resources. The reboot happens very late in boot phase which results almost double the boot time.

This MMIOL situation can also be an issue for hot plug support. Padding resources need to be allocated in advance for the root ports that need hot plug support. Evenly distributed resources will not meet the resource demand of hot plug support.

Therefore, WCS has added the BIOS functionality to pre-allocate MMIOL to IIO stacks. A setup screen "OEM IIO MMIO Low" has been created. The default values can be pre-allocated for the actual resource demands. For more setup details, see Chapter 15 Configuring Platform BIOS (Aptio Setup).

6.1.7 Automatic IRQ Assignment

The BIOS automatically assigns IRQs to devices in the system. No method is provided to manually configure the IRQs for devices.

6.1.8 Gen1/Gen2/Gen3 Speed Selection

In general, IIO will negotiate Gen1 vs. Gen2 vs. Gen3 speed per the inband mechanism defined in the Gen3 PCI Express Specification. In addition, IIO can be prevented from negotiating Gen3 or Gen2 speed if Gen2 or Gen3 fuse is blown, that is, Gen2 or Gen3 is disabled, none of the ports would ever train to Gen2 or Gen3, even if software attempted it.

6.1.8.1 Gen2 Configuration

When the link is operating at Gen2 speed, BIOS may need to configure the level of de-emphasis for an upstream component by programming the LNKCON2 (Dev 1-10:Func0:Offset C0h) register bit[6].

Note: If a PCIe link does not train up to full width, it is recommended to perform a Secondary Bus Reset on the link to get it full width.

6.1.9 Max_Payload_Size

IIO will support a Max_Payload_Size of 256B.

6.1.10Device and Slot Power Limits

All add-in devices must power-on to a state in which they limit their total power dissipation to a default maximum according to their form-factor (10W for add-in edge connected cards). When BIOS updates the slot power limit register of the root ports within the IIO, the IIO will automatically transmits a Set_Slot_Power_Limit message with corresponding information to the attached device. It is the responsibility of platform BIOS to properly configure the slot power limit registers in the IIO. Failure to do so may result in attached endpoints remaining completely disabled in order to comply with the default power limitations associated with their form-factors.

6.1.11ASPM Control

The BIOS should provide a user option to enable disable ASPM.

6.1.12Direct Media Interface 2(DMI2)

The Direct Media Interface 2 (DMI2) is the connection between the processor and Intel® C620 series chipset (PCH). The DMI is an extension of the standard PCI Express specification with special commands/features.

6.1.12.1 Direct Media Interface Configuration

The configuration registers for the DMI port reside in an Root Complex Register Block (RCRB) and adhere to the capability structures formats outlined for internal links as defined by PCIe* specification. DMIRCBAR defined the DMI RCRB Base Address. BIOS must program the DMI interface DMIRCBAR and must allocates on a 4KB boundary. System BIOS must also enable DMIRCBAR. The address ranges must be reserved and reported using ACPI_CSR objects.

6.1.12.2 Virtual Channel

VCO, VC1, VCm, and VCp are supported. VC0 traffic is handled as snoop/non-snoop based on NS bit in DMI packet. Use of VC1 for true isochronous. VCm for Intel® Managability Engine (Intel® ME). Private Virtual Channel VCp for legacy isochronous traffic.

Virtual Channel configuration must be configured on both sides of the link and must match in terms of the number of VCs, VC ID, and TC/VC mapping. This document covers only the SA side of the DMI port. Please refer to PCH BIOS Specification for details related to configuration on the PCI side of the link.

The virtual channel 1 configuration for the DMI port must be programmed by the BIOS to ensure that the PCH is enabled for Isoch/VC1 operation. Please refer to PCH BIOS Specification for details on when VC1 is enabled on the PCH.

6.1.15.3 PCI-e Tuning for Gen3

The BIOS has to be ported to support PCI-e tuning (CTLE values) for PCI-e slots on the WCS platform. The BIOS need to be validated with all WCS supported PCI Gen3 cards (ex: FPGA and M2 PCIe cards).

7 PCH Support

PCH configuration is platform-specific, including GPIO initialization, PCIe root ports, ME, IOAPIC, etc. BIOS follows intel guidelines to program and configure PCH devices for Olympus hardware platform.

7.1 Reference Code

Intel® Xeon® Scalable Platform BIOS must have complete support for Intel® C620 series chipset (PCH) reference code provided by Intel. This should also include PCH policy management provided by Intel.

7.2 PCH Configuration

Flexible high speed I/O lanes may be configured for different interface: SATA, sSATA, USB3, 1GbE, and PCIe Root Ports. Based on the description of Olympus hardware block, the following flexible I/O ports should be configured as below to meet the platform needs.

- Flexible I/O ports 1-3 should be configured for USB3.
- Flexible I/O ports 19-26, listed on the hardware schematics as PCIE_RP[12-15], PCIE_RP[16-19], are wired to two MiniSAS-HD x4 connectors. They can be configured dynamically either as two PCIe Gen3 2X4 expansion from PCH root ports or as SATA expansion ports. See the next section for more details.
- Flexible I/O ports 15-18 should be configured for sSATA ports 2-5.
- Flexible I/O ports 7-10 are configured as PCIe Root Port 0 for Gen3 x4 M.2 module 1 and flexible I/O ports 11-14 as PCIe Root Port 4 for Gen3 x4 M.2 module 2.

Note that QAT and 10 GbE are internal devices, separate from flexible I/O ports. BIOS should configure QAT as a 1x16 PCIe device and enable 10 GbE device.

7.2.1 PCIe/SATA Expansion

As listed above, the two MiniSAS-HD x4 connectors can be configured dynamically either as two PCIe Gen3 2X4 expansion from PCH root ports or as SATA expansion ports.

Eight PCH GPIO pins are assigned to be dedicated to do the hardware control to configure the eight flexible high speed I/O lanes either as PCIe or SATA. BIOS uses ME tool spsFITc to program the settings of the soft strappings for the eight flexible high speed I/O lanes as “Assigned based on GPIO pin polarity”. At the BIOS build time, the programmed soft strappings are integrated into the descriptor region of ME in the BIOS SPI flash image. At an AC cycle, the soft strappings are loaded by the suspend power rail before the point of time when reset vector gets loaded. Depending on the polarity of the eight PCH GPIO pins, the eight flexible high speed I/O lanes will be configured accordingly.

On BMC side, there are two BMC GPIO pins that each control four of the eight PCH GPIO pins. The goal is to be able to configure on the granularity of four lanes. Therefore, if needed, four lanes can be configured as PCIe while the other four lanes can be configured as SATA, though there will less likely be such a SKU.

There are no jumpers on the board for this functionality.

For history, other approaches were considered other than using eight GPIO pins as above because there were short of GPIO pins at the beginning. One was to change the soft strapping in BIOS code or to use the same mechanism as the flash update tool. The other one was to use two BIOS binaries, one for each of PCIe and SATA. The first approach was not compliant with Intel PCH security guide line. The other approach was not good because BIOS has the policy of using one BIOS binary that is provisioned dynamically for different EGs.

7.3 PCH SATA

The BIOS initializes the embedded SATA controllers in the chipset and any SATA devices that are connected to these controllers. From a software standpoint, SATA controllers present the same register interface as the PATA controllers. Hot plugging of SATA drives during the boot process is not supported by the BIOS and may result in undefined behavior.

7.3.1.1 Compatible Mode

A controller that operates in compatible mode emulates a legacy IDE controller, which is a nonstandard extension of the ISA-based IDE controller. In compatible mode, the controller requires two ISA-style dedicated IRQs (14 and 15) that cannot be shared with other devices. Because compatible mode requires dedicated resources, the ATA controller for the boot device (which is usually integrated in chipsets on the motherboard) is the only controller on a system that is likely to operate in compatible mode.

7.3.1.2 Native Mode (a.k.a Enhanced Mode)

A controller that operates in native mode acts as a true PCI device that does not require dedicated legacy resources and can be configured anywhere in the system. ATA controllers running in native mode use their PCI interrupt for both channels and can share this interrupt with other devices in the system, like any other PCI device. Add-in ATA controllers generally operate in native mode

7.3.1.3 AHCI Mode

AHCI gives software developers and hardware designers a standard method for detecting, configuring, and programming SATA/AHCI adapters. AHCI is separate from the SATA standard, although it exposes SATA's advanced capabilities (such as native command queuing). AHCI should be the default mode for WCS.

7.3.2 Handling of Hard Disk Passwords during Pre-Boot

The data stored on disks can be optionally protected via passwords at a hard disk level. The challenge mechanism in such storage devices is implemented within the firmware on the hard disk. This feature necessitates user intervention during pre-boot, which is not desired on WCS Servers. During the discovery phase of hard disks, it is mandatory that BIOS skips the enumeration of those devices that are password protected and thus implement a seamless boot experience.

7.4 PCH PCI Express

There are no PCI Express slots from Intel® C620 series chipset (PCH).

7.5 PCH USB

7.5.1 Removable Media Drives

The BIOS supports booting from USB mass storage devices connected to the chassis USB port, such as a USB flash drive device. The BIOS supports USB 2.0 media storage devices that are backward compatible to the USB 1.1 specification.

7.5.2 USB Initialization

During the power on self-test (POST), the BIOS initializes and configures the USB subsystem.

The BIOS should be capable of initializing and using the following types of USB devices:

- USB Specification-compliant keyboard and mouse.
- USB Specification-compliant mice
- USB Specification-compliant storage devices that utilize bulk-only transport mechanism
    - Bootable USB flash drive
    - Bootable USB hard drive
    - Bootable USB optical drive

USB devices are scanned to determine if they are required for booting.

The BIOS supports USB 1.1 and USB 2.0 compliant devices and host controllers.

During the pre-boot phase, the BIOS should support the hot addition and hot removal of USB devices. For example, if a USB device is hot plugged, the BIOS should detect the device insertion, initializes the device, and makes it available to the user. Onboard USB controllers should be initialized by BIOS. This should not prevent the operating system from supporting available USB controllers, including add-in cards.

7.5.3 I/O Port 60/64 USB Support

The BIOS should support PS/2 emulation of USB keyboard and mouse via I/O Port 60/64 by generating the IRQ1 / IRQ12. Emulation has to handle keyboard data, keyboard commands, mouse data, mouse commands and KBC commands.

7.5.3.1 Emulation disabled without KBC

The USB Keyboard data will be placed in the BDA keyboard buffer. The application accesses to the I/O port 60/64 to get data will fail and accesses to data from the IRQ handler will also fail. Mouse data will not be used at all. All the commands sent to the KBC and devices will fail.

- Keyboard Data
    - USB Int9 (converts into PS2 format ASCII/SCAN code and place it BDA KBD buffer).
- Mouse Data
    - Not used.
- Keyboard & Mouse Command to KBC (I/O Port 60/64)
    - Not handled.

7.5.3.2 Emulation Enabled without KBC

- Keyboard Data
    - USB Int9 (converts into PS2 format ASCII/SCAN code and place it BDA keyboard buffer).
    - USB scanner (converts into IBM standard SCAN code).
    - Emulation driver sends via I/O Port 60/64 by Generating the IRQ1.
- Mouse Data
    - Emulation driver sends via I/O Port 60/64 by Generating the IRQ12
- Keyboard & Mouse Command to KBC (I/O Port 60/64)
    - Emulation Driver sends to PS2 and USB

7.6 PCH Platform Thermal Features

See the later chapter of “Manageability” in the section of “PCH thermal Management”

7.7 PCH SPI Flash

There are two 32 MB flash parts for BIOS. See the hardware specifications for details.

8 M.2 Storage Support

The BIOS must support M.2 storage devices, viz. M.2 NVMe and M.2 AHCI. This should include detection of M.2 storage devices during pre-boot. BIOS must also support booting from M.2 devices both in compatibility mode (Legacy) as well as native (UEFI) mode. The M.2 devices must be detected automatically as PCIe x4 devices.

8.1 BIOS Requirements

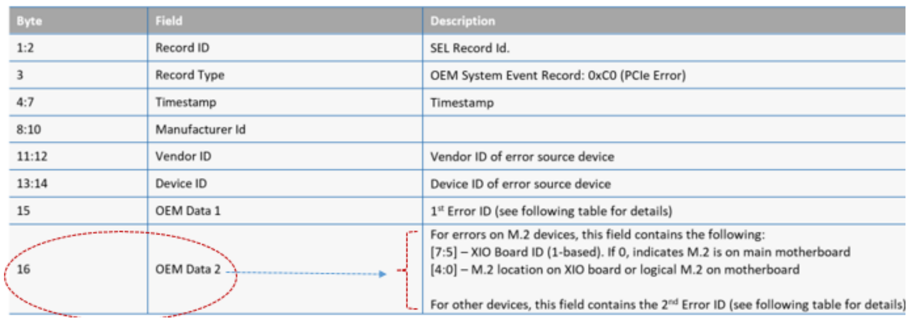
To support this new approach, BIOS must implement necessary changes to ensure enough MMIO resources being allocated for all M.2 devices, to define a mechanism for optionally selecting a specific M.2 device as bootable, and to define a mechanism for visually identifying a specific M.2 device location reported in SEL or OS.

8.1.1 MMIO Resource Allocation

M.2 device supports non-prefetchable 64-bit MMIO address. However, Mt. Olympus platform supports only 64-bit MMIO prefetchable address. To ensure enough MMIO resources being allocated for all M.2 devices and any future increase in number of M.2 devices, BIOS should enable 64-bit MMIO setup option as default. This default option allows any prefetchable 64-bit MMIO capable devices being allocated with 64-bit MMIO resources and provides more 32-bit MMIO resources available for M.2 devices.

8.1.2 Mechanism to Identify M.2 Onboard Device

Mt. Olympus may support any onboard devices, which directly connect to any Root Ports from PCU or CPU. The platform configuration support[TBD] for onboard devices defines the M.2 disk logical location diagram, which provides a map of how the disk logical location number for each onboard device should be configured in bits [31:19] of its corresponding Root Port's slot capabilities register. Once the M.2 disk logical location diagram is defined, BIOS can read the disk logical location number from the Root Port's slot capabilities register when reporting any uncorrectable error generated by a specific M.2 onboard device in byte 16 of PCIe extended SEL as formatted below. Note that bits [7:5] of byte 16 must be zero and bits [4:0] of byte 16 holds the disk logical location number for M.2 onboard device.



Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC0 (PCIe Error)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	
11:12	Vendor ID	Vendor ID of error source device
13:14	Device ID	Device ID of error source device
15	OEM Data 1	1st Error ID (see following table for details)
16	OEM Data 2	For errors on M.2 devices, this field contains the following: [7:5] – XIO Board ID (1-based). If 0, indicates M.2 is on main motherboard [4:0] – M.2 location on XIO board or logical M.2 on motherboard For other devices, this field contains the 2nd Error ID (see following table for details)

8.1.3 Mechanism to Select Bootable Device

To support selection of a specific bootable M.2 device, BIOS can use the existing Boot Option Configuration to specify the M.2 disk logical location, as mentioned in section 8.1.2. The format of defining which specific bootable onboard M.2 device is described in Data 2 and Data 5 as below:

- Data 2 [5:2] should be 0010b (Force Boot from Default HD)
- Data 5 [7:5] as RSVD
- Data 5 [4:0] should be M.2 disk logical location

9 Manageability

9.1 Intel® Server Platform Services (Intel® SPS)

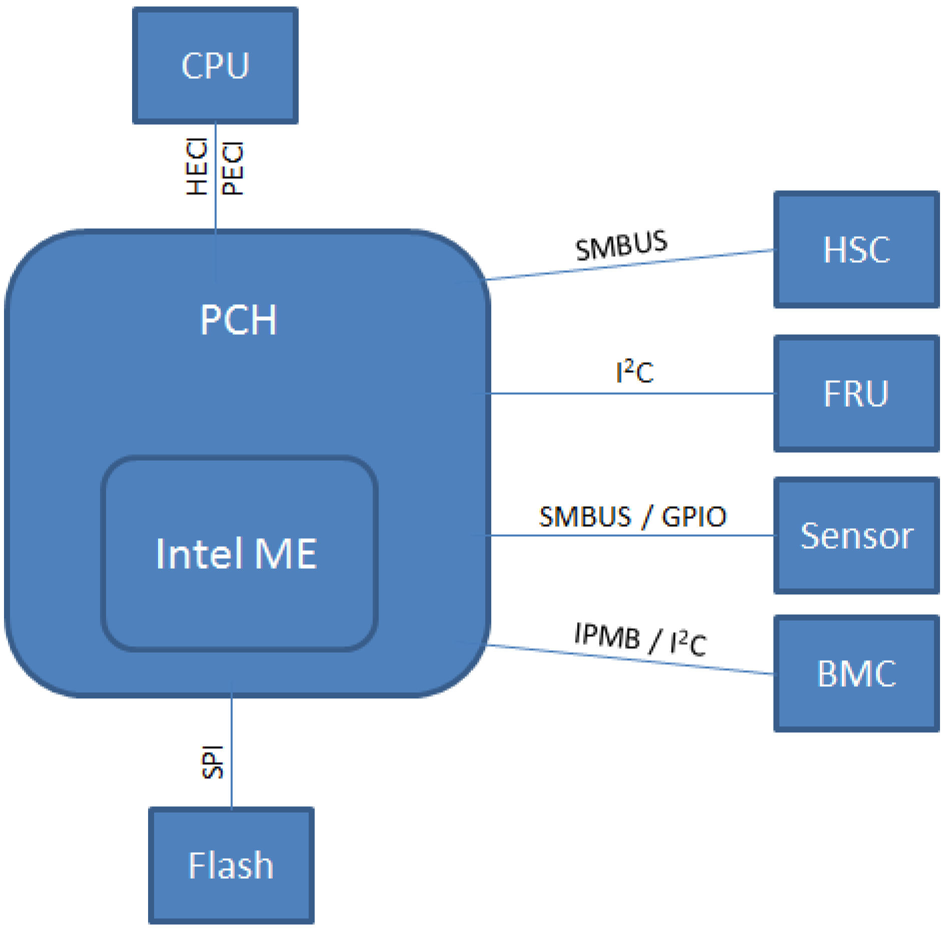
Intel® SPS is the name for the firmware stack running on the PCH in Intel® Xeon® Scalable Platforms. The Intel® Managability Engine (Intel® ME) and Intel® SPS FW are mandatory component on Intel® Xeon® Scalable Platforms.

Intel® SPS Functions in one of the following modes:

- Silicon Enabling (SiEn):
  Basic hardware configuration functionality supports HECI-1 and IPMI interfaces. Silicon enabling is the base configuration.
- Node Manager (NM):
  Implements Node Manager functionally with an IPMI interface
- Datacenter and Node Manager (DNM):
  Implements DCMI and Node Manager functionally with an IPMI interface

WCS platforms utilize the Node Manager (NM) mode.

The figure below shows the interfaces that are used by ME FW to communicate with other FW/SW components:



Interfaces

HECI interface

- Used for BIOS (HECI-1) and OSPM (HECI-2) interaction

SPI interface

- SPI flash access

SMBus interface

- BMC (IPMI bridging), HSC, Sensors

The Intel® ME interfaces with multiple FW/SW components in the system to perform its functionality; the following is a description of some of these interfaces:

- HECI-1 interface: This interface shows up as a host PCI device.
- ME Interacts with the BIOS during POST. ME FW uses DCMI-HI over HECI 1 as its channel for DM in-band communications.
- HECI 2 supports run-time (SMI/SCI) communication with the ME.
- PECI interface is used for sending PECI commands to the CPU (read/write MSRs)
- SPI the ME communicates with Flash using the SPI interface.
- SM Line 1, can be used for board sensor (LM75) support. The ME is the master on the bus.

- SMBUS (STM1), Intel® ME is the master on the bus the default address should be 0x48. Bus provides sensor access.
- ME has GPIO read support.
- FRU: Intel® SPS firmware provides an internal FRU stored in the ME SPI flash. This is accessible via SMLink 1 using the Master Read-Write IPMI command.

9.1.1 BIOS Communications

HECI provides a mechanism for BIOS to communicate with the Intel® ME. HECI is bi-directional asynchronous interface for passing messages between the BIOS and ME FW.

In addition, HECI also provides a mechanism for BIOS to communicate with the Innovation Engine, which is an in-situ PCH feature that replaces BMC. For the Intel® Xeon® Scalable Platform BIOS, the BIOS should support IPMI messaging/communication over HECI in order to implement all BIOS-IE interactions that are functionally equivalent to BIOS-BMC interactions.

9.1.2 IPMI System Interface

The BIOS should enable the System Interface to the BMC in early POST for the logging of POST errors and events.
Due to AST1250 needing 3-5 seconds to decompress its firmware from flash memory and boot to embedded OS when AC on, BIOS power on may have to wait up to 60 seconds before it can send the first command to BMC.

9.1.3 BMC to SMS interfaces.

The KCS interface is specified solely for SMS messages. SMM messages between the BMC and an SMI Handler will typically require a separate interface, though the KCS interface is designed so that system software can detect if a transaction was interrupted. The KCS Interface is designed to support polled operation. For further information and implementation guidance: Refer to the IPMI 2.0 specification section 9 Keyboard Controller Style (KCS) Interface.

9.1.4 Block Transfer (BT) Interface

The BT interface is a supported BMC to SMS system interface. The BT Interface buffers blocks of message data is before the management controller of available data. This is different from the SMIC and KCS interfaces, which are byte-transfer oriented.

The host side of the BT Interfaces designed for interrupt or polled operation. Implementations can elect to provide a system interrupt from the assertion of the B2H_ATN or SMS_ATN (BMC -to-Host attention or System Management Software attention) states. Note that implementing an interrupt must not preclude driver software from the using the interface in a polled manner.

The Host interface to the baseboard management controller (BMC) requires a block of 3 contiguous I/O locations on the system board. (A reference implementation fixes this at locations E4h:E6h. The interface circuitry will decode the lower 2 address lines, SA[1..0]). A general -purpose chip select will be used to generate the select line for the interface, which is to reside in system I/O space. The I/O address offsets are defined as follows

Offset	Read	Write
0	BT_CTRL – Control Register	
1	BMC2HOST buffer	HOST2BMC buffer
2	BT_INTMASK – interrupt mask register	

The BT interface provides support for implementations that allow the submission and asynchronous completion of commands.

Request Messages are sent to the BMC from system software using a write transfer through the BT Interface. The message bytes are organized according to the following format specification:

BT Interface/BMC Request Message Format

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5:N
Length	NetFn/Lun	Seq	Cmd	Data

Reference IPMI Specification, Version 2.0, Section 11.1 BT Interface-BMC Request Message Format.

BMC-BT Interface Response Message Format

Response Messages are read transfers from the BMC to system software via the BT Interface. Note that with a few exceptions (e.g., Cold Reset command) the BMC always returns response to a request delivered via the BT interface in order to deliver the completion code, regardless of whether the response has data in the Data field. The message bytes are organized according to the following format specification:

BT Interface/BMC Response Message Forma

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6:N
Length	NetFn/Lun	Seq	Cmd	Completion Code	Data

Reference IPMI Specification, Version 2.0, Section 11.2 BMC-BT Interface Response Message Format.

9.1.5 Serial Port Map

The AST1250 provides three sets of virtual UART controllers, which comply with 16550. Each UART controller has in-built 16x8 transmit FIFO buffer and 16x8 receive FIFO buffer, both can be enabled or disabled. Control signals:

Signal	IO	Type	Description

NCTS	Input	CMOS	clear to send modem status
NDCS	Input	CMOS	data carrier detect modem status
NDSR	Input	CMOS	data set ready modem status
NRI	Input	CMOS	ring indicator modem status
NDTR	Output	CMOS	data terminate ready modem status
NRTS	Output	CMOS	request to send modem status
TXD (Must)	Output	CMOS	transmit serial data output
RXD(Must)	Input	CMOS	receive serial data input

Intel® Node Manager Hardware Requirements:

9.2 Node Manager BIOS Requirements

The BIOS modifications for the Intel® Server Platform Services Intel® ME firmware will contain:

Common Intel® ME features:

- Intel® ME firmware feature query
- MCTP over PCIe
- HECI-1 initialization
- Disabling Global Platform Reset capabilities
- Integrated Clock Controller (ICC) programming
- Clearing ME_WAKE_STS in PRSTS

Power limiting functionality for Intel® NM:

- Initialization message with CPU configuration
- HECI-2 initialization
- BIOS runtime requirements:
    - ASL code to handle P-state/T-state limit changes
    - ASL code to handle dynamic CPU core idling requests
    - ASL code to provide Power Meter functionality

The Intel® ME will only receive messages via HECI-1 from the single System Boot Strap Processor running the system BIOS code

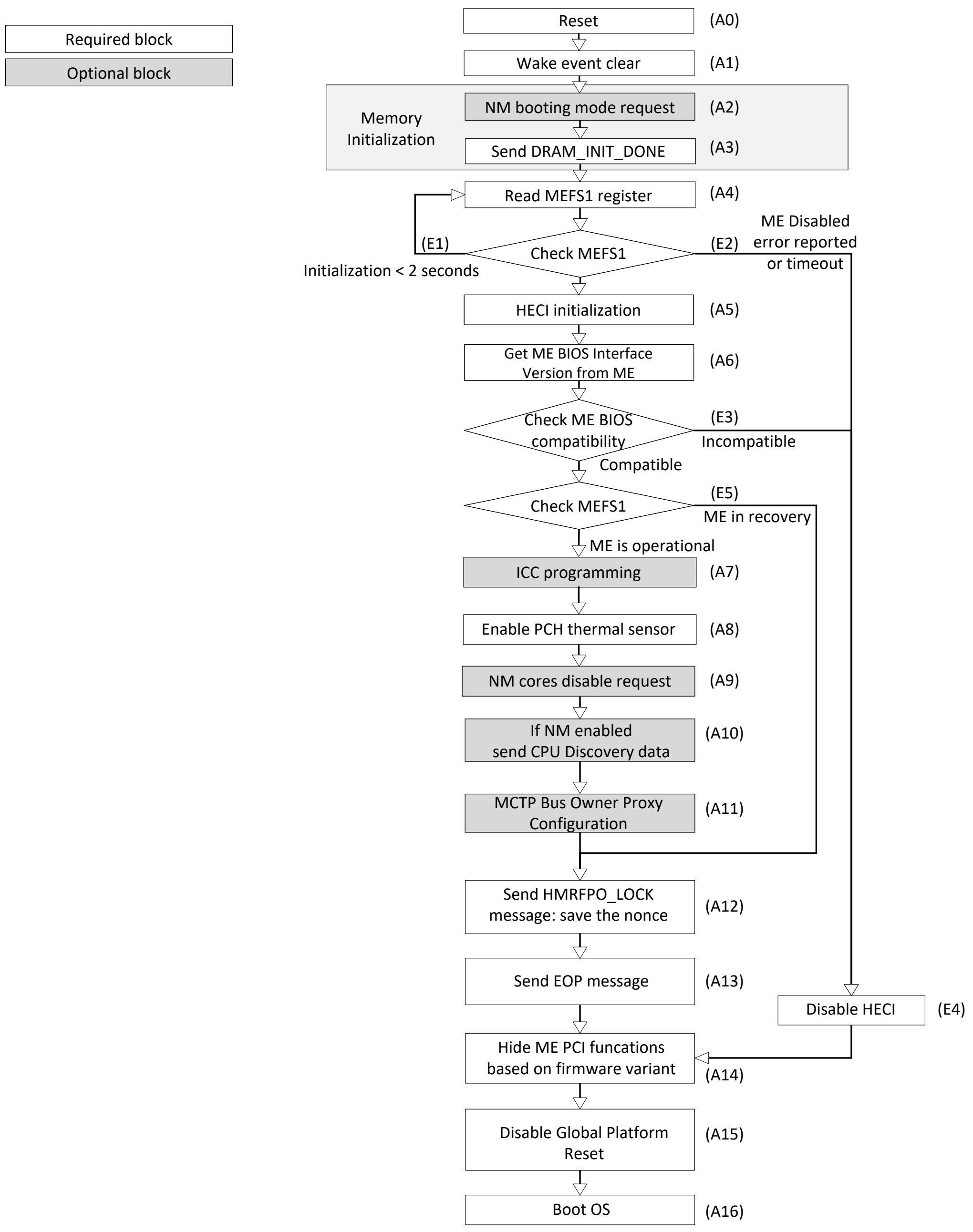
For the Node Manager Fast NM Limiting, the following settings shall be used:

Parameter	**Description**
Fast NM Limiting enabled	true – fast limiting enabled
Fast ramp down step	Ramp generation step in 1/8 Watt per CPU every 10 ms

Fast ramp up step	Ramp generation step in 1/8 Watt per CPU every 10 ms
Fast ramp delay	Delay before starting ramp generation as a multiplier of the pooling interval
Polling interval	Polling interval: 1=>10ms

9.2.1 BIOS POST Requirements

The BIOS POST modifications for the Intel® Server Platform Services Intel® ME firmware are described for all firmware variants:


Required block
Optional block
Reset (A0)
Wake event clear (A1)
Memory Initialization
NM booting mode request (A2)
Send DRAM_INIT_DONE (A3)
Read MEFS1 register (A4)
(E1) Check MEFS1 (E2)
Initialization < 2 seconds
ME Disabled error reported or timeout
HECI initialization (A5)
Get ME BIOS Interface Version from ME (A6)
Check ME BIOS compatibility (E3)
Incompatible
Compatible
Check MEFS1 (E5)
ME in recovery
ME is operational
ICC programming (A7)
Enable PCH thermal sensor (A8)
NM cores disable request (A9)
If NM enabled send CPU Discovery data (A10)
MCTP Bus Owner Proxy Configuration (A11)
Send HMRFPO_LOCK message: save the nonce (A12)
Send EOP message (A13)
Disable HECI (E4)
Hide ME PCI funcations based on firmware variant (A14)
Disable Global Platform Reset (A15)
Boot OS (A16)

BIOS POST Requirements

(A0) At reset BIOS receives control at the architectural reset vector

(A1) BIOS must initialize PCH power management by clearing the PCH wake reason in ME_WAKE_STS in the PRSTS configuration register see [EDS]. This bit must be cleared to allow platform shutdown.

