



Intel[®] Management Module

Technical Product Specification



Revision 1.1

October, 2006

Enterprise Platforms and Services Division - Marketing

Revision History

Date	Revision Number	Modifications
March 2005	0.7	Initial release.
April 2005	0.9	Preliminary draft release
June 2005	1.0	Released version
October 2006	1.1	Updated Section 5.6, SNMP

Disclaimers

Information in this document is provided in connection with Intel® products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

This document contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product is available. Verify with your local sales office that you have the latest datasheet before finalizing a design.

The Intel® Management Module may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel, Pentium, Itanium, and Xeon are trademarks or registered trademarks of Intel Corporation.

*Other brands and names may be claimed as the property of others.

Copyright © Intel Corporation 2006. All rights reserved.

Table of Contents

1. Introduction	15
1.1 Chapter Outline.....	15
1.2 Intel® Management Module Use Disclaimer	15
2. Intel® Management Module Overview	16
2.1 Management Module SKU Availability.....	16
2.2 Intel® Management Module Hardware Feature Set	16
3. Board Architecture	19
3.1 Intel® Management Module Installation and Configuration	19
3.2 System Overview of Baseboard Management Controller (BMC).....	19
3.3 Connectors and Jumper Blocks	21
3.3.1 Intel® Management Module Connector	21
3.3.2 Generic Communications Module (GCM) Interface	24
3.3.3 Jumper Block Definition	25
3.4 Physical Dimensions.....	26
3.5 External Interfaces to Intel® Management Module BMC.....	27
3.5.1 Other Interface Support	27
3.6 5V Standby Operation	28
4. Professional Edition Firmware.....	29
4.1 Platform Determination	29
4.1.1 Platform Mismatch Detection.....	29
4.2 Power System.....	30
4.2.1 Power Supply Interface Signals.....	30
4.2.2 Power-up Sequence	31
4.2.3 Power-down Sequence.....	31
4.2.4 Power Control Sources	32
4.2.5 Manual Power Control	33
4.2.6 Power State Retention	34
4.2.7 AC Power Loss	34
4.3 Advanced Configuration and Power Interface (ACPI)	34
4.3.1 ACPI Power Control.....	34
4.3.2 ACPI State Synchronization	35
4.3.3 ACPI S1 Sleep Support	35

4.3.4	ACPI S4 Sleep Support	35
4.3.5	ACPI Power State Notify	35
4.4	System Reset Control	36
4.4.1	Reset Signal Output	36
4.4.2	Reset Control Sources	36
4.4.3	Front Panel System Reset	36
4.4.4	Warm Boot	36
4.4.5	BMC Command to Cause System Reset	37
4.4.6	Watchdog Timer Expiration	37
4.4.7	FRB3 Failure	37
4.5	BMC Reset Control	37
4.5.1	BMC Exits Firmware Update Mode	37
4.6	System Initialization	37
4.6.1	Processor Temperature and Voltage Threshold Setting	37
4.6.2	Fault Resilient Booting (FRB)	38
4.6.3	Boot Control Support	41
4.7	Integrated Front Panel User Interface	41
4.7.1	Chassis ID LED	41
4.7.2	Front Panel / Chassis Inputs	42
4.7.3	Chassis Intrusion	42
4.7.4	Diagnostic Interrupt (Front Panel NMI)	42
4.7.5	CMOS Clearing	43
4.7.6	Secure Mode Operation	43
4.7.7	Set Fault Indication Command	44
4.8	Private Management I ² C Buses	44
4.9	Watchdog Timer	45
4.10	System Event Log (SEL)	45
4.10.1	Servicing Events	45
4.10.2	SEL Entry Deletion	45
4.10.3	SEL Erasure	45
4.10.4	Timestamp Clock	46
4.11	Sensor Data Record (SDR) Repository	46
4.11.1	SDR Repository Erasure	46
4.11.2	Initialization Agent	47
4.12	Field Replaceable Unit (FRU) Inventory Device	47

4.12.1	BMC FRU Inventory Area Format.....	47
4.13	Diagnostics and Beep Code Generation	47
4.14	NMI	48
4.14.1	Signal Generation	48
4.14.2	Signal State Monitoring.....	49
4.15	SMI Generation.....	49
4.16	Processor Sensors	49
4.17	Fan Management.....	50
4.17.1	Nominal Fan Speed	50
4.17.2	Sleep State Fan Control	51
4.17.3	Fan Redundancy Detection	51
4.17.4	Hot Swap Fan Support	52
4.18	Power Unit Management	52
4.18.1	Power Supply Status Sensors	52
4.18.2	Power Gauge/Nozzle.....	53
4.18.3	Power Unit Redundancy	54
4.19	System Memory RAS and Bus Error Monitoring.....	54
4.19.1	SMI Timeout Sensor	54
4.19.2	Memory Sensor	55
4.19.3	Critical Interrupt Sensor	55
4.19.4	Memory Board Sensors	55
4.19.5	DIMM Sensors	55
4.19.6	System Memory Redundancy Monitoring.....	56
4.19.7	Memory Hot Plug	60
4.19.8	BIOS/BMC Interface for Memory RAS.....	60
4.19.9	Single-bit ECC Error Throttling Prevention	61
4.20	Hot Plug PCI Support	62
4.21	Event Logging Disabled Sensor	62
4.22	Event Message Generation and Reception	62
4.23	BMC Self Test.....	62
4.24	BMC and BIOS interaction with Error Logging	64
4.24.1	SMI Handler.....	64
4.24.2	PCI Bus Error.....	64
4.24.3	Processor Bus Error	65
4.24.4	Memory Bus Error.....	65