(A2) BIOS reads the mode of booting from the Intel® NM Firmware Status register in HECI-2 interface and sets proper BIOS booting mode. Two modes are supported Performance and Power Optimized.

(A3) After BIOS configures UPI, memory and MCTP it sends the DRAM_INIT_DONE message to Intel® ME.

(A4) BIOS reads Intel® ME Firmware Status #1 register in HECI-1 interface to learn the state of the firmware running in Intel® ME

(E1) If the register indicates that Intel® ME firmware is initializing, BIOS returns to step (A4). The timeout to exit this loop is 2 seconds.

(E2) If the register indicates that Intel® ME firmware is in one of the states:

Hard Intel® ME –Disabled

Nonzero Error Code is reported

or if the 2 seconds timeout elapses BIOS continues boot procedure without communication with Intel® ME. At the end BIOS needs to jump to step (E5) and disable Intel® ME functions on PCI.

(A5) BIOS initializes HECI interfaces

(A6) If the Intel® ME Firmware Status #1 says that Intel® ME runs Intel® SPS firmware in operational or recovery mode BIOS sends to Intel® ME "Get ME-BIOS Interface Version" message to check Intel® ME-BIOS interface definition compatibility and supported firmware features. In case Intel® ME runs in recovery mode, all Intel® ME features are disabled.

Implementation Note: Intel® ME -BIOS interface version used by Intel® ME operational and recovery firmware may differ if operational firmware was updated and recovery not. Thus, BIOS may need to support a range of Intel® ME-BIOS interface versions

(E3) If BIOS and Intel® ME firmware are not compatible, BIOS logs error for user, disables all HECI functions and jumps to step (E4) without any more communication with Intel® ME.

(E4) It is expected that BIOS POST should inform the user about any problem with Intel® ME communication. At minimum MEFS1 and MEFS2 should be logged so user is warned about any problem with Intel® ME firmware initialization.

(E5) If MEFS1.CurrentState indicates that Intel® ME firmware runs in recovery mode BIOS jumps to step (A11).

(A7) BIOS can configure selected Integrated Clock parameters. This step is optional. It is not required when the ICC settings provided into FITc during Intel® ME region image creation do not require any adjustments.

(A8) BIOS enables PCH thermal sensor

(A9) When Intel® NM is enabled BIOS reads from the NMFS the number of CPU cores that should be disabled in each processor package.

(A10) When Intel® NM is enabled in Intel® ME firmware, BIOS sends the host configuration information to Intel® ME. The information whether Intel® NM functionality is enabled can be found in Get Intel® ME BIOS Interface Version response

(A11) If MCTP Bus Owner Proxy capability is enabled in Intel® ME firmware BIOS sends Set MCTP Bus Owner Proxy Configuration HECI message to Intel® ME

(A12) If Direct Intel® ME Firmware Update is supported BIOS sends HMRFPO_LOCK message to retrieve the nonce word from Intel® ME firmware and protects the Intel® ME region area

with SPI Protected Range Register (PRx). Nonce could be then used to perform the Direct Intel® ME Firmware Update and should be preserved in the memory region hidden and protected from the OS (SMM memory region).
Implementation Note: Since SMM memory is locked before option ROMs execution, this step also needs to be done before Option ROMs execution.

(A13) BIOS sends END_OF_POST message to Intel® ME to indicate that OS is to be loaded.
(A14) BIOS disables or hides Intel® ME devices
(A15) BIOS disables Global Platform Reset capability in PCH to prevent OS from initiating Global Platform Reset
(A16) BIOS invokes OS loader

For further information on BIOS POST requirements, please refer to the Intel® Server Platform Services (Intel® SPS) Firmware Intel® ME BIOS Interface for Grantley.

9.2.2 General BIOS requirements

Id	**Description**
G1	BIOS must check in the MEFS1 and MEFS2 registers whether Intel® ME is functional. If not, it should skip all Intel® ME-related tasks. Note: The BIOS should log this event in the BMC System Event Log.
G2	BIOS should log and display MEFS1 and MEFS2 if error is detected in the MEFS1.Error Code. The BIOS should also add a record to the BMC System Event Log
G3	BIOS must send the register based DRAM_INIT_DONE message to Intel® ME
G4	System BIOS should send SPS_GET_MEBIOS_INTERFACE message to Intel® ME firmware and check the Intel® ME-BIOS compatibility
G5	When IPMI sensor for PCH Thermal Sensor is enabled in Intel® ME firmware BIOS must configure and enable the sensor in PCH
G6	BIOS must hide Intel® ME devices from OS
G7	System BIOS must send END_OF_POST message to Intel® ME via HECI-1
G8	BIOS must execute the Warm Reset Notification subflow during each S0 entry procedure

BIOS requirements for Intel® NM enabled firmware

Id	Description
NM1	BIOS should support performance vs. power optimized POST execution.
NM2	BIOS must pass processors configuration data via HECI-1 interface
NM3	BIOS must initialize the HECI-2 interface.
NM4	BIOS must implement performance change notifications support in ACPI tables
NM5	BIOS should implement processor utilization notifications support in ACPI tables
NM6	BIOS must support UEFI EDK II Framework with MP Services Protocol in order to support Intel® NM Power Thermal Utility

NM7	If BIOS Thermal Utility feature is desired, the BIOS must opt-in to support Intel® NM Power Thermal Utility. This is required for Intel® NM to support the Intel® NM Power Thermal Utility feature. This feature is not required in WCS
NM8	BIOS should implement HW change notification via HECI-1 interface.
NM9	BIOS must check the signature in Expansion ROM Header of the Intel® NM Power Thermal Utility
NM10	BIOS must send Intel® NM Host Configuration is before PCI enumeration.
NM11	BIOS must send Intel® NM Host Configuration message at least 2 ms before PCI Enumeration. This is required for successfully launching Intel® NM Power Thermal Utility Expansion ROM

9.2.3 Intel® ME requirements for system memory

HECI-1 MISC_SHDW register defines an interface for Intel® ME to request BIOS to allocate a block of system memory for Intel® ME use. This block of memory is called Intel® ME UMA. Intel® SPS firmware does not use the Intel® ME UMA so it always requests for zero bytes of Intel® ME UMA.

9.2.4 Platform Power Capping

The Intel® ME should be support Fast NM (Node Manager) power capping.

To enable this feature, sample averaging for power must be disabled in the Hot Swap Controller.

On the ADM1278 Hot Swap Controller, the register to be configured is at address 0xD4 (PMON_CONFIG). The PWR_AVG bits (bits [13:11]) should be set to ‘b000.

This will disable sample averaging for power in the HSC and improve Fast NM Limiting reaction time.

9.2.5 Platform power limit at boot policy

Intel® ME FW implements platform power limit during boot. If BIOS escapes Low Frequency Mode (LFM) while under power cap, Intel® ME will retain processors in LFM mode.

After Intel® ME FW boots but before RAPL control would be available it will enforce the power limit by using PROCHOT#.

Once RAPL is up, Intel® ME FW will use RAPL controls to apply configured power limit policies

9.2.6 Handling Intel® ME errors

System BIOS may encounter several erroneous situations signaled in Intel® ME firmware status registers. The errors can be fatal for Intel® ME, but not necessary fatal for the server system. The way BIOS handles the errors should depend on Intel® ME role in the system. If it is silicon enabling only functionality is likely that the system will function quite well without Intel® ME. Therefore, it is recommended to just log the Intel® ME error to system log and boot the system normally. If Intel® NM is enabled in Intel® ME the error action depends on importance of Intel® ME services to system health. E.g. if power limiting functionality is crucial to protect system power supply, or server farm power supply it may be necessary to halt the system on Intel® ME error, or boot OS in LFM mode. To handle such situations, it is recommended for BIOS to implement user configuration option in BIOS setup to define Intel® ME error handling policy. The system administrator should be able to define whether the server system should:

- halt on Intel® ME error, or
- boot in LFM mode, or
- boot normally.

The WCS Blade should boot normally in the presence of ME errors.

The following erroneous situation can be recognized in Intel® ME firmware status registers:

1. Fatal Intel® ME error If nonzero error code is reported in MEFS1.ErrorCode, Intel® ME firmware is not running. This error should be traced in system log. The error action should depend on Intel® ME role in the system design as described above. The corrective action may require rewriting all Intel® ME region content using Security Strap Override jumper.

2. Intel® ME in recovery mode If MEFS1.CurrentState is 2, Intel® ME firmware is running in recovery mode. It means that Intel® ME functionality is reduced to just versioning and firmware update. There can be several reasons of the recovery mode. The reasons are listed in MEFS2.RecoveryCause:

   0. Intel® ME recovery jumper asserted
   1. Security strap override jumper asserted
   2. IPMI command
   3. Invalid flash master access configuration
   4. Intel® ME internal error

   Cases 0..2 are caused by user actions and are not erroneous, but it is recommended to trace them to the system log for informational purposes.

   Case 3 is a flash descriptor configuration fault and requires update of flash descriptor region to fix it.

   Case 4 may indicate Intel® ME firmware or hardware error and should be logged in the system log and reported to Intel support. BIOS may consider performing Global Platform Reset (see 3.9) to recover from this situation, but it should be traced to avoid repeating global reset in a loop if Intel® ME error is persistent. Every boot with positive Intel® ME status should clear the trace of global reset caused by Intel® ME error.

3. Intel® ME flash region errors

Several Intel® ME status bits may indicate errors in Intel® ME region on flash:

MEFS1.FPTorFactoryDefaultsBad

When this bit is set the current configuration may even work and Intel® ME can be operational but reset to factory defaults is not possible and it should be fixed as soon as possible by writing the whole Intel® ME region with valid Intel® ME region image. BIOS should trace this error in system log and choose to boot system normally or handle Intel® ME recovery according to MEFS1.CurrentState.

MEFS1.RecoveryBUPLoadFault

When this bit is set the Intel® ME recovery firmware is broken in the Intel® ME region and it should be fixed as soon as possible by writing the whole Intel® ME region with valid Intel® ME region image. BIOS should trace this error in system log and may boot system normally if MEFS1.CurrentState says that Intel® ME is operational.

MEFS2.MFSFailure

When this bit is set Intel® ME informs that Intel® ME File System failure has been detected during recent Intel® ME boot. If possible this situation is automatically fixed by restoring factory defaults. It means that user modifications to Intel® ME configuration are lost. This situation should be traced in system log. Restore to factory defaults is not possible if also FPT or Factory Defaults Bad bit is set in MEFS1 register. It is likely that Intel® ME will start in recovery mode if reset to factory defaults was not possible. If reset to defaults was successful and Intel® ME started in normal, operational mode, this bit will be cleared at next Intel® ME restart. The only corrective action needed is to restore user configuration.

MEFS2.TargetImageBootFault

When this bit is set it says that the desired operational image could not start and either rollback operational image is loaded if it is dual-image configuration, or Intel® ME runs in recovery mode. Normal Intel® ME firmware update should be performed to recover from this situation.

9.2.7 Intel® ME firmware status registers

The Intel® ME firmware writes status information about its current state in two 32-bit registers in the HECI-1 PCI configuration space: HFS at offset 40h, and GS_SHDW at offset 48h. BIOS should read these registers to determine current state of the firmware. If the MEFS1.ErrorCode value is not zero, the code should be made available to the BMC, so BMC can log the error as a part of the POST error logging. This information is not available to BMC in any other way.

9.2.8 EOP indication requirement

System BIOS is a trusted platform component and its interaction with Intel® ME firmware does not necessitate any strong security requirements. In order for Intel® ME firmware to determine when host applications start running on the host system, BIOS is required to send an END_OF_POST message to Intel® ME right before OS boot or OS resume. Upon receiving this message, Intel® ME imposes higher security requirements on the host interfaces. The message must be sent every time when BIOS POST is executed, i.e. on S5 resume and after G3. If it is sent Intel® ME silently drops it.

The EOP message is not to be sent if MEFS1.CurrentState indicates that Intel® ME firmware is disabled, or MEFS1.ErrorCode indicates that Intel® ME is in error state. Intel® ME firmware is not running in this case.

The END_OF_POST response message is sent by Intel® ME to BIOS in response to the END_OF_POST message. BIOS should wait for the response to make sure Intel® ME received EOP notification. The MEFS1.EOPStatus bit can also be observed to make sure Intel® ME accepted the END_OF_POST request message.

If EOP response is not received and MEFS.EopStatus indicates that Intel® ME did not receive EOP, BIOS POST should notify the BMC and log a System Event Log before taking error action appropriate for Intel® ME role in the system. See BIOS requirements section above.

Note: BIOS must send END_OF_POST message prior to boot to EFI Shell as EFI Shell, like OS, allows user to run applications

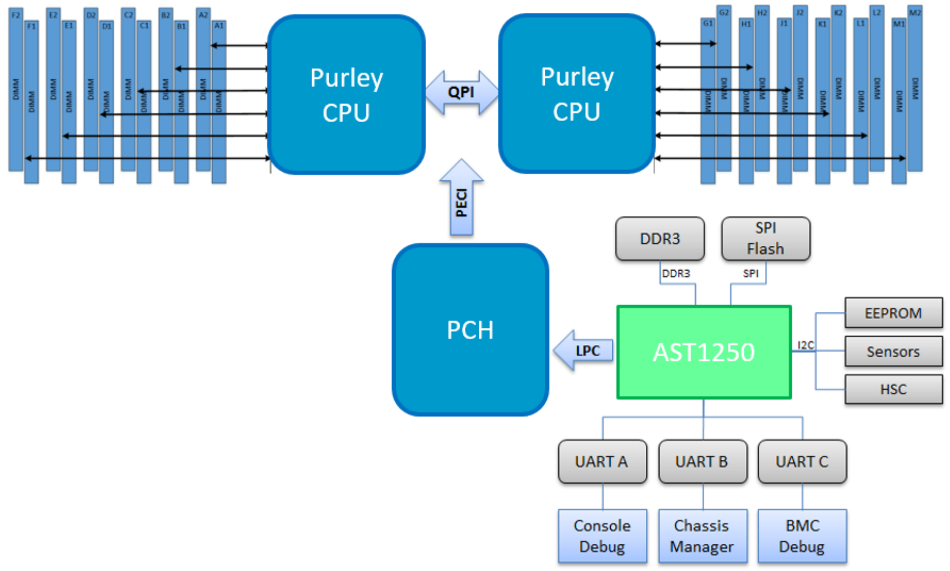
9.2.9 PCH thermal Management

Intel® SPS firmware offers monitoring PCH temperature using standard IPMI sensor defined in [ME IPMI]. The IPMI sensor provides PCH temperature retrieved by Intel® ME from PCH using PCH Thermal Sensor. Intel® SPS firmware does not require sending any MEI messaging related to the PCH Thermal Management. It is only required to enable the sensor by setting bit Enable Thermal Sensor (ETS) in Thermal Sensor Enable and Lock (TSEL) register at offset 8 in Thermal Reporting Registers region of PCI device 31 function 6.