4.24.5	System Limit Error	65
4.24.6	Processor Failure.....	65
4.25	Light Guided Diagnostics.....	66
4.26	On-line Firmware Update.....	66
4.26.1	Online Update Flow	67
4.26.2	Update Related SEL Logging	68
4.26.3	SDR Update Component of Online Update Process	69
4.27	Messaging Interfaces.....	69
4.27.1	Channel Management	70
4.27.2	User Model	70
4.27.3	Media Bridging.....	70
4.27.4	Request/Response Protocol	70
4.27.5	Native CLI	71
4.27.6	Host to BMC Communication Interface	71
4.27.7	IPMB Communication Interface	73
4.27.8	EMP Interface	76
4.27.9	LAN Interface.....	79
4.27.10	ICMB Interface.....	83
4.28	Event Filtering and Alerting.....	86
4.28.1	Platform Event Filtering (PEF)	87
4.28.2	OEM Action.....	87
4.28.3	Dial Page Alerting	88
4.28.4	Alert over LAN	89
4.28.5	Alert over Serial/PPP	89
5.	Advanced Edition Features	90
5.1	Feature Support Detection.....	90
5.2	Feature Configuration Model	90
5.3	HTTP Server.....	90
5.3.1	Browser Interaction.....	91
5.3.2	Program Interaction	101
5.3.3	Auto Refresh.....	105
5.3.4	Authentication and Encryption	105
5.3.5	Customization and Internationalization.....	106
5.4	SMTP Alerting.....	107
5.4.1	Email Alerting Configuration Parameters.....	108

5.4.2	Initialization and Setup.....	108
5.5	Telnet.....	108
5.5.1	Common CLI.....	109
5.5.2	Native CLI Command Interface	110
5.5.3	Configuration	115
5.5.4	Terminal Mode and CLI	115
5.5.5	Telnet and CCLI.....	115
5.5.6	Web Console and CCLI	116
5.5.7	Scripting Support	116
5.6	SNMP	116
5.6.1	PET Trap Enhancements	117
5.7	Keyboard/Video/Mouse (KVM) Redirection	117
5.7.1	KVM Data Encryption	118
5.7.2	Intel Advanced Remote Server Control	119
6.	Error Reporting and Handling.....	120
6.1	SEL Event Format	120
6.1.1	OEM Event Types.....	121
6.2	Timestamp Format.....	121
6.3	SEL Translation Process	121
6.3.1	Generator ID	121
6.3.2	EvM Rev (Event Message Revision)	122
6.3.3	Sensor Type	122
6.3.4	Sensor Number.....	134
6.3.5	Event Dir/Event Type.....	135
6.3.6	Event Data 1-3.....	138
6.4	BIOS Generated IPMI Events for the Intel® E7520 Chipset.....	141
6.5	SEL Translation Examples	148
6.5.1	Example 1	148
7.	Command Support.....	150
7.1	Command Queuing.....	150
7.2	Power On/Off Issues Related to Commands	150
7.3	BMC Command Tables	150
7.4	Completion Codes	185
7.5	Command Support.....	185
7.6	RMCP+ Command Interface.....	191

8. Environmental Specifications	192
8.1 Environmental Ratings.....	192
8.1.1 Maximum Ratings	192
8.1.2 Maximum Current Requirements	192
8.1.3 BMC Thermal.....	193
8.1.4 DC Specifications	193
8.1.5 LPC Signals (3.3V PCI)	194
8.1.6 FMLBus DC Specifications	195
8.2 AC Specifications.....	195
8.2.1 I ² C Interface	195
8.2.2 LPC Interface.....	196
8.2.3 UART interface	198
Appendix A: OEM SDR Record Formats.....	199
Power Unit Map Record.....	200
System Information Record	201
TControl Fan Speed Control Record	201
Fan Redundancy Map Record.....	203
PC87427 SIO-3 Fan Measurement Configuration Record Format.....	204
Web Server Map Table Size Record Format.....	205
Web Server Map Table Entry Record Format	205
OEM SDR Tag Record Format.....	206
Glossary.....	207

List of Figures

Figure 1. Intel® Management Module Block Diagram	20
Figure 2. Primary Side Dimensions	26
Figure 3. Secondary Side Dimensions.....	27
Figure 4. External Interfaces to Sahalee BMC.....	28
Figure 5. Power Supply Control Signals	30
Figure 6. Power State Transitions.....	33
Figure 7. BMC Online Update Images	67
Figure 8. BMC IPMB Message Reception	75
Figure 9. System Summary Web Page.....	93
Figure 10. System Event Log Web Page.....	94
Figure 11. Power Control Web Page	95
Figure 12. IPMI Commands Web Page	96
Figure 13. IPMI Commands Web Page With IPMI Error.....	97
Figure 14. IPMI Commands Web Page with Invalid Command	97
Figure 15. Web Server Configuration - Top Part	98
Figure 16. Web Server Configuration - Lower Part.....	99
Figure 17. Authentication Prompt	100
Figure 18. Example 'wget' Command-line IPMI Request.....	102
Figure 19. Output Timing Measurement Conditions	196
Figure 20. Input Timing Measurement Conditions	196
Figure 21. LPC Clock Waveform	197

List of Tables

Table 1. Supported Management Features by Tier	17
Table 2. IMM Connector Pin-out (J1L1).....	21
Table 3. Generic Communications Module (GCM) Interface	25
Table 4. Flash Jumper (J1B1).....	25
Table 5. Supported Product IDs.....	29
Table 6. Power Control Initiators.....	32
Table 7. Power State Transitions	33
Table 8. ACPI Power States	34
Table 9. System Reset Sources and Actions.....	36
Table 10. BMC Reset Sources and Actions.....	37
Table 11. Requirements for Processor Status	38
Table 12. Chassis ID LED Indicator States.....	41
Table 13. Secure Mode vs. ACPI State	44
Table 14. BMC Beep Codes	48
Table 15. Processor Sensors.....	49
Table 16. BMC Self Test Results	63
Table 17. Online Update SEL Events	68
Table 18. Standard Channel Assignments	70
Table 19. KCS Interface Addresses.....	71
Table 20. SMS/SMM Status Register Bits	72
Table 21. BMC IPMB LUN Routing.....	75
Table 22. Supported RMCP+ Cipher Suites	80
Table 23. Supported RMCP+ Payload Types	81
Table 24. ICMB Bridge Request Response Data Packet Format	85
Table 25. IPMB Bridge Request Response Data Format	85
Table 26. LUN Table.....	86
Table 27. OEM PEF Action Priorities	87
Table 28. OEM Event Filter Table.....	88
Table 29. Initial Pages	91
Table 30. User Access Levels	100
Table 31. "Onload" Function Arguments.....	103
Table 32. webCmdPage Sub-commands	104