9.3 Baseboard Management Controller (BMC)

The baseboard is populated with an AST1250 Baseboard Management Controller (BMC). This controller is independent to the host system software. This microprocessor has its own memory, processor and IPMI compliant system firmware. The host system can be managed by the BMC when the host system operating systems is off, or the system is soft powered down.

Architecture overview:



The AST1250

BUS No	Device	Slave Address
I2C1	AMI Decoder – MG9086	0xC0
I2C2	Baseboard FRU	0xA8
	Outlet Temp Sensor	0x9C
	Inlet Temp Sensor	0x9E
I2C3	FPGA Mezz card Temp Sensor	0x98
I2C4	ME_HOST_SMB	
I2C5	ME_SMLINK0	0x2C
I2C6	10G Re-timer	0x30
I2C8	Hot-Swap Controller (HSC)	0x40

9.3.1 LPC Interface

The AST1250 supports a 33 MHz LPC bus interface. The LPC performs serial transfer of cycle type, address, and data, synchronized with the 33 MHz PCI clock. The AST1250 supports 3 sets of KCS mode registers. For IPMI, LPC provides hardware path for KCS interface.

9.3.2 UART Interface

AST1250 has 3 independent UART interfaces 2 fully compliant with the 16550 standard and separate transmit and receive FIFO buffer (16x8). Chassis Manager serial commands will go through UART (TBD) of the BMC. UART (TBD) is able to redirect system console and share the same debug console header on the motherboard. UART (TBD) is dedicated to BMC debug. The baud rate for all UART communications is 115.2k.

9.3.3 GPIO Controller

The AST1250 supports up to 110 GPIO pins (14 sets). Each GPIO pin can be programmed to support 8mA or 12mA driving strength.

9.3.4 I^{2}C Interface

The AST1250 has 15 sets of multi-functional I2C/SMBus bus controllers. Each controller can be programmed as a master or a slave controller. DMA command mode support transmitting/receiving 512 bytes at a time.

9.3.5 SPI Controller

BMC SPI Controller (SPI) implements 3 types of application mode: SPI Master, SPI Slave to AHB bus bridge or SPI Pass-through.

9.3.6 Keyboard Controller Style (KCS) Interface

The KCS interface is one of the supported BMC to SMS interfaces. The KCS interface is specified solely for SMS messages. SMM messages between the BMC and an SMI Handler will typically require a separate interface, though the KCS interface is designed so that system software can detect if a transaction was interrupted.

The KCS Interface is designed to support polled operation. Implementations can optionally provide an interrupt driven from the OBF flag, but this must not prevent driver software from the using the interface in a polled manner. This allows software to default to polled operation. It also allows software to use the KCS interface in a polled mode until it determines the type of interrupt support. Refer to the IPMI 2.0 specification for further information.

9.3.7 IPMI Interface

9.3.7.1 Intelligent Platform Management Interface (IPMI)

The term Intelligent Platform Management refers to autonomous monitoring and recovery features implemented directly in platform management hardware and firmware. The key characteristic of Intelligent Platform Management is that inventory, monitoring, logging, and recovery control functions are available independent of the main processors, BIOS, and operating system. Platform management functions can also be made available when the system is in a powered down state

BIOS should be responsible for the initialization or startup of certain functions in the management controller, such as setting the initial timestamp time in the SEL and/SDR devices. BIOS should also perform tests of the platform management controller during POST. It is required that BIOS include provisions for checking and reporting on the basic health of BMC by executing the Get Self-Test Results command and checking the result.

It’s expected that BIOS features that take advantage of IPMI. For example, it is expected that the BIOS will use IPMI to log POST errors, or to log ‘system boot’ events so that events can be tracked relative to the last boot time. Another expectation the system will utilize the IPMI Watchdog Timer function with BIOS.

The BIOS should include support for serial port sharing; where by the BMC serial controller can be shared between the BMC and BIOS to provide IPMI basic mode support and serial console redirection. There is also a set of ‘boot flags’ that BIOS can must read to direct its operation following a system management initiated reset, power cycle , or power up.

9.3.8 SMM

System Management Mode. A special mode of Intel IA -32 processors, entered via an SMI. SMI is the highest priority non -maskable interrupt. The handler code for this interrupt is typically located in a physical memory space that is only accessible while in SMM. This memory region is typically loaded with SMI Handler code by the BIOS during POST.

9.3.9 SMI Handler

Certain platform management events come from baseboard interrupts. Such as correctable and uncorrectable ECC errors, critical NMIs (Non-maskable Interrupts) such as PCI PERR (parity error), PCI SERR (system error), bus timeout interrupts, hardware initialization failures, DIMM changes/failures, POST errors, etc. The platform management hardware maps these ‘critical interrupts’ to the system SMI (System Management Interrupt) signal. The SMI Handler runs, and, as part of handling these critical interrupts, generates an Event Message to cause the event to get logged in the SEL. The SMI Handler can also take autonomous, ‘emergency’ action, such as powering off or resetting the system, or propagating an NMI to the operating system.

The SMI Handler is typically a routine that is loaded and initialized into a protected area of memory by the BIOS.

SMI is the highest priority non-maskable interrupt in the system. When asserted, it switches the processors into ‘System Management Mode’ (SMM). Upon entry into SMM, the processor state is saved, and a memory configuration is entered where the SMI Handler has full access to system memory and I/O space. This allows the SMI Handler to implement its management functions in an OS-independent manner. The key aspect to this being that the SMI Handler code will run even if the OS is ‘hung’. This makes it ideal for implementing certain critical and emergency management functions.

9.3.10Critical Events and System Event Log Restrictions

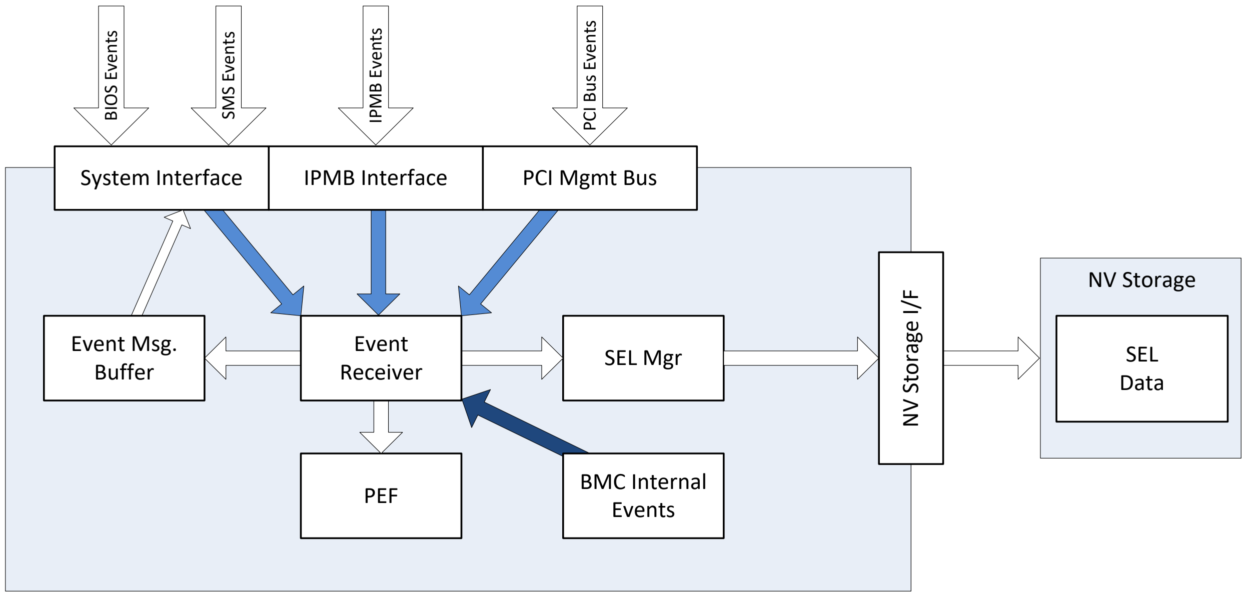
The platform’s System Event Log should be at least 4KB. Therefore, it is important to refrain from filling the System Event Log with non-critical ‘clutter’.

The System Event Log is primarily intended for capturing Critical Events. These include events that require immediate logging to guarantee that they’re available for ‘post-mortem’ analysis, and events that may require quick system responses, such as system power off, or shutdown.

Critical events include out-of-range temperature and voltage events, hardware failures such as power supply or fan failures, interrupts and signals that affect system operation such as NMIs and PCI PERR(parity error) and SERR (system error). Critical Events also include events that impact system data integrity, such as the uncorrectable ECC errors, or system security, such as TPM presence change.

In addition to events that indicate 'failure' conditions, events that indicate impending failures are also considered to be critical events. This includes events for reaching 'warning levels' for things such as system temperature or error counts. The assertion of 'Predictive Fault' information is also considered critical, particularly if the monitored device does not have a direct 'failure' indication.

Non-critical events, such as the return to an 'OK' state from a 'Warning' state should not be sent as critical events. Non-critical system information is normally obtained by System Management Software polling sensors and management controllers for their status.



The image above presents a conceptual illustration of the manner in which Event Message scan be handled by a Baseboard Management Controller device that uses an external non-volatile storage device to hold the System Event Log.

The figure shows a BMC with a shared system messaging interface where Event Messages can be delivered from either BIOS, SMS (system management software / OS), or an SMI Handler, and an IPMB interface and through which it can receive Event Messages from the Intelligent Platform Management bus. The BMC can also generate 'internal' Event Messages.

When the BMC receives a message via the system or IPMB interfaces, a 'Message Handler' function recognizes the message as being for the 'Event' functionality in the BMC and passes the message information on to the 'Event Receiver' function. The Event Receiver function then takes the message content and issues a request to a 'SEL Mgr.' function that formats the message as an SEL Entry and calls the FLASH Interface to have the data stored.

The Event Receiver function is also responsible for driving the response message back through the messaging system. This way, message acknowledgment or error reporting can be provided

Please refer to the Intelligent Platform Management Bus Communications Protocol Specification for additional information on Event Message handling

9.3.11Watchdog Timer

IPMI defines common command interfaces for configuring and accessing a watchdog timer function in the BMC. This timer can be used as an aid in monitoring the health of BIOS and system software. The watchdog timer can be used by different types of software such as BIOS, pre-boot, OS, and system management software. Once started the timer must be periodically reloaded by software in order to keep it from expiring. If software ceases to run, the timer will expire and generate a timeout act ion.

The IPMI definition allows different actions to be selected to occur on a watchdog timeout. This includes reset, power off, power cycle, etc. and a 'pre-timeout interrupt' option that, if provided, can be used to generate a system interrupt shortly before the timeout. The definition includes 'timer use' fields that keep track of what type of software (BIOS, OS, System Management Software, etc.) started the timer. The timeout action and 'timer use' information can be automatically logged to the SEL when the timeout occurs. This provides a record of when the timeout occurred, what software was using the timer, and what action was taken.

Watchdog Timer Actions

The following actions are available on expiration of the Watchdog Timer:

- System Reset
- System Power Off
- System Power Cycle
- Pre-timeout Interrupt(OPTIONAL)

The System Reset on timeout, System Power Off on timeout, and System Power Cycle on timeout action selections are mutually exclusive. The watchdog timer is stopped whenever the system is powered-own. A command must be sent to start the timer after the system powers up

9.3.12Watchdog Timer Use Field and Expiration Flags

The watchdog timer provides a 'timer use' field that indicates the current use assigned to the watchdog timer. The watchdog timer provides a corresponding set of 'timer use expiration' flags that are used to track the type of timeout(s) that had occurred.

The timeout use expiration flags retain their state across system resets and power cycles, as long as the BMC remains powered. The flags are normally cleared solely by the 'Set Watchdog Timer' command; with the exception of the "don't log" flag, which is cleared after every system hard reset or timer timeout.

The Timer Use fields indicate:

BIOS FRB2 timeout	An FRB-2 (fault-resilient booting, level 2) timeout has occurred. This indicates that the last system reset, or power cycle was due to the system timeout during POST, presumed to be caused by a failure or hang related to the bootstrap processor
BIOS POST timeout	In this mode, the timeout occurred while the watchdog timer was being used by the BIOS for some purpose other than FRB -2 or OS Load Watchdog.
OS Load timeout	The last reset or power cycle was caused by the timer being used to 'watchdog' the interval from 'boot' to OS up and running. This mode requires system management software, or OS support. BIOS should clear this flag if it starts this timer during POST.
SMS 'OS Watchdog' timeout	This indicates that the timer was being used by System Management Software. During run-time, System Management Software (SMS) starts the timer, then periodically resets it to keep it from expiring. This periodic action serves as a 'heartbeat' that indicates that the OS (or at least the SMS task) is still functioning. If SMS hangs, the timer expires, and the BMC generates a system reset. When SMS enables the timer, it should make sure the 'SMS' bit is set to indicate that the timer is being used in its 'OS Watchdog' role

9.3.13Using the Timer Use field and Expiration flags

The software that sets the Timer Use field is responsible for managing the associated Timer Use Expiration flag. For example, the BIOS sets the BIOS FRB2 timeout, the BIOS is responsible for acting on clearing the associated Timer Use Expiration flag.

9.3.14BIOS Support for Watchdog Timer

If a system 'Warm Reset' occurs, the watchdog timer may still be running while BIOS executes POST. Therefore, BIOS should take steps to stop or restart the watchdog timer early in POST. Otherwise, the timer may expire later during POST or after the OS has booted.

9.3.15Watchdog Timer Event Logging

By default, the BMC should automatically log the corresponding sensor-specific watchdog sensor event when timer expiration occurs. After 10 consecutive watchdog resets events are logged without a successful BIOS POST the BMC should temporarily suspend logging consecutive watchdog resets until POST succeeds. This requires the watchdog to clear the consecutive system event log counter before clearing the associated Timer Use Expiration flag. The purpose of suspending watchdog system event logging after 10 consecutive events is to prevent the SEL from rolling and overwriting important error messages that occurred before the initial watchdog event.

9.3.16Console Redirection with Serial Port Sharing

BMC provides a serial port for BIOS to use as the console and debug port as the standard solution for server design. The console redirection includes the functionality to have the serial output from Intel® C620 series chipset (PCH) to be redirected to BMC which in turn to be redirected to CM. For information on serial port sharing functionality refer to the WCS-Software-BladeAPI specification.

9.3.17BMC and BIOS communication

Due to AST1250 needing 3-5 seconds to decompress its firmware from flash memory and boot to embedded OS when AC on, BIOS power on must wait about 60 seconds then send the first command to BMC.

9.3.18Dynamic BIOS Configuration

The Dynamic BIOS Configuration feature is a requirement to support all WCS flavors through one BIOS image. Bios configuration i.e. Setup and Boot order settings are different across WCS flavor systems. Dynamic BIOS configuration can be selected through BIOS setup or though the BMC/Chassis manager Interface.

Refer to the WCS-Software-BladeAPI specification V2.9.5 or above for the Get/Set BIOS Config commands details. WCS flavor configuration changes across different systems will be provided.