Table 33. Native CLI Command Set	110
Table 34. Input and Output Sequence	116
Table 35. SEL Event Records.....	120
Table 36. BMC Generator IDs	121
Table 37. Sensor Type Codes	122
Table 38. Event/Reading Type Code Ranges	136
Table 39. Event Request Message Event Data Field Contents	137
Table 40. Generic Event/Reading Type Codes	139
Table 41. BIOS Generated IPMI Events	141
Table 42. Memory Error Events	142
Table 43. Examples of Event Data Field Contents for Memory Errors	143
Table 44. PCI Error Events	143
Table 45. Examples of Event Data Field Contents for PCI Errors	144
Table 46. FRB2 Error Events.....	144
Table 47. Examples of Event Data Field Contents for FRB2 Errors	144
Table 48. MCH Hub Interface Error Events	145
Table 49. Examples of Event Data Field Contents for MCH Hub Interface Errors.....	145
Table 50. System Bus Error Events.....	146
Table 51. Examples of Event Data Field Contents for System Bus Errors	146
Table 52. Memory Buffer Error Events	147
Table 53. Examples of Event Data Field Contents for Memory Buffer Errors.....	147
Table 54. BMC Intelligent Platform Management Interface (IPMI) Commands	151
Table 55. BMC Watchdog Timer Commands	152
Table 56. BMC IPMI Messaging Support Commands	152
Table 57. BMC Chassis Commands	154
Table 58. Chassis Control Command Results	155
Table 59. Boot Control Commands.....	155
Table 60. Supported Boot Option Parameters	156
Table 61. BMC Event Receiver Device Commands	156
Table 62. PEF Commands.....	156
Table 63. Supported PEF Configuration Parameters	157
Table 64. BMC Sensor Device Commands	158
Table 65. BMC FRU Inventory Device Commands.....	158
Table 66. BMC SDR Repository Device Commands.....	159
Table 67. BMC SEL Device Commands	159

Table 68. LAN Commands.....	160
Table 69. Supported LAN Configuration Parameters	160
Table 70. EMP Commands.....	162
Table 71. Supported Serial/Modem Configuration Parameters	162
Table 72. Terminal Mode Commands.....	164
Table 73. Intel OEM Terminal Mode Commands.....	164
Table 74. Intel General Application Commands	165
Table 75. Intel® Platform-specific Commands.....	173
Table 76. Platform Information Parameters	174
Table 77. Supported SMTP Configuration Parameters	175
Table 78. Supported HTTP Configuration Parameters	176
Table 79. Supported HTTP Channel Configuration Parameters.....	177
Table 80. Supported SNMP Configuration Parameters	178
Table 81. Supported Telnet Configuration Parameters	179
Table 82. Supported KVM Configuration Parameters.....	180
Table 83. Supported User Feature Configuration Parameters	181
Table 84. SOL 2.0 Commands	182
Table 85. Supported SOL 2.0 Parameters.....	182
Table 86. Intel OEM Terminal Mode Text Commands.....	182
Table 87. ICMB Bridge Device Commands	184
Table 88. Completion Codes	185
Table 89. Application Command Support Matrix	186
Table 90. Chassis Command Support Matrix	187
Table 91. Sensor/Event Command Support Matrix.....	187
Table 92. Storage Command Support Matrix	188
Table 93. Transport Command Support Matrix.....	189
Table 94. Intel General Application Command Support Matrix.....	189
Table 95. Intel Advanced Feature Command Support Matrix.....	190
Table 96. Intel Platform Specific Command Support Matrix	190
Table 97. BMC RSSP/RAKP Messages	191
Table 98. OEM Payload Types	191
Table 99. Absolute Maximum Ratings	192
Table 100. DC Specifications for 5V Tolerant Signals.....	194
Table 101. DC Specifications for LPC Signals (3.3 V).....	194
Table 102. FML Bus DC Specifications	195

Table 103. LPC Clock Specifications 197

Table 104. UART Specifications 198

Table 105. Intel OEM SDR Subtypes 199

Table 106. Power Unit Map SDR Record Format..... 200

Table 107. Example Power Unit Map SDR Record for a 2+1 Power Subsystem 200

Table 108. System Information SDR Record Format..... 201

Table 109. Temperature Fan Speed Control SDR Record Format..... 202

Table 110. Stepwise Linear Type Temperature Sensor Sub-record..... 202

Table 111. Clamp Type Temperature Sensor Sub-record..... 203

Table 112. Fan Redundancy Map SDR Record Format 203

Table 113. PC87427 SIO-3 Configuration SDR Record Format..... 204

Table 114. Map Table Size Record Format 205

Table 115. Map Table Entry Record Format..... 205

Table 116. OEMSDR Tag Record Format 206

This page intentionally left blank

1. Introduction

This Technical Product Specification (TPS) provides details about the architecture and feature set of the Intel® Management Module (IMM). This is a technical document meant to assist people with understanding and learning more about the specific features of the board. The target audience for this document is anyone wishing to obtain more in depth detail of the Intel® Management Module than what is made available in the Users Guide.

This is one of several technical documents available that describe management on Intel® Management Module-compatible server boards. All of the functional sub-systems that make up the module are described in this document. However, some low-level detail of specific sub-systems is not included.

Each of the server boards listed above has a board-specific Technical Product Specification. These documents contain additional technical information about the specific server platforms.

1.1 Chapter Outline

This document is divided into the following chapters

- Chapter 1 – Introduction
- Chapter 2 – Intel® Management Module Overview
- Chapter 3 – Board Architecture
- Chapter 4 – Professional Edition Firmware
- Chapter 5 – Advanced Edition Firmware
- Chapter 6 – Error Reporting and Handling
- Chapter 7 – Command Support
- Chapter 8 – Environmental Specifications
- Appendix A – Intel OEM Sensor Data Record Formats

1.2 Intel® Management Module Use Disclaimer

Intel Corporation server building blocks are components that need adequate airflow to cool. Intel ensures through its own chassis development and testing that when Intel server building blocks are used together, the fully integrated system will meet the intended thermal requirements of these components. It is the responsibility of the system integrator who chooses not to use Intel developed server building blocks to consult vendor datasheets and operating parameters to determine the amount of air flow required for their specific application and environmental conditions. Intel Corporation cannot be held responsible, if components fail or the server board does not operate correctly when used outside any of their published operating or non-operating limits.

2. Intel® Management Module Overview

The Intel® Management Module is a 2.25-inch by 2.75-inch printed circuit board that, when installed in the Flexible Management Connector (FMC) on an EPSD server board, provides an increased level of manageability over the basic server management available to the server board. The Intel® Management Module includes an integrated Baseboard Management Controller (BMC) that functionally replaces the onboard instrumentation included on the server board.

2.1 Management Module SKU Availability

In this document, the name Intel® Management Module is used to describe the family of boards that will be made available under a common product name. The core features for the boards are the same; however each board will have the following distinctions:

- Intel® Management Module – Professional Edition
- Intel® Management Module – Advanced Edition

Throughout this document, all references to the Intel® Management Module or BMC will refer to the core features of the board. Features that are specific to one distinct SKU will be noted in the appropriate sections describing the individual SKUs. The board you select to use may or may not have all the features described based on the listed board differences.

2.2 Intel® Management Module Hardware Feature Set

- IPMI 2.0 based Baseboard Management Controller (BMC)
- Professional Edition
 - 64Kx16 SRAM
 - 4Mx16 Flash memory
- Advanced Edition
 - 256Kx16 SRAM
 - 4Mx16 Flash memory
 - Digital Visual Interface (DVI)
 - KVM FPGA
 - Generic Communication Module (GCM) management LAN connector

Table 1 summarizes the supported features for each management level. The onboard platform instrumentation is included as a reference and is the base level of a server platform that does not have an Intel® Management Module installed.

Table 1. Supported Management Features by Tier

Element	Onboard Platform Instrumentation	Intel® Management Module - Professional Edition	Intel® Management Module - Advanced Edition
IPMI Messaging, Commands, and Abstractions	Yes	Yes	Yes
Baseboard Management Controller (BMC)	Yes	Yes	Yes
Sensors	Limited	Yes	Yes
Sensor Data Records (SDRs) and SDR Repository	Limited	Yes	Yes
FRU Information	Limited	Yes	Yes
Autonomous Event Logging	Yes	Yes	Yes
System Event Log (SEL)	92 entries	3276 entries	3276 entries
BMC Watchdog Timer, covering BIOS and run-time software	Limited	Yes	Yes
IPMI Channels, and Sessions	Limited	Yes	Yes
EMP (Emergency Management Port) - IPMI Messaging over Serial/Modem. This feature is also referred to as DPC (Direct Platform Control) over serial/modem.	No	Yes	Yes
Serial/Modem Paging	No	Yes	Yes
Serial/Modem Alerting over PPP using the Platform Event Trap (PET) format	No	Yes	Yes
DPC (Direct Platform Control) - IPMI Messaging over LAN (available via both onboard network controllers)	Yes	Yes	Yes
LAN Alerting using PET	Yes	Yes	Yes
Platform Event Filtering (PEF)	Yes	Yes	Yes
ICMB (Intelligent Chassis Management Bus) - IPMI Messaging between chassis	No	Yes	Yes
PCI SMBus support	No	Yes	Yes
Fault Resilient Booting	Limited	Yes	Yes
BIOS logging of POST progress and POST errors	Errors Only	Yes	Yes
Integration with BIOS console redirection via IPMI v2.0 Serial Port Sharing	No	Yes	Yes
Access via web browser	No	No	Yes
SNMP access	No	No	No
Telnet access	No	No	Yes
DNS support	No	No	Yes
DHCP support (dedicated NIC only)	No	No	Yes
Memory Sparing/Mirroring sensor support	No	Yes	Yes
Alerting via email	No	No	Yes
Keyboard, Video, Mouse (KVM) redirection via LAN	No	No	Yes
High speed access to dedicated NIC	No	No	Yes

This document will provide an overview of the Intel® Management Module architecture and details of its features and functionality including BIOS interactions and support. See the Technical Product Specification for the supported Intel® Server Board for a detailed description of the server board features, and functionality and details about the onboard platform instrumentation.

3. Board Architecture

This section specifies the hardware aspects of the Intel® Management Module.

3.1 Intel® Management Module Installation and Configuration

A newly installed Intel® Management Module may be either new from the factory and installed for the first time, or it may have been moved from one server platform to another. The Baseboard Management Controller (BMC) cannot tell which board it is on. Therefore, when an Intel® Management Module is first installed into a system, the BMC operational code and Sensor Data Records (SDRs) must be updated for normal operation. This must be done whether the module is new from the factory, or was moved from another system.

Installation of the Intel® Management Module is detailed in the *Intel® Management Module Installation and User's Guide*. This document is available on the Intel® Server Deployment Toolkit that came with your server board.

3.2 System Overview of Baseboard Management Controller (BMC)

The Intel® Management Module BMC is an Application Specific Integrated Circuit (ASIC) with a Reduced Instruction Set Computer (RISC)-based processor and many peripheral devices embedded. It is designed to be the central server management controller in an enterprise server system. The BMC contains the logic needed for executing the firmware, controlling the system, monitoring sensors, and communicating with other systems and devices via various external interfaces.

The BMC controls various server management functions, such as the system power/reset control, a variety of types of sensor monitoring, system initialization, fault resilient booting (FRB), etc.

The memory subsystem consists of flash memory to hold the BMC operation code, firmware update code, System Event Log (SEL), and a Sensor Data Record (SDR) repository and BMC persistent data. RAM is used for data and occasionally code for flash programming and a EEPROM is used for the field replacement unit (FRU) inventory information and the event filter table.

The front panel interface to the BMC has the standard user controls for power, reset, diagnostic interrupt (Front Panel NMI), and an optional sleep button, in addition to indicators for working conditions and faults. The BMC also provides an interface for an Intel® Local Control Panel (LCP), which is a liquid crystal display (LCD) that can provide a textual user interface to BMC functions. The LCP is not directly controlled by the BMC, but interfaces to the BMC as a satellite IPMB controller.

A block diagram for the Intel® Management Module is on the following page.