BIOS setup should display the current BIOS flavor, this is for status display and an option to select the new BIOS flavor (may be called “Change BIOS flavor”). This setup options should be at the top of Boot option selection page. The defined BIOS flavors are listed below, other flavors which are not listed here are not supported.

	BIOS Setup menu	**IPMI OEM command**	**BIOS Name or Project Version example**
Dynamic BIOS Configuration	WCS General	0x00	C104n.BS.1A01.GN1
	Azure	0x01	C104n.BS.1A01.AZ1
	Azure PV1	0x11	C104n.BS.1A01.AU1
	Bing Online	0x02	C104n.BS.1A01.BA1
	Bing Offline	0x12	C104n.BS.1A01.BB1
	Exchange	0x03	C104n.BS.1A01.EA1

Below are BIOS requirements to support this feature.

1. BIOS flavor is updated though the BMC/Chassis manager Interface
   During POST BIOS get the BIOS flavor from BMC using IPMI OEM command “Get BIOS Config” and Byte 3 refer to chosen BIOS Configuration.
   - If there is no change in the BIOS flavor request continue with system boot flow.
   - If BIOS flavor change requested
     - All related setup options (current setting and default setting) will be changed based on the Flavor selection.
     - The BIOS name or Project Version in the setup and SMBIOS data is updated based on the BIOS flavor.
     - Update the current BIOS flavor information in the setup.

- Set the Active BIOS Configuration filed in the BMC
- Do the system reset for the configuration to take effect.

2. BIOS flavor is updated though BIOS setup
    - In addition to the steps listed above Update the Chosen BIOS configuration value in the BMC using the Set BIOS config command after user saved the BIOS flavor selection.
    - Do the system reset for the configuration to take effect.

When user selects update setup default values "Change BIOS flavor" option should not be updated.

In addition to the above requirements BIOS need to inform the available BIOS flavors to BMC using OEM IPMI command. The format of available BIOS configurations are an array of 4-byte BIOS configuration tuples, see description of this in the Get BIOS Config command response (Byte 4-255) in Blade API specification.

9.3.19Diagnose BMC

The first task for the BMC is BMC diagnosis. The BIOS has to validate whether the BMC is available. If BMC is not available even after resetting and retrying, then BIOS sets a flag and does nothing to support the BMC. Below is the flow chart and implementation:

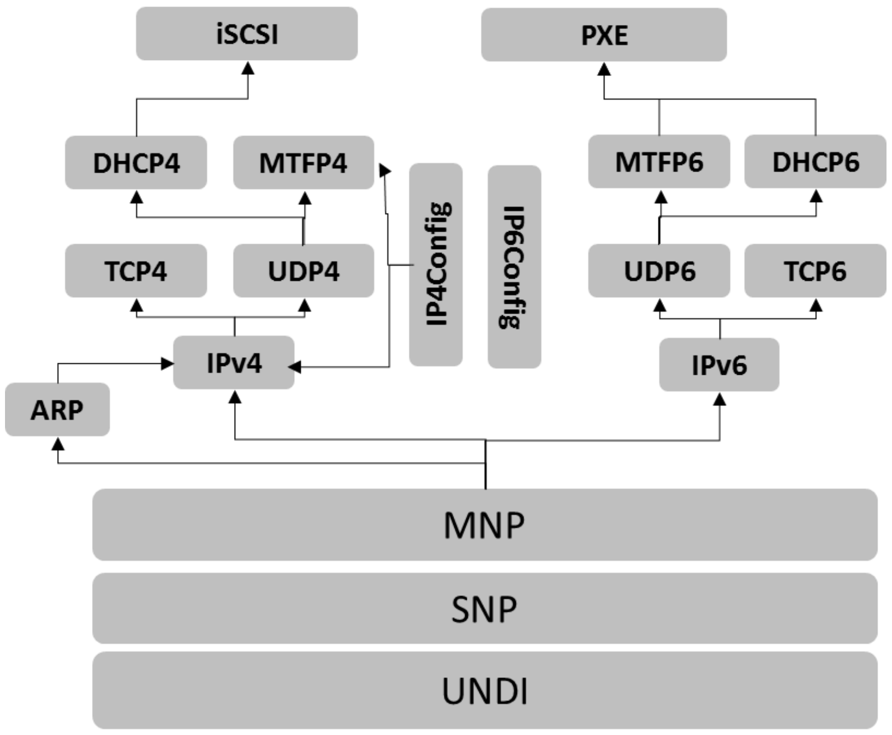
The BIOS should read the BMC_READY_GPIO on the BMC to determine if the BMC is available. An error retry timeout mechanism should be implemented for robustness. After the retry period times out and the BMC is not available, the system should set the BMC not available flag and continue POST. If the BMC READ GPIO pin is true then the BIOS should perform the Get Self Test Result and interpret the response. Should the BMC fail the Self-Test the system should set the BMC not available flag and continue POST.
X

10 Networking

10.1 PXE boot

WCS Intel® Xeon® Scalable Platform BIOS must support PXE boot in legacy as well as UEFI boot environments. For the legacy boot environment, this must be accomplished through CSM.

For UEFI boots, BIOS must implement following network stack as per UEFI 2.4 specification.



Network performance must be consistent in terms of network speed and PXE boot installation time between legacy and UEFI boot environments. BIOS integrates the NIC driver as well as the EDK network driver. No additional BIOS code is required to improve the speed.

10.2 MAC Address

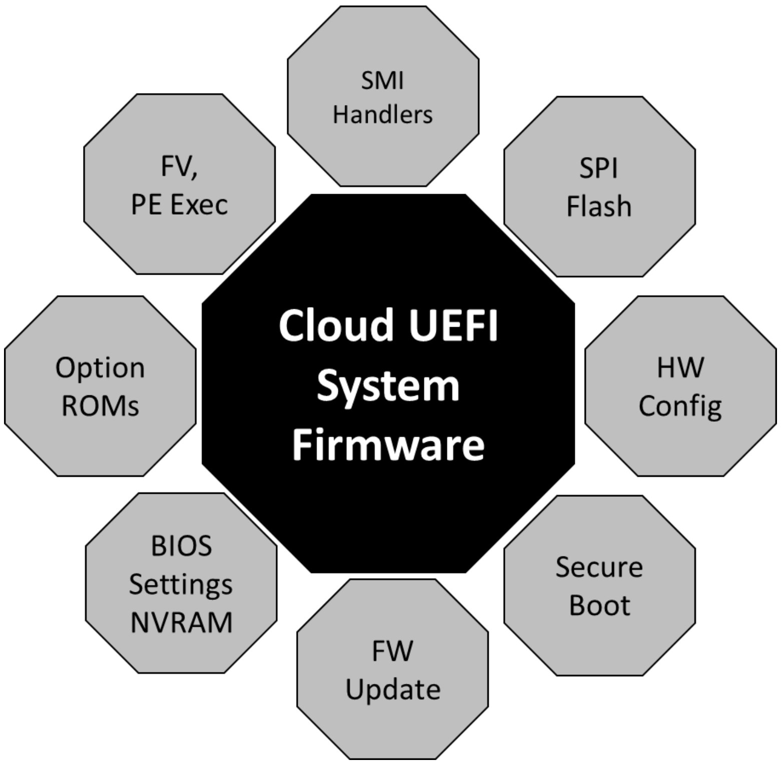
MAC address is assigned at manufacturing and stored as a fixed value in the add-in NIC card.

10.3 ARC Naming

If a boot device is not listed in the boot order list, BIOS support under both legacy and UEFI boot should still be available as usual as those already listed the boot order list. The BIOS code change can be copied from previous WCS blades if this functionality is not already part the current AMI BIOS code base.

11 Security

The climate for platform attacks is ever increasing. As operating systems become successively more hardened, attacks have been migrating into the platform firmware. It is important that one must properly identify and secure various platform assets. Typical classification of various assets of the Cloud UEFI System firmware are captured in the schematic below:



This chapter captures various security features that we must implement in a WCS BIOS in order to successfully protect the Cloud platform. These include:

- TPM
- Secure Boot
- Signed BIOS Update
- Intel® TXT

11.1 Trusted Platform Module (TPM) Initialization

The BIOS must implement support for a Trusted Platform Module (TPM) 2.0 component according to the TCG TPM Main and PC Client Specifications. In supporting the standard, the BIOS must implement all requirements outlined in those specifications, such as storage and generation of encryption keys, certificates, and platform attestation credentials.

11.1.1Physical Presence

Certain TPM commands can be executed only if physical presence is asserted. For the WCS platform, Physical presence is asserted through the BMC flag. The physical presence flag in the BMC is controlled through Get/Set TPM Physical Presence IPMI commands documented in the WCS – Software blade API specification version 3.8 or later.

TPM Administrator uses the Set TPM Physical Presence IPMI command before starting TPM admin operations like clearing the TPM. WCS UEFI BIOS uses the Get TPM Physical Presence command to check whether TPM physical presence is set before executing the TPM pending operations like clearing the TPM. And BIOS need to clear the physical presence flag using the Set TPM Physical Presence IPMI command after executing the TPM pending operations.

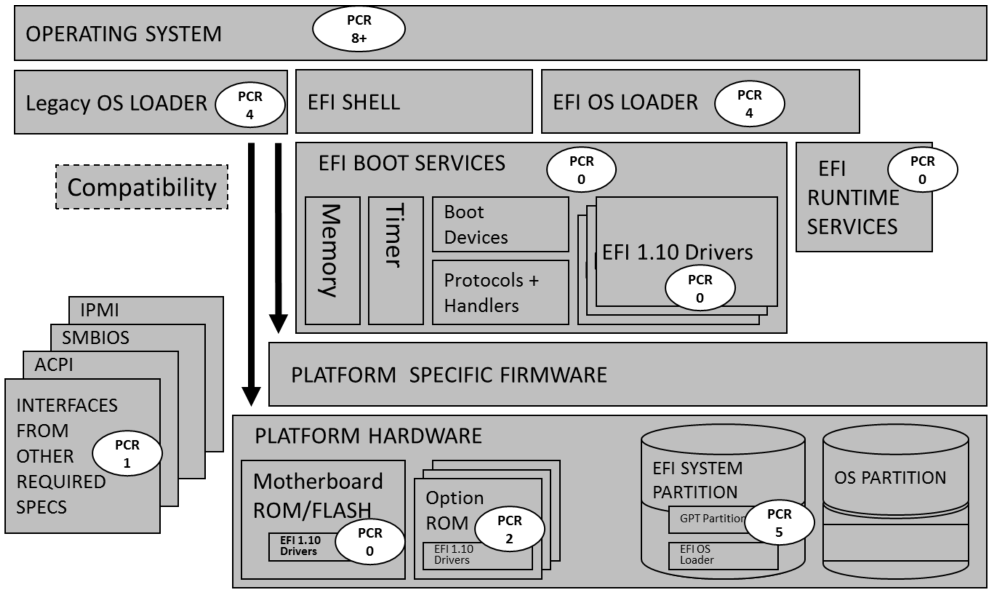
Other physical presence methods like user input during pre-boot should not be used as it requires local user present.

11.1.2PCR Measurement

BIOS is responsible for PCR measurements for various pre-boot components. TPM provides about 16 PCRs for BIOS & OS usages. PCR usage is pre-defined by TCG specification. Following is a classification of PCR measurement types.

- PCR Measurement Types
    - PCR[0] – BIOS Code
    - PCR[1] – BIOS Data
    - PCR[2] – Option ROMs
    - PCR[4] – OS Loader
    - PCR[5] – GPT Partition
    - PCR[8+] – PCRs for OS

PCR measurements of the pre-boot stack are schematically captured as below:

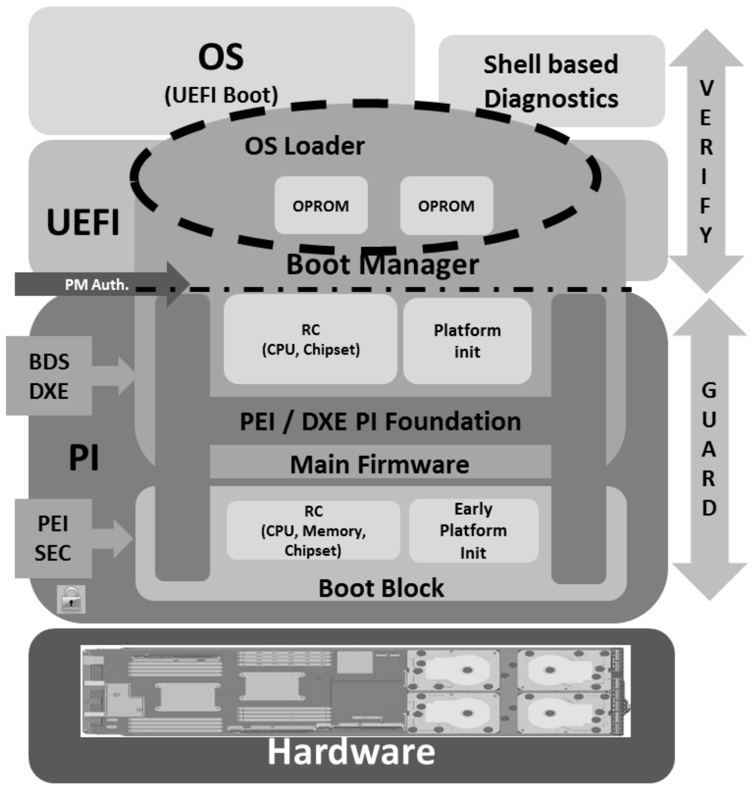


11.2 Secure Boot

11.2.1UEFI Secure Boot Overview

UEFI Secure Boot defines how a platform's firmware can authenticate a digitally signed UEFI image, such as an operating system loader or a UEFI driver stored in an option ROM thus providing the capability to ensure that those UEFI images are only loaded in an owner authorized fashion and providing a common means to ensure platforms security and integrity over systems running UEFI-based firmware.

The BIOS should be UEFI 2.4 compliant and WHCK UEFI compliant. Typical schematic of the UEFI pre-boot software stack with secure boot enabled is shown below:



11.2.2UEFI Secure Boot's Authenticated Variables

The UEFI Authenticated Variable Service is defined as an enhancement to the UEFI Variable Service. It provides a means to ensure the integrity of specified variables by prepending authentication data. This is done with an EFI_VARIABLE_AUTHENTICATION_2 descriptor as described in Section 7.2 of the UEFI Specification 2.4.

UEFI Secure Boot's image and certificate policies are controlled by the following UEFI authenticated variables, which are defined in section 3.2 of UEFI Specification 2.4

- **Platform Key (PK)** - The platform key establishes a trust relationship between the platform owner and the platform firmware. The platform owner enrolls the public half of the key (PKpub) into the platform firmware. The platform owner can later use the private half of the key (PKpriv) to change platform ownership or to enroll a Key Exchange Key. For UEFI 2.4, the recommended Platform Key format is RSA-2048.
- **Key Exchange Key (KEK)** - The Key exchange keys establish a trust relationship between the operating system and the platform firmware. The public part of the key (KEKpub) is enrolled into the platform firmware. Each operating system (and potentially, each third party application which need to communicate with platform firmware) can later use the private half of the key (KEKpriv) to communication with firmware in trusted manner. For UEFI 2.4, the recommended Key Exchange Key format is RSA-2048. The KEK can also contain authorized signing certificates.