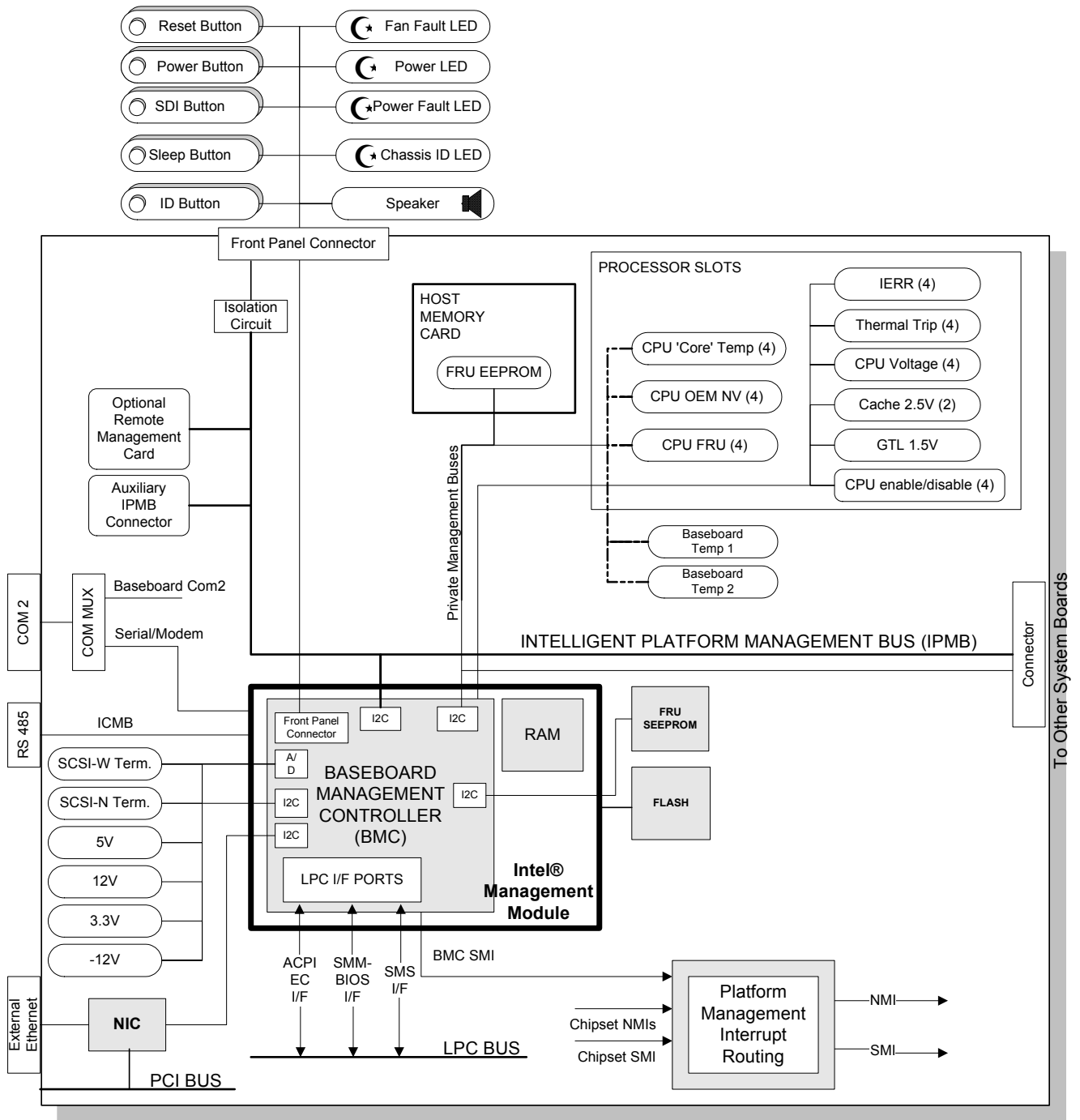


Figure 1. Intel® Management Module Block Diagram

3.3 Connectors and Jumper Blocks

The Intel® Management Module connects to the management buses built into the server board via the Flexible Management Connector (FMC). The Intel® Management Module - Advanced Edition has an additional connector that attaches to the Generic Communication Module (GCM). The tables below provide the pin-outs for each connector.

One jumper block is provided on the Intel® Management Module. Jumper J1B1 to write-protects the boot block of the Intel® Management Module BMC firmware. The BMC firmware boot block should be write-protected during normal operation and should only be left unprotected when a boot block update is required. See the release notes for each BMC release to determine if a boot block update is required.

3.3.1 Intel® Management Module Connector

Each Intel® Server Board or Platform that supports the Intel® Management Module provides a 120-pin connector (J1L1) to connect the module to the server board. The connector pin-out is provided in the following table.

Table 2. IMM Connector Pin-out (J1L1)

FMC Signal Name	FMC Pin	Description
DVI_TX1M	2	Green TMDS differential DVI output of graphics chip
DVI_TX0M	3	Blue TMDS differential DVI output of graphics chip
DVI_TX1P	4	Green TMDS differential DVI output of graphics chip
DVI_TX0P	5	Blue TMDS differential DVI output of graphics chip
DVI_CLK_TX1CM	8	TMDS differential DVI clock output of graphics chip
DVI_TX2M	9	Red TMDS differential DVI output of graphics chip
DVI_CLK_TX1CP	10	TMDS differential DVI clock output of graphics chip
DVI_TX2P	11	Red TMDS differential DVI output of graphics chip
SIO_MS_DAT	14	KVM mouse data from SIO
SIO_KB_DAT	15	KVM keyboard data from SIO
SIO_MS_CLK	16	KVM mouse clock from SIO
SIO_KB_CLK	17	KVM keyboard clock from SIO
PS2_MS_DAT	18	KVM passthrough mouse data from PS2 connector
PS2_KB_DAT	19	KVM passthrough keyboard data from PS2 connector
PS2_MS_CLK	20	KVM passthrough mouse clock from PS2 connector
PS2_KB_CLK	21	KVM passthrough keyboard clock from PS2 connector
KM_INHIB_N	22	KVM enable of server board Switch for mouse and keyboard
FML_SDA	25	Fast Management Link Data In. This signal is driven by the FML slave, i.e. NIC controller
FML_MCL_I2CSCL	26	Fast Management Link Clock Out. This signal is driven by the FML master, i.e. FMM. When not configured as FML, this signal is used as I ² C clock.

FMC Signal Name	FMC Pin	Description
FML_SINTEX	27	Fast Management Link Slave Interrupt/Clock Extension. This signal is driven by the FML slave, and has a dual usage: Used as an alert signal for the slave to notify master that data is ready to be read from slave Used as a clock extension (stretching) for the slave to indicate to the master to extend its low period of the clock
FML_MDA_I2CSDA	28	Fast Management Link Data Out. This signal is driven by the FML Master. When not configured as FML, this signal is used as I ² C data
ICH_LCLK	31	LPC 33Mhz clock input
USB_M	32	Reserved for future use as USB input. Server board can leave as NC
FMM_SYSIQ	33	KCS interrupt signal from FMM card.
USB_P	34	Reserved for future use as USB input. Server board can leave as NC
ICH_LAD1	35	LPC address/data bus Bit 1
FMM_RSMRST_N	36	When this signal is asserted, the FMM is held in reset. This is a standby reset indication, and should be driven by a standby monitor device such as the Heceta7 or Dallas Semiconductor* DS1815
ICH_LFRAME_N	37	LPC Cycle Framing
ICH_LAD0	38	LPC address/data bus Bit 0
ICH_LAD3	39	LPC address/data bus Bit 3
ICH_LPCPD_N	40	LPC power down indication
ICH_LAD2	41	LPC address/data bus Bit 2
FMM_LPCRST_N	40	LPC bus reset. Must be properly buffered on serverboard to ensure monotonicity
DFP_CLK	46	Serial clock signal for DFP EDID device. Must connect to DFP_CLK pin on the graphics chip.
DFP_DAT	48	Serial data signal for DFP EDID device. Must connect to DFP_DAT pin on the graphics chip.
IPMB_I2C_5VSB_SDA	49	Connects to IPMB header
IPMB_I2C_5VSB_SCL	50	Connects to IPMB header
SMB_I2C_3VSB_SDA	51	This bus should connect to the PCI slots, ICH, and mBMC (host I/F). An isolated version of this bus (non-standby) should connect to the DIMMs, and clock buffer(s)
SMB_I2C_3VSB_SCL	52	This bus should connect to the PCI slots, ICH, and mBMC (host I/F). An isolated version of this bus (non-standby) should connect to the DIMMs, and clock buffer(s)
PERIPH_I2C_3VSB_SDA	53	This bus should connect to the mBMC (Peripheral I/F), SIO, Heceta, Front panel header. A level shifted version of this bus (5V standby) should connect to the power supply header
PERIPH_I2C_3VSB_SCL	54	This bus should connect to the mBMC (Peripheral I/F), SIO, Heceta, Front panel header. A level shifted version of this bus (5V standby) should connect to the power supply header
MCH_I2C_3V_SDA	55	This bus should connect to the Northbridge and I/O bridge (MCH and PXH respectively in the Intel E7520 chipset). In a system that supports PCI Hot Plug, this bus should also connect to the power control devices if possible (such as the MIC2591 for PCI-Express for example)
MCH_I2C_3V_SCL	56	This bus should connect to the Northbridge and I/O bridge (MCH and PXH respectively in the Intel E7520 chipset). In a system that supports PCI Hot Plug, this bus should also connect to the power control devices if possible (such as the MIC2591 for PCI-Express for example)
LAN_I2C_3VSB_SDA	57	LAN usage

FMC Signal Name	FMC Pin	Description
LAN_I2C_3VSB_SCL	58	LAN usage
HDD_FLT_LED_N	64	Drive Fault LED output driven when FMM detects a bad drive from the Hot Swap controller on the Hot Swap disk Drive sub-system.
FMM_PS_PWR_ON_N	65	Power On Request to the system power supply
COOL_FLT_LED_N	66	Cool Fault LED output driven when FMM detects a bad Fan if SSI front panel is detect.
FMM_CPU_VRD_EN	67	This signal is driven by the FMM to enable the CPU VRDs and allow the VRD power good chain to complete. This signal can also be used to keep the system in reset for an extended time, beyond what the chipset RST_BTN_N can provide.
FMM_SCI_N	68	SCI event request. If ACPI EC is supported by FMM, this signal is used for ACPI interrupts.
ICH_PWR_BTN_N	69	FMM pass through of Front panel power button to chipset
FMM_SPKR_N	72	FMM uses this to create Beep Codes on the system audible alarm. This signal is configured as an Open Drain buffer in the FMM and must be pulled up to 3.3V standby on the motherboard
FP_NMI_BTN_N	73	NMI / Diagnostic interrupt from front panel. Actual NMI generated by SMBUS command to mBMC
FP_SLP_BTN_N	74	Front panel Sleep Button input, if used
FP_ID_BTN_N	75	Front panel ID button, will cause the ID light to toggle
SYS_PWR_GD	76	Signal from the end of the server board VRD power good chain. This signal should be the last VRD power good indication generated on the server board. Usually this would be the signal feeding the Chipset Power OK input. Used by FMM in conjunction with RST_PWRGD_PS to determine if all critical VRDs have successfully reached their nominal value.
CPU2_SKTOCC_N	80	Indicates that a Processor is in the application processor socket
CLK_32K_RTC	81	This signal is used for "Synchronized clock with system RTC". FMM can synchronize own RTC with system RTC. IPMI define synchronized method. clock comes from the Chipset RTC function.
CPU1_SKTOCC_N	82	Indicates that a Processor is in the primary processor socket. If this socket is detected empty and there's an attempt to power up the system, the FMM will output an Error Beep Code and prevent the System from turning on
FMM_SOUT	85	EMP/SOL Serial Data Out. This is the Serial Port data output from FMM and should be connected to the SIN signal in the SIO3 device
FMM_SIN	86	EMP/SOL Serial Data In. This is the Serial Port data input into the FMM and should be connected to the SOUT signal in the SIO3 device
FMM_DCD_N	87	EMP/SOL Data Carrier Detect. This is the Serial Port Data Carrier Detect input into the FMM and should be connected to the DCD signal in the SIO3 device
FMM_RTS_N	88	EMP/SOL Request to Send. This is the Serial Port Request to Send output from FMM and should be connected to the CTS (Clear to Send) signal in the SIO3 device
FMM_DTR_N	89	EMP/SOL Data Terminal Ready. This is the Serial Port Data Terminal Ready output from FMM and should be connected to the DSR (Data Set Ready) signal in the SIO3 device
FMM_CTS_N	90	EMP/SOL Clear to Send. This is the Serial Port Clear to Send input into the FMM and should be connected to the RTS (Ready to Send) signal in the SIO device
ICMB_RX	93	Inter Chassis Communication Management Bus receive data
ICMB_TX	94	Inter Chassis Communication Management Bus transmit data
ICMB_TX_EN	96	Inter Chassis Communication Management Bus transceiver enable