- **Authorized Signature Database (DB)** - This database contains authorized signing certificates and digital signatures. An image signed with a certificate enrolled in DB (or KEK) or whose digital signature is enrolled in DB is authorized to execute. DB is the white list
- **Forbidden Signature Database (DBX**) - This database contains forbidden certificates and digital signatures. An image signed with a certificate enrolled in DBX or whose digital signature is enrolled in DBX, is never allowed to run. DBX is the black list.
- **Setup Mode** - When Setup Mode is null, no Platform Key is enrolled, and the platform is said to be operating in setup mode. While in setup mode, the platform firmware does not authenticate images and secure boot policy can be configured by writing the PK, KEK, DB and DBX variables. When Setup Mode is not null, a Platform Key is enrolled, and the platform is operating in User Mode. User Mode requires that all executable be authenticated before they are permitted to run if they were loaded from locations (fixed disk, removable disk or option ROM) enabled by the Secure Boot Policy PCD's described below.
- **SecureBoot** – When set (1), the platform is operating in secure boot mode and performs image verification based on the data stored in DB and DBX and the Secure Boot Policy PCD's described below.

11.2.3Secure Boot Policy

OEM's and IBV's can customize their platform's image verification policy by overriding the default policy values value for each type of device (fixed media, removable media or option ROM).

The possible policies are:

- ALWAYS_EXECUTE:
  - Always trust the executable. Allow it to run.
- NEVER_EXECUTE:
  - Never trust the executable. Do not allow it to run.
- ALLOW_EXECUTE_ON_SECURITY_VIOLATION:
  - Run images that are properly signed but whose signature is not found in the authorized database or is found in forbidden database.
- DEFER_EXECUTE_ON_SECURITY_VIOLATION:
  - Defer the running of images that are properly signed but whose signature is not found in the authorized database or is found in the forbidden database and add a record in the image execution information table as defined in the UEFI Specification.
- DENY_EXECUTE_ON_SECURITY_VIOLATION:
  - Do not run images that are properly signed but whose signature is not found in the authorized signature database or is found in the forbidden database and add a record in the image execution information table.
- QUERY_USER_ON_SECURITY_VIOLATION:
  - Query the user for a decision when an image is properly signed but its signature is not found in the authorized database or is found in the forbidden database and add a record in the image execution information table if user does not allow the image to run. Currently we use UI pop-up window as the query method.

11.3 Signed BIOS Update

The Secure Flash Update and Recovery feature will be enabled. ODMs need to follow proper instruction to build a Secure Flash Update enabled BIOS image, such as by removing default public keys and private keys before starting to build the BIOS images (.BIN and .ROM). Make sure that all relevant BIOS parameters tokens are appropriately set as required.
The built BIOS image (.ROM) will be sent to Microsoft for signing. Through a Microsoft CODESIGN server, Microsoft will sign the BIOS image with a public key and private key pair. The signed BIOS image (.CAP) will be sent back to the ODMs for validation. Following test cases for signed BIOS image validation should be included.

	Signed BIOS .CAP	**Unsigned BIOS .BIN**	**Tampered BIOS**	**Signed Recovery**	**Unsigned Recovery**
Signed BIOS .CAP	PASS	FAIL	FAIL	PASS	FAIL
Unsigned BIOS .BIN	PASS	PASS	N/A	PASS	PASS

To update a signed BIOS, Secure Flash Update enabled AMI utility (AFU) should be used. Final BIOS release package will include .BIN (for ME update) and .CAP (for BIOS update). BIOS release notes should include signed BIOS recovery steps.

11.4 Intel® Trusted Execution Technology (Intel® TXT) for Servers BIOS Requirements

11.4.1Summary of BIOS Initialization

The following is a list of BIOS requirements for supporting Intel® TXT for Servers:

- Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d). BIOS must implement "DMAR' table as specified in the Intel® Virtualization Technology for Directed I/O Architecture Specification 1.2.

- Implement the Firmware Interface Table (FIT). FIT to include the following records:
  - Exactly One type 0 record (FIT header)
  - One or more type 1 records (Microcode Update)
  - One type 2 record (BIOS AC Module)
  - One or more type 7 records (BIOS Startup Module Entry)
  - Zero or one type 8 record (TPM Policy Record)
  - Zero or one Type 9 record (BIOS Public Policy Record)
  - Zero or one Type 10 record (Intel® TXT Configuration Policy Record)
- Initialize the Trusted Platform Module (TPM) according to the TCG TPM PC Client specifications, except TPM_STARTUP.

- Provide hooks to call the BIOS AC module depending on the current state of the platform.
- Initialize Intel® TXT processor registers.
- Allocate Intel® TXT device memory.
- Initialize Intel® TXT chipset registers.
- Lock memory and SMRAM configuration before exiting Init BIOS code.
- Meet BIOS update requirements to prevent no-boot conditions. See Section 6.7.5.
- Provide a BIOS-based SINIT ACM and copy the SINIT ACM into memory as specified in the Intel® Trusted Execution Technology (Intel® TXT) Software Development Guide.

11.5 Security Code Review

Security code reviews are a critical part of firmware development life cycle. The source code for with WCS UEFI BIOS shall be subjected to security code reviews by security experts within Microsoft. Issues found during code review to be fixed on a priority basis and the priority will be determined by Microsoft team. To make the security code review efficient, the UEFI BIOS source must be constructed using latest versions of AMI Core and Technology modules, latest version of Intel reference code, Microcode and ME firmware. AMI CRB BIOS version is used for WCS BIOS shall be employed to track all the BIOS components version details.

All the UEFI security vulnerabilities found by third parties and applicable fixes by AMI should be included in to the Mt. Olympus source.

BIOS need to be tested with CHIPSEC tool.

11.6 BIOS firmware Volume Checksum SEL log

BIOS has to check for Firmware volume (FV) corruption and do the BMC SEL entry only if the FV is corrupted. In a normal boot where FV is not corrupted we should not see any FV checksum SEL log. FV corruption can be detected by checking the integrity of FV header and FFS files in FV.

For the format of the FV corruption SEL entry refer to the System event logs section.

FV Main corruption SEL entry should be done before executing any module from FV main. Boot block FV corruption SEL entry needs to be done as soon as the IPMI stack is available in PEI. It is accepted that Boot block FV is corruption can be detected only after part of the modules executed in boot block FV.

12 Error Handling

12.1 Platform Error Handling

Error handling relates to host of RAS features that are implemented on WCS Intel® Xeon® Scalable Platform platform. These include both error correction as well as logging. The latter, i.e. error logging, is particularly useful in planning for maintenance actions. On the other hand, error correction helps in restoring functionality of an otherwise failing node. So, the error handling operations can be classified as –

- Error Correction
- Error Logging
- Reporting status

12.1.1Error Sources and Types

One of the major requirements of server management is to correctly and consistently handle system errors. System errors that can be enabled and disabled individually or as a group can be categorized as follows:

- Processor Bus Error
- Memory ECC Error
- Ultra Path Interconnect (UPI) Error Events
- PCI Express* Errors
- Power On Self Test (POST) Error
- Power Error
- Software NMI Events
- MEMHOT# and PROCHOT#
- Machine Check Error
- Other IIO Error

The errors can be further categorized based on severity as –

- Correctable Errors (CE)
- Uncorrectable Errors (UCE)
  - Non-Fatal
  - Fatal

The BMC is capable of receiving errors from sensors and error signals. However, it is the responsibility of the BIOS to inform the BMC of system errors that BIOS has received. The BIOS should provide a mechanism for grouping errors, suppressing errors and setting error thresholds where applicable. When informing the BMC of errors, the BIOS is responsible for complying with the IPMI messaging format for System Event Log as specified in “**WCS Software Blade API Specification**”.

12.2 Summary of BIOS Platform Error Handling

The following table presents high level requirements for handling of platform errors by the BIOS during runtime.

Component	Type	Policy	SMM	OS
Memory	CE	Policy to log based on threshold and leaky bucket	• Log IPMI record to BMC • Create WHEA record	Poll for WHEA record
Memory	UCE		• Log IPMI record to BMC • Create WHEA record • Generate NMI	BSD using WHEA record and ERST
UPI	CE		• Log IPMI record to BMC • Create WHEA record	Poll for WHEA record
UPI	UCE	Genrates CATERR BMC logs SEL record	• None	
PCIe/IIO	CE	Policy to halt log based on threshold	• Log IPMI record to BMC • Create WHEA record	Poll for WHEA record
PCIe/IIO	UCE Non-Fatal		• Log IPMI record to BMC • Create WHEA record • Generate NMI	BSD using WHEA record and ERST
PCIe/IIO	UCE Fatal		• Log IPMI record to BMC • Create WHEA record • Generate NMI	BSD using WHEA record and ERST

12.3 Memory Error Handling

BIOS must implement two forms of Memory Error Handling, i.e. boot time as well as runtime. During boot time, BIOS should detect and isolate faulty DIMMs. During runtime, BIOS should first attempt to correct the errors, if feasible. If the errors are uncorrectable, BIOS must isolate enact user define policy, as specified in this section.

12.3.1Independent Channel Mode

In non-ECC and x4 SDDC configurations, each channel is running independently (non-lockstep), that is, each cache-line from memory is provided by a channel. To deliver the 64-byte cache-line of data, each channel is bursting eight 8-byte chunks. Back to back data transfer in the same direction and within the same rank can be sent back-to-back without any dead-cycle. The independent channel mode is the recommended method to deliver most efficient power and bandwidth as long as the x8 SDDC is not required.

12.3.2System BIOS Responsibility

This section provides a general overview of the System BIOS responsibilities for handling ECC implementations. There are four general items defined as:

- Initialize Hardware—During POST (Power On Self Test) the BIOS needs to detect the presence of ECC and Parity memory installed in the system. This information also needs to be stored in some form of NV RAM. The BIOS should also initialize all DRAM memory used in the system once ECC is enabled in the chipset with the error generation turned off.
- Report Errors—The BIOS needs to report errors back to the user. In situations where DMI BIOS Event Logging support is present, the event log must be updated with the error. Refer to the "DMI BIOS Support: Interface Requirements Revision 2.1" document for details on this interface. In situations where DMI BIOS Event Logging support is not present, the BIOS has to implement BMC SEL log per IPMI spec to leverage the BMC capabilities.
- Setup/Configuration Interface—The BIOS should provide a setup interface to allow the user to enable/disable ECC or parity checking in the system. In situations where DMI BIOS Event Logging support is present, setup should also provide a switch for enabling/disabling the logging of these type of events.
- Correction of Errors—Single-bit ECC errors are correctable. This is due to the fact that the specific data bit that has the error can be identified and corrected before the data is passed back to the requesting mechanism. The BIOS advanced menu should have a settable option for setting the single bit ECC error threshold and sliding time window. The BIOS is responsible for logging single bit errors in the System Event Log when the error count exceeds the threshold within the given time window. This is optimal because it allows the user to keep on working and in the best case being notified that there was an error. This allows the user to take whatever measures are necessary to correct the situation later. This may include replacing a bad DIMM in the case of hardware errors or simply having the memory in question "scrubbed" for noise-related errors.

"Scrubbing" involves reading a memory location and then writing the value and error bits back to DRAM. If there was a noise-related error it will be corrected. Otherwise, the incorrect checking information will always show up as an error and look like a hardware-related error if that specific memory location is not ever written out.

12.3.3Faulty DIMMs

The BIOS provides detection of a faulty or failing DIMM. A DIMM is considered faulty if it fails the memory test. A faulty DIMM should not halt POST. When initializing memory, the BIOS should detect any failed DIMM. If the failure can be isolated to a rank the rank should be disabled, otherwise the entire channel must be disabled. The error event should be logged in the System Event Log and boot sequence should continue.

12.3.4Correctable Errors

For each correctable error that occurs when the log threshold is reached, the BIOS will log a "Correctable Error" SEL entry. No other actions will be taken, and the system will continue to function normally

12.3.5Uncorrectable Errors

Uncorrectable Errors after POST should also be logged to the SEL and generate an NMI as described below.

12.4 NMI Generation

The BIOS shall generate NMIs to halt the system progress when uncorrectable errors occur at runtime.

Uncorrectable errors will be recorded in machine check. The handling of MCE must be appropriately implemented along with the NMI handing using modules in Intel Reference Code.

Should it be deemed necessary to mask any NMI events, the BIOS porting guide must be provided in writing to Microsoft with a list of NMI events which are believed necessary to be masked.

12.5 Enhanced Machine Check Architecture (EMCA)

Enhanced Machine Check Architecture (EMCA) Generation 2, or Enhanced Error Logging, enables the SMI handler to provide a detailed error log to the OS when MCA events occur. The log complements the machine check bank information and is expected to assist OS with better error isolation and predictive failure analysis. For example, BIOS should provide detailed information about the location of a failed DIMM in case of a memory error. BIOS must have a platform policy defined in order to enable/disable the feature through BIOS Setup. The default policy for this feature should be set as disabled.

12.5.1Capability Detection

BIOS detects the processor's Enhanced Error Logging capability by reading IA32_MCG_CAP (MSR 179h). Bit 26 indicates support for Enhanced Error Logging and the EXTENED_MCG_PTR MSR at address 793h. Note that SMM_MCA_CAP is only accessible when in SMM. Attempts to access this register from code running outside of SMM will result in a #GP fault.

12.5.2EMCA Generation 2

EMCA Gen 2 is a RAS feature that redirects Machine Check Exceptions (MCE) and Corrected Machine Check Interrupts (CMCI) to firmware first (via SMI) before sending it to the OS handler. Enhanced MCA enables BIOS-based recovery from errors.

With EMCA Gen 2 enabled, BIOS is able to configure each machine check bank to assert SMI instead of MCE and CMCI. The BIOS SMI handler is allowed to correct the error if possible before, optionally, causing an MCE or CMCI to be signaled once the SMI handler exits.

The BIOS is expected to implement Enhanced Machine Check Architecture (EMCA) Generation 1 and Generation 2. The BIOS detects the processor's EMCA Gen 2 capability by reading IA32_MCG_CAP (MSR 179h). Bit 25 of that register indicates support for Enhanced MCA. To determine which specific MCA banks support EMCA Gen 2, BIOS must read SMM_MCA_CAP[BANK_SUPPORT] (MSR 17Dh).

The BIOS should support SMI Generation for the redirection of correctable errors and uncorrectable to SMI for a specific MCA bank. The SMM handler may choose to handle a MCA error or it may simply log the error and propagate the MCA event to the OS. Refer the Intel® Xeon® Scalable Platform BIOS writers guide for implementation of the SMM EMCA Handler.

12.6 Error Injection

For all categories of errors, BIOS must support error injections using tools supplied by Intel RAS Tool, ITP based scripts provided by Intel and other error injection means for PCI / M.2 add-on devices.