FMC Signal Name	FMC Pin	Description
FMM_RI_BUF_N	97	Ring Indicator from the EMP serial port on the server board
RST_PWRGD_PS	101	Power good signal from power subsystem. In typical system, this signal is connected to PWR_OK signal on power supply. This signal is monitored by the FMM to detect a power supply failure
LAN_SMBALERT_N	102	Alert signal from the motherboard NIC (LOM).
ICH_SLP_S4_N	103	Power Off request from the chipset
ICH_SMI_BUFF_N	105	SMI signal from chipset. This signal is monitored by the FMM to detect an "SMI Time-out" condition. If this signal is asserted for longer than a predefined SMI Time-out timer, an event is logged and the FMM interrogates the chipset for further data, such as fatal errors.
CHPSET_ERR_ALERT_N	106	When available from chipset, indicates that a error occurred and FMM will need interrogate chipset for further data, such as fatal errors. If not available, leave as NC.
FP_RST_BTN_N	109	Front panel Reset Button input.
ICH_RST_BTN_N	110	Passthrough of front panel Reset button to the chipset. FMM chassis control command will also use this.
FP_PWR_BTN_N	113	Front panel power button input.
FMM_IRQ_SMI_N	116	FMM might use this signal to generate an SMI to the system.
FMM_PRES_N	120	When FMM is present, this signal is asserted. This signal can be used to notify the BIOS that a module is present (via routing to GPIO), as well as to control any logic which behaves differently when FMM is present, such as the FML mux (if supported), etc

The _N or _L symbol appended to a signal name indicates the signal is active low ("Not").

FMC Signal Name	FMC Pin	Description
3.3V VAUX	23, 24, 29, 30	3.3V power rail provided by the server board, when power is on this rail pulls current from the system 3.3V power rail. When power is off this rail pulls current from 3.3V S/B generated on the motherboard from the system 5V SB power rail.
3.3V SB	61, 62, 79, 71, 107, 108, 119	This rail pulls current from the 3.3V S/B regulator on the motherboard that has the system 5V SB power rail as its input.
5V SB	83, 84	This rail pulls current from the system 5V S/B power rail.
GND	1, 6, 7, 12, 13, 43, 44, 91, 92, 99, 100	System electrical ground

3.3.2 Generic Communications Module (GCM) Interface

This is the interface used to connect the Intel® Management Module – Advanced Edition to the Generic Communications Module (GCM). The GCM contains a LAN interface chip that is dedicated to the Intel® Management Module – Advanced Edition as the LAN based management interface for all Advanced features. The GCM is mounted on a PCI expansion slot cover and can be put in any empty PCI slot in an Intel pedestal chassis.

Note: The GCM does not actually occupy a PCI slot on the server board, it connects to the chassis in an empty PCI expansion slot. When mounted in an Intel rack chassis, the GCM

connects directly to the chassis in a reserved location as described in the *Intel® Management Module Installation and Users Guide*.

The communication path between the GCM and Intel® Management Module is via a cable that plugs into a 5-pin header (J2M1) on each of the boards. The pin out of the connector is shown below.

Table 3. Generic Communications Module (GCM) Interface

Pin #	GMC Signal	Description
1	GCM_SDA	SM Bus Serial Data
2	GND	Ground
3	GCM_SCL	SM Bus Clock
4	3.3V SB	3.3V S/B power
5	GCM_SMBAlert	SM bus Alert

3.3.3 Jumper Block Definition

Table 4. Flash Jumper (J1B1)

1 - 2	Unprotect
2 - 3	Protect (Default)