12.7 WHEA Support

12.7.1ACPI Platform Error Interface

For the purposes of OS level platform error handling, BIOS must have complete support for Microsoft "Windows Hardware Error Architecture"(WHEA). According to ACPI Spec 5.0, "ACPI Platform Error Interfaces" (APEI) incorporates the Microsoft WHEA as an ACPI standard feature. The BIOS must publish WHEA-specific ACPI tables that describe the platform error interfaces for the OS. BIOS must also implement the ASL code to support and enable WHEA capability in the platform.

The BIOS must provide the following ACPI tables:

- Hardware Error Source Table (HEST) :
  - Extracts error information from platform hardware error registers.
- Error Injection (EINJ) Table :
  - Details the mechanism to inject a simulated HW Error to test WHEA error flow.
- Error Record Serialization Table (ERST) :
  - Persistent store of the WHEA Error Record to describe the serialization interface of the platform to the OS.
- Boot Error Record table (BERT) :
  - Captures fatal errors from the last boot that the BIOS or OS were unable to process.

12.8 Error Logging

12.8.1Runtime Error Logging via SMI Handler

The SMI handler is used to handle and log system level events that are not visible to the server management firmware. The SMI handler must pre-process all system errors, including those that are normally considered to generate an NMI. Sensors are managed by the BMC. The BMC is capable of receiving event messages from individual sensors and logging system events. For more information on BMC logged errors, see the BMC Specification.

12.8.2System Event Log

The BMC provides a mechanism for logging events in the IPMI standard System Event Log format. Reference the WCS-Software-BladeAPI specification for further details.

12.8.3Logging Format Conventions

The BIOS complies with the logging format defined in the IPMI specification. IPMI requires the use of all but two bytes in each event log entry, called Event Data 2 and Event Data 3. An event generator can specify that these bytes contain OEM-specified values. The system BIOS uses these two bytes to record additional information about the error. Event Data 2 and 3 are undefined for all other events that are logged by the BIOS.

The system BIOS sensors are logical entities that generate events. The BIOS ensures that each combination of sensor type (such as memory) and event type (sensor-specific) has a unique sensor number.

12.8.4Memory Error Logging and Reporting

12.8.4.1 Memory Error Reporting during BIOS POST

Memory errors are reported through MRC debug, System Event Logging and SMBIOS event logging. The following table shows the association of DIMM slots with SEL logging numbers:

DIMM Number in SEL	Mt. Olympus DIMM slot
1	A1
2	A2
3	B1
4	B2
5	C1
6	C2
7	D1
8	D2
9	E1
10	E2
11	F1
12	F2
13	G1
14	G2

15	H1
16	H2
17	J1
18	J2
19	K1
20	K2
21	L1
22	L2
23	M1
24	M2

Memory Error Reporting Agent Summary

Platform Element	Description
System Event Log	When a memory error occurs at runtime, the BIOS will log the error into the BMC System Event Log

12.8.4.2 Runtime Errors

The BIOS is responsible for logging single bit and multi bit ECC errors in the System Event Log. For multi bit ECC errors the BIOS should scan the machine check register on boot. Single bit errors should be logged when the single bit error count exceeds the threshold within the threshold time window.

The hardware is programmed to generate an SMI on correctable data errors in the memory array. The SMI handler records the error and the DIMM location to the system event log. Uncorrectable errors in the memory array are mapped to the SMI because the BMC cannot determine the location of the bad DIMM. The uncorrectable errors may have corrupted the contents of SMRAM. The SMI handler must log the failing DIMM number to the BMC if the SMRAM contents are still valid. The ability to isolate the failure down to a single DIMM may not be available on certain errors, and / or during early POST. Refer to the WCS-Software-BladeAPI specification for the event log entry format.

12.8.5Ultra Path Interconnect (UPI) Error Events Logging Format Convention

Refer to the WCS-Software-BladeAPI specification for the event log entry format.

12.8.6PCI Express* Errors Logging Error Format Convention

The hardware is programmed to generate an SMI on PCIe correctable, uncorrectable (non-fatal) and uncorrectable fatal errors. The correctable PCIe errors are reported to the BMC as PCIe Bus Correctable errors. PCIe non-fatal and fatal errors are reported to the BMC as PCIe Bus Uncorrectable errors. The system event log for these errors includes the location of the device reporting an error which includes the

PCIe link number, PCI bus number, PCI device number, and the PCI function number. An NMI is generated for PCIe Uncorrectable errors after they are logged. Refer to the WCS-Software-BladeAPI specification for the event log entry format.

Add PCie Error handling and masking if required. PCIe error masking should be used sparingly. Microsoft has to approve all masked PCIe errors. The ODM must seek written approval from Microsoft on the masking of any PCIe NMI events.

12.8.7Handling of PCIe Correctable error logging Limit

When PCIe correctable errors reach the threshold limit BIOS should log the BMC SEL event as per the format described in Table 1 PCIe Error Event Log 1.

Event Data 1 filed has updated with 0xC = Correctable error logging Limit Reached.

13 Firmware Update

13.1 BIOS Update Utility

The flash ROM contains system initialization routines. The complete ROM is visible; starting at physical address 4 GB minus the size of the flash ROM device. A 16-KB parameter block in the flash ROM is dedicated to storing configuration data that controls the system configuration (ESCD). Application software must use standard APIs to access these areas; application software cannot access the data directly

BIOS update utility or utilities should support the following environments: MS DOS, UEFI Shell, Windows-Server 2012, and WinPE 4.0. The utilities load a fresh copy of the BIOS into the flash ROM. The BIOS update may affect the following items:

- The system BIOS, including the recovery code, setup utility and strings.
- Onboard devices and other option ROMS for the devices embedded on the server board.
- Memory reference code.
- Microcode update

13.2 Signed BIOS Updates

Provision should be available to facilitate signing of the BIOS images. For the WCS platforms, the BIOS signing will be done by Microsoft.

13.3 BIOS Recovery

BIOS must support mechanism to perform recovery operation via USB Mass Storage device.

13.4 Back-up Partition

Intel® Xeon® Scalable Platform will have a redundant flash part. In case of BIOS image corruptions, BIOS should have support for performing recovery from a backup partition.

The current implementation by Intel® Customer Reference Board Crescent City will be leveraged into Mt. Olympus design. BMC will have GPIO pins that control the chip select signal, SPI muxing between PCH and BMC as well as ME_BIOS_SWAP.

13.5 One BIOS Image Considerations

For the WCS Intel® Xeon® Scalable Platform generation, one BIOS image must be supported, irrespective of the OEM or ODM. In addition, all platform flavors must be consolidated into one SKU.

Platform flavors mainly differ in setup defaults. Thus, appropriate setup defaults must be loaded during the boot based on the flavor for which the platform is provisioned. The platform flavor can be obtained via a BMC OEM command, see the chapter of BMC in section “Dynamic BIOS Configuration” for details.

In addition, some of the BIOS modules may be different with platform flavor. For such deviations, BIOS should have provision to load flavor specific drivers or launch flavor specific pre-boot functionality.

13.6 SEL Record

On each BIOS attempt, BIOS must log SEL record indicating that firmware update is attempted.

14 OS Boot Support

14.1 Software Design Specification: UEFI Operating System Support

- Windows Server 2012 or later
- WinPE 4.0 or later

14.2 Software Design Specification: Legacy Operating System Support

- Windows Server 2012 or later
- WinPE 4.0

14.3 Bootable Device types

BIOS must support OS boots from following types of devices / media:

- Network boot (PXE)
- USB media
- Disk devices
    - M.2 AHCI
    - M.2 NVMe
    - SATA
    - SAS
- EFI shell

15 BIOS POST Codes

This section presents list of postcodes that BIOS must implemented for the WCS Intel® Xeon® Scalable Platform.

There are two parts of the postcodes. First, WCS will acquired the latest version of AMI postcodes from the current AMI BIOS base code. Second, the postcodes from Intel RC will also be includes as part of the WCS requirements.

In addition to the BIOS Post Codes, a jumper will be used to turn on or turn off the serial debug output on the fly. The jumper is wired to a PCH GPIO pin as defined in the PCH GPIO pins list. The BIOS code has the support to detect the GPIO pin on the fly during boot when a status with message is coded for the current debug level.

16 Field Replaceable Unit (FRU)

The Field Replaceable Unit (FRU) information is used to primarily to provide inventory information. The system FRU storage format and configuration should comply with the Intel® "Platform Management FRU Information Storage Definition v1.0" specification revision 1.2.

The FRU information is stored by BMC in its in non-volatile memory. BIOS needs to query BMC to acquire the FRU data and display under BIOS setup accordingly.

The FRU should contain the following areas:

- Common Header
- Internal Use Area (Optional)
- Chassis Info Area
- Board Info Area
- Product Info Area
- Multi Record Area (128 bytes)

FRU Sample Layout:

```
Internal Use Area
---------------------------------------------------
Internal use data          = ""
---------------------------------------------------
Chassis Info Area
---------------------------------------------------
   Chassis Type         = "17h"
   Chassis Part Number     = "X873021-001"
   Chassis Serial Number    = "Auto Generated" 12 bytes auto generated.
   Chassis Custom field 1    = "2.0"
   Chassis Custom field 2    = Supplied by Microsoft
---------------------------------------------------
Board Info Area
---------------------------------------------------
   M/B Language Code      = "19h"
   M/B Manufacturer Date/Time = Generated by MFG
   M/B Manufacturer Name    = "Microsoft"
   M/B Product Name       = Supplied by Microsoft
   M/B Serial Number      = Generated by MFG
   M/B Part Number       = Supplied by Microsoft
   M/B Fru File ID       = "01"
---------------------------------------------------
Product Info Area
---------------------------------------------------
   PD Language Code       = "19h"
   PD Manufacturer Name     = "Microsoft"
   PD Product Name       = Supplied by Microsoft
   PD Part/Model Number     = Supplied by Microsoft
   PD Version          = "1.0"
   PD Serial Number       = Generated by MFG,same as board area above
   PD Asset Tag         = Microsoft Supplied, 10 bytes required.
   PD Fru File ID        = "01"
   PD Custom field 1      = Supplied by Microsoft
   PD Custom field 2      = Supplied by Microsoft
   PD Custom field 3      = Supplied by Microsoft
```

```
--------------------------------------------------
Multi Record Area
--------------------------------------------------
Filled by Microsoft, 128 bytes anticipated.
```

17 System Event Logs

WCS blade should support all standard IPMI 2.0 and DCMI system event logs. In addition, the system event logs defined in the following sections should be supported.

The following table shows the OEM record type and sensor type numbers that are reserved for the BIOS and BMC. Also see the WCS-Software-Blade-API specification for details on the SELs for BMC as well as ME.

Table 2. OEM Record Types Reserved for Use by BIOS and BMC

OEM Record Types	Owner
0xC0 – 0xCF	BIOS
0xD0 – 0xDF	BMC

Table 3. OEM Sensor Types Reserved for Use by BIOS and BMC

OEM Sensor Types	Owner
0xC0 – 0xCF, 0xD2	BIOS
0xD0 – 0xD1, 0xD3 - 0xDF	BMC

17.1 System Event Logs Generated by BIOS

This section describes the System Event Logs that are generated by the BIOS.

17.1.1QuickPath Interconnect (QPI) Error Logging

The BMC logs QPI errors using the following SEL format.

Table 4 QPI Error Event Log

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	System Event Record: 0x02
4:7	Timestamp	Timestamp
8:9	Generator Id	Byte 1 = 0x01 (Generated by BIOS) Byte 2 = 0x00
10	Evm Rev	0x04
11	Sensor Type	0x07

12	Sensor #	0x9D
13	Event Dir | Event Type	[7] Assertion: 0 [6:0] Event Trigger: 0x6F
14	Event Data 1	[7:6] ‘b10 = OEM code in byte 2 [5:4] ‘b10 = OEM code in byte 3 [3:0] Offset from Event/Reading Code for discrete event status 0xB - Uncorrectable Error 0xC - Correctable Error
15	Event Data 2	QPI Error Code: 0x32 - Tx CRC Error (initiates LLR_Req sequence) 0x31 - Rx CRC Error with LLR Success (no phy reset) 0x30 - Rx CRC Error with LLR Success after phy reset 0x22 - Phy In-band Reset & no width change21h - Not supported 0x20 - Phy Initialization Abort 0x1F - Unsupported Config Request from Message Channel 0x15 - Link Layer RBT Error 0x14 - Link Layer L0p Retrain error 0x13 - Link Layer Control Error 0x12 - Unsupported/Undefined Packet 0x11 - Tx Unsuccessful Link Layer Retry 0x10 - R3QPI Control Error 0x03 - latency buffer over/under run 0x02 - Drift Buffer overrun 0x00 - phy control error
16	Event Data 3	Bit 7:5 Reserved Bit 4 QPI Link ‘b0 = QPI 0 ‘b1 = QPI 1 Bit 3:0 Bitmap for processor number Bit0 = 1st processor Bit1 = 2nd processor Bit2 = 3rd processor Bit3 = 4th processor

17.1.2Memory ECC Error Logging

The BIOS logs memory ECC errors using the following SEL format. Refer to the WCS-Software-Blade-BIOS specification for details on the error logging.

The “Last Boot Error” value in Event Data 2 indicates fatal errors that caused the system to freeze or reset without triggering SMI, thus preventing the BIOS from handling the error. At the subsequent reboot, the “sticky” MCA and the “sticky” global Fatal Error Status CSR’s remain valid so BIOS can generate the BMC SEL entry along with the POST error code and WHEA BERT.

Table 5 Memory ECC Error Event Log

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	System Event Record: 0x02
4:7	Timestamp	Timestamp
8:9	Generator Id	Byte 1 = 0x01 (Generated by BIOS) Byte 2 = 0x00
10	Evm Rev	0x04
11	Sensor Type	0x0C
12	Sensor #	0x87
13	Event Dir | Event Type	[7] Assertion: 0 [6:0] Event Trigger: 0x6F
14	Event Data 1	[7:6] ‘b10 = OEM code in byte 2 [5:4] ‘b10 = OEM code in byte 3 [3:0] Offset from Event/Reading Code for discrete event status 0x0 = Correctable Error 0x1 = Uncorrectable Error 0x5 = Correctable ECC error logging Limit Reached
15	Event Data 2	0x00 = Single Bit Error warning threshold (Event/Reading Type Code = 0h for Correctable Error) if supported. 0x01 = Single Bit Error critical threshold (Event/Reading Type Code = 5h for Correctable ECC error logging limit reached) if supported. 0x10 = Last Boot Error 0xFF = unspecified Other values are reserved
16	Event Data 3	DIMM Number (1-base)

17.1.3PCI Express Error Logging

The BIOS logs PCIe errors using the following SEL format. The log entries should be added to the SEL in the order shown here. There can be one or more PCIe Error Event Log 2 entries depending on the number of errors that occurred.

Refer to the WCS-Software-Blade-BIOS specification for details on the error logging.

The "Last Boot Error" value in Event Data 2 indicates fatal errors that caused the system to freeze or reset without triggering SMI, thus preventing the BIOS from handling the error. At the subsequent reboot, the "sticky" MCA and the "sticky" global Fatal Error Status CSR's remain valid so BIOS can generate the BMC SEL entry along with the POST error code and WHEA BERT.

Table 6 PCIe Error Event Log 1

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	System Event Record: 0x02
4:7	Timestamp	Timestamp
8:9	Generator Id	Byte 1 = 0x01 (Generated by BIOS) Byte 2 = 0x00
10	Evm Rev	0x04
11	Sensor Type	0x13 (Critical Interrupt)
12	Sensor #	0xA1
13	Event Dir | Event Type	[7] Assertion: 0 [6:0] Event Trigger: 0x6F
14	Event Data 1	[7:6] 'b10 = OEM code in byte 2 [5:4] 'b10 = OEM code in byte 3 [3:0] Offset from Event/Reading Code for discrete event status 0x4 = PCI PERR 0x5 = PCI SERR 0x7 = Bus Correctable Error 0x8 = Bus Uncorrectable Error 0xA = Bus Fatal Error 0xC = Correctable error logging Limit Reached 0xF = Last Boot PCIe Error
15	Event Data 2	Bit [7:3] Device Number Bit [2:0] Function Number
16	Event Data 3	Bit [7:0] Bus Number

Table 7 PCIe Error Event Log 2

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC0 (PCIe Error)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	IANA Enterprise ID. See Device ID command for details.
11:12	Vendor ID	Vendor ID of error source device
13:14	Device ID	Device ID of error source device
15	OEM Data 1	1st Error ID (see following table for details)
16	OEM Data 2	For errors on M.2 devices, this field contains the following: [7:5] – [0000] [4:0] –logical M.2 on motherboard For other devices, this field contains the 2nd Error ID (see following table for details)

Table 8 PCIe Error Event Log 2 Error ID Detail

ID	Error	Default Error Severity	Transaction Response	Default Error Logging
70	Receiver Error	0	Respond per PCI Express specification	CORERRSTS
71	Bad TLP	0	Respond per PCI Express specification	CORERRSTS
72	Bad DLLP	0	Respond per PCI Express specification	CORERRSTS
73	Replay Time-out	0	Respond per PCI Express specification	CORERRSTS
74	Replay Number Rollover	0	Respond per PCI Express specification	CORERRSTS
75	Received ERR_COR message from downstream device	0	Respond per PCI Express specification	RPERRSTS
76	PCI Express Link Bandwidth changed	0	No Response -- This error is not associated with a cycle. I/O module detects and logs the error. Log per "Link bandwidth change notification mechanism" ECN	XPCORERRSTS
78	Advisory Non-Fatal Error	0	Respond per PCI Express specification	CORERRSTS
80	Received "Unsupported Request" completion status from downstream device	1	Coherency interface to PCI Express read: I/O module returns all 1s' and normal response to the coherent interface to indicate master abort Coherency interface to PCI Express NP write: I/O module returns normal response PCI Express to PCI Express	XPUNCERRSTS

			read/NP-write: 'Unsupported request' is returned2 to original PCI Express requester.	
81	Sent a PCI Express "Unsupported Request" response, on inbound request for address decode, request type, or other reason	1	PCI Express read: "Unsupported request" completion is returned on PCI Express PCI Express non-posted write: 'Unsupported request' completion is returned on PCI Express. The write data is dropped PCI Express posted write: I/O module drops the write data. Header is logged where possible.	XPUNCERRSTS HDRLOG
82	Received "Completer Abort" completion status from downstream device	1	Coherency interface to PCI Express read: I/O module returns all '1s' and normal response to the coherency interface Coherency interface to PCI Express NP write: I/O module returns normal response PCI Express to PCI Express read/NP-write: Completer Abort' is returned 3 to original PCI Express requester.	XPUNCERRSTS
83	Sent a PCI Express "Completer Abort" condition on inbound request for address decode, request type, or other reason	1	PCI Express read: 'Completer Abort' completion is returned on PCI Express PCI Express non-posted write: 'Completer Abort' completion is returned on PCI Express. The write data is dropped PCI Express posted write: I/O module drops the write data. Header is logged where possible.	XPUNCERRSTS HDRLOG
84	Completion timeout on NP transactions outstanding on PCI Express/DMI	1	Coherency interface to PCI Express read: I/O module returns normal response to the coherency interface and all 1's for read data Coherency interface to PCI Express nonposted write: I/O module returns normal response to the coherency interface PCI Express to PCI Express read/nonposted write: UR2 is returned on PCI Express Header is logged where possible.	UNCERRSTS HDRLOG
85	Received PCI Express Poisoned TLP	1	Outbound-Read Completion or Inbound Write: the packet is sent to the coherent interface. If POISFEN bit is set, the poison packet is set to its destination with poison bit set. If POISFEN is cleared, the packet is dropped. Peer to Peer read completion or write request: I/O module forwards packet with poisoned data to the destination port normally, but with the poison bit set. If POISFEN bit is set and the poison TLP severity is set as non-fatal, then this error is logged as an Advisory Non-fatal. Also, received poisoned TLPs that are not forwarded over the coherency interface are always treated as an Advisory Nonfatal	UNCERRSTS HDRLOG

			error, if severity is set to non-fatal. Header is logged where possible.	
86	Received PCI Express Unexpected Completion	1	Respond Per PCI Express Specification Header is logged where possible.	UNCERRSTS HDRLOG
87	PCI Express Flow Control Protocol Error4	1	Respond Per PCI Express Specification	UNCERRSTS
88	Received ERR_NONFATAL Message from downstream device	1	Respond per PCI Express specification	RPERRSTS
89	Received a Request from a downstream component that is unsupported	1	For Non-posted requests: 'unsupported request' response is sent and a bit in XPUNCERRSTS is logged. For posted requests, they are simply logged and dropped. Header is logged where possible.	UNCERRSTS HDRLOG
8A	Received a Request from a downstream component that is to be completer aborted	1	For Non-posted requests: 'completer abort' response is sent and a bit in XPUNCERRSTS is logged. For posted requests, they are simply logged and dropped. Header is logged where possible.	UNCERRSTS HDRLOG
8B	ACS Violation	1	Respond per PCI Express specification. An ACS violation will cause a violating request to be aborted and logged. Header is logged where possible.	UNCERRSTS HDRLOG
90	PCI Express Malformed TLP4	2	Respond Per PCI Express Specification Header is logged where possible.	UNCERRSTS HDRLOG
91	PCI Express Data Link Protocol Error4	2	Respond Per PCI Express Specification	UNCERRSTS
92	PCI Express Receiver Overflow	2	Respond Per PCI Express Specification	UNCERRSTS
93	Surprise Down	2	Respond Per PCI Express Specification	UNCERRSTS
94	Received ERR_FATAL message from downstream device	2	Respond Per PCI Express Specification	RPERRSTS
97	Outbound switch header queue parity error	2	Undefined. This is a fatal error for the given root port.	XPUNCERRSTS
98	MSI writes greater than a DWORD	1	Drop the transaction	XPUNCERRSTS
99	Outbound Poisoned Data	1	The poisoned data packet is sent but logged here.	XPUNCERRSTS

17.1.4BIOS Update Log

BIOS logs any updates to the BIOS flash image using the following SEL format. One SEL entry is logged at the start of the update, and another entry is logged when the update is completed.

The Firmware Naming table referred to in the table can be found in the Software-Arch specification.

OEM Data 2-4 refer to the version of the BIOS in the flash image.

Table 9 BIOS Update Event Log

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC1 (BIOS Update)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	IANA Enterprise ID. See Device ID command for details.
11	OEM Data 1	Update Status [7:4] 0x0 – Success 0xF – Failure Update Stage [3:0] 0x0 – Update Started 0x1 – Update Completed
12	OEM Data 2	Current BIOS Major Version [7:0] Major Version – corresponds to position 4 of the Firmware Naming table
13	OEM Data 3	Current BIOS Minor Version [7:0] Minor Version – corresponds to position 5 of the Firmware Naming table
14	OEM Data 4	Current BIOS Additional Version Information [7:4] Hotfix Version – corresponds to the numeric value in position 6 of the Firmware Naming table [3:0] Development/Test Version – corresponds to position 7 of the Firmware Naming table
15	OEM Data 5	Reserved = 0x00
16	OEM Data 6	Reserved = 0x00

Example record:

The current BIOS version is C2000.BS.1C02.GN2.1.bin. The following command is used to update the BIOS to C2000.BS.1C03.GN1.3.bin:

```
AFUWINx64.EXE C2000.BS.1C03.GN1.3.bin /X /P /N /R /B /ME /REBOOT
```

At the start of update, a SEL entry with the following fields is logged:

OEM Data 1 = 0x00

OEM Data 2 = 0x01 (mapped from 1C02, major version = 0x1C)

OEM Data 3 = 0x02 (mapped from 1C02, minor version = 02)

OEM Data 4 = 0x21 (mapped from GN2.1, hotfix = 2, test = 1)

At the end of the update, if the update completed successfully, a SEL entry with the following fields is logged:

OEM Data 1 = 0x01

OEM Data 2 = 0x01 (mapped from 1C03, major version = 0x1C)

OEM Data 3 = 0x03 (mapped from 1C03, minor version = 03)

OEM Data 4 = 0x13 (mapped from GN1.3, hotfix = 1, test = 3)

17.1.5Intel Memory Reference Code (MRC) Errors

BIOS logs any errors that occur during memory initialization using the following SEL format. The MRC major and minor error codes are provided by Intel’s Grantley-HSX-EP_RC_User_Guide, in the MRC Error Codes and Warning Log sections.

Table 10 Intel Memory Reference Code (MRC) Error Event Log

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC2 (MRC Error)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	IANA Enterprise ID. See Device ID command for details.
11	OEM Data 1	MRC Major Error Code
12	OEM Data 2	MRC Minor Error Code
13	OEM Data 3	Socket Number or 0xFF (unspecified)
14	OEM Data 4	Channel Number or 0xFF (unspecified)
15	OEM Data 5	DIMM Number or 0xFF (unspecified)
16	OEM Data 6	Rank Number or 0xFF (unspecified)

17.1.6 Intel® Xeon® Scalable Processor IIO Module Errors

The BIOS logs errors in the CPU IIO module using the following SEL format. Refer to the IIO Module Error Codes section in the Intel® Skylake EDS Vol. 1 specification for details and the error codes.

Table 11 Intel® Xeon® Scalable Processor IIO Module Errors

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	System Event Record: 0x02
4:7	Timestamp	Timestamp
8:9	Generator Id	Byte 1 = 0x01 (Generated by BIOS) Byte 2 = 0x00
10	Evm Rev	0x04
11	Sensor Type	0x13 (Critical Interrupt)
12	Sensor #	0xA7
13	Event Dir | Event Type	[7] Assertion: 0 [6:0] Event Type: 0x70 (OEM Discrete)
14	Event Data 1	Bit [7:6] ‘b10 Bit [5:4] ‘b10 Bit [3:0] 0x00 = Other IIO
15	Event Data 2	Bit [7:0] Error ID. Refer to the IIO Module Error Codes section in the Intel® Haswell EDS Vol. 1 (section 11.1.7) for details.
16	Event Data 3	Bit [7:5] CPU Number Bit [4:3] Reserved Bit [2:0] ‘b000 = IRP0 Error ‘b001 = IRP1 Error ‘b010 = IIO-Core Error ‘b011 = VT-d Error Other values are reserved

17.1.7BIOS Firmware Volume (FV) Checksum

The firmware volume header checksum is used to verify BIOS binary corruption. The FV header checksum is logged to the SEL using the following format.

Boot block FV SEL entry needs to be done as soon as the IPMI stack is available in PEI. FV Main SEL entry should be done before executing any module from FV main.

The FV header Checksum is a 16-bit checksum of the firmware volume header. A valid header sums to zero.

Table 12 BIOS FV Checksum Error Event Log

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC3 (BIOS FV Checksum)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	IANA Enterprise ID. See Device ID command for details.
11	OEM Data 1	Firmware Volume Type 0x1 – FV Boot block 0x2 – FV Main 0x3 to 0xFF - Reserved
12	OEM Data 2	Checksum – LSB [7:0] - LSB of FV header checksum
13	OEM Data 3	Checksum – MSB [7:0] - MSB of FV header checksum
14	OEM Data 4	Reserved = 0x00
15	OEM Data 5	Reserved = 0x00
16	OEM Data 6	Reserved = 0x00

19.1.9 BIOS Settings change SEL event

BIOS need to log the SEL entry when BIOS settings are changed. BIOS settings can be changed using different methods like listed below.

1. Through the BIOS setup
2. Using OS based utility (ex: SCEWIN)
3. IPMI interface (ex: Set System Boot Options IPMI Command)

Below table specifies the format of the SEL log. BIOS logs the SEL entry after the BIOS settings have applied.

Table 198 BIOS Settings change SEL event

Byte	Field	Description
1:2	Record ID	SEL Record Id.
3	Record Type	OEM System Event Record: 0xC4 (BIOS Settings change)
4:7	Timestamp	Timestamp
8:10	Manufacturer Id	IANA Enterprise ID. See Device ID command for details.
11	OEM Data 1	BIOS settings change method 0x1 – BIOS Setup 0x2 – OS based utility 0x3 – IPMI Interface 0x4 to 0xFF - Reserved
12	OEM Data 2	Reserved = 0x00
13	OEM Data 3	Reserved = 0x00
14	OEM Data 4	Reserved = 0x00
15	OEM Data 5	Reserved = 0x00
16	OEM Data 6	Reserved = 0x00

18 Appendix: Commonly Used of Acronyms

This section provides definitions of acronyms used in the WCS system specifications.

Acronym	Description
ACPI	Advanced Configuration and Power Interface
AHCI	Advanced Host Controller Interface
BIOS	Basic input/output System
BMC	Baseboard Management Controller
CM	Chassis Manager
CMOS	Complementary Metal-Oxide-Semiconductor
CTS	Clear to Send
DDR4	Double Data Rate Type 4
DHCP	dynamic host configuration protocol
DIMM	dual inline memory module
ECC	Error Correcting Code
EEPROM	Electrically Erasable Programmable Read-Only Memory
FRU	Field Replaceable Unit
GPIO	General Purpose Input Output
I2C	Inter Integrated Circuit
IPMI	Intelligent Platform Management Interface
LAN	Local Area Network
LPC	Low Pin Count
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PCH	Platform Control Hub
PDU	Power Distribution Unit
PECI	Platform Environment Control Interface
PNP	Plug and Play
POST	Power-on Self Test
PSU	Power Supply Unit
PXE	Pre-boot Execution Environment
REST	Representational State Transfer
RTS	Ready to Send
RU	Rack Unit
SAS	Serial Attached SCSI
SATA	Serial AT Attachment
SDR	Sensor Data Record
SMBUS	System Management Bus
SMBIOS	System Management BIOS
SPD	Serial Presence Detect
SPI	Serial Peripheral Interface
SSD	Solid State Drive
TDP	Thermal Design Power

TPM	Trusted Platform Module
UART	Universal Asynchronous Receiver/Transmitter
UEFI	Unified Extensible Firmware Interface
WMI	Windows Management Interface