3.4 Physical Dimensions

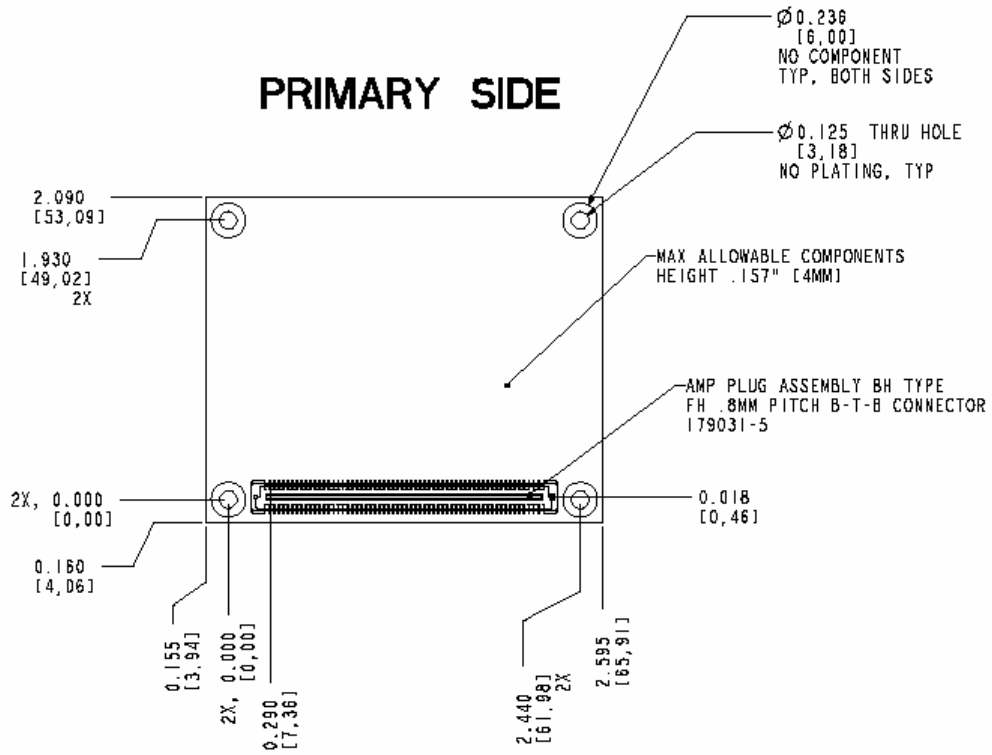


Figure 2. Primary Side Dimensions

SECONDARY SIDE

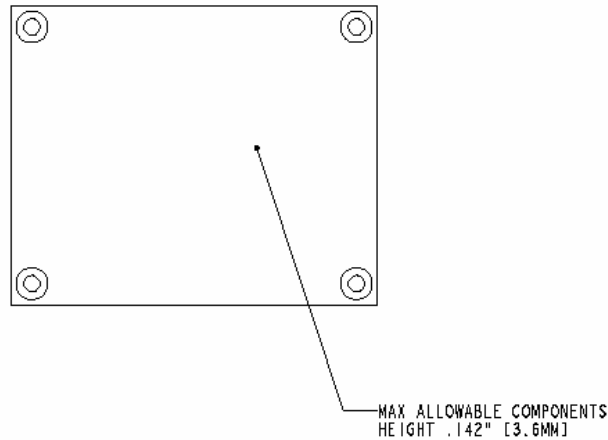


Figure 3. Secondary Side Dimensions

3.5 External Interfaces to Intel® Management Module BMC

The external world interfaces, namely the host system, LAN protocol, serial/modem interface on the Emergency Management Port (EMP), and the system SCSI, interact with the BMC through the corresponding interface modules as shown in Figure 2.

Also built into the BMC are the power supply control functions and front panel control functions. The BMC communicates with the internal modules using its private I²C bus or the IPMB bus. The host interacts with the IMM BMC through the KCS/LPC interface. External devices interact with the BMC using the IPMB, LAN and EMP interfaces to communicate with the internal modules using a private I²C bus or the IPMB bus.

3.5.1 Other Interface Support

Other IPMI 2.0 based interfaces outlined in section 4.27.

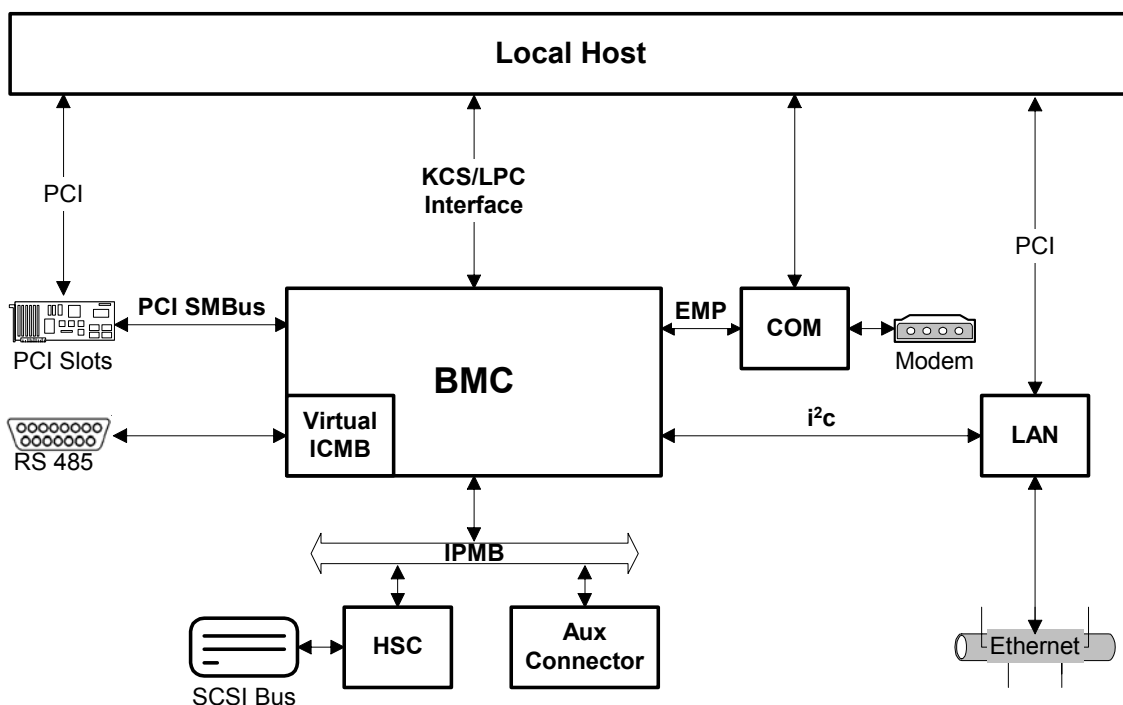


Figure 4. External Interfaces to Sahalee BMC

3.6 5V Standby Operation

The Intel® Management Module operates on 5V standby power. 5V standby is a low power 5V supply that is active whenever the system is plugged into AC power. Certain other devices on the server board also operate on 5V standby power to provide complete management functionality. 5V standby is used by the following onboard management devices:

- Intel® Management Module BMC and associated RAM, Flash, and SEEPROM
- Onboard NICs that support IPMI-over-LAN and LAN Alerting, Wake-On LAN, and Magic Packet* operation.
- Emergency management port
- IPMB
- PCI SMBus
- ICMB transceiver card (if present)
- IPMB isolation circuit
- System Status LED on the front control panel
- System Identify LED

4. Professional Edition Firmware

This chapter provides a high-level description of the functionality associated with the architectural blocks that make up the Intel® Management Module. It also includes information on the IPMI command support and functionality of the Intel® Management Module BMC.

4.1 Platform Determination

The BMC provides an indication, via the *Get Device ID* IPMI command, of the type of platform that it manages. Each platform/server board type has a unique product ID as described in the table below.

Table 5. Supported Product IDs

Product ID Bytes		Product Description
MS Byte	LS Byte	Name
0x00	0x21	Intel® Server Board SE7520AF2
0x00	0x22	Intel® Server Board SE7520JR2
0x00	0x23	Intel® Server Board SE7520BD2
0x01	0x04	Intel® Server Board SRHW4

A newly installed Intel® Management Module BMC may not be able to automatically determine the platform type. In this case, the BMC will return the platform ID 0000h indicating an unspecific/unknown platform. The BMC boot block cannot determine the platform type and will always return product ID 0000h.

4.1.1 Platform Mismatch Detection

Early in POST, the BIOS will communicate a unique numeric platform ID code to the BMC. The BMC will see if the firmware matches the indicated platform type. If a mismatch occurs, the BMC will log a self test error code of C5h, 00h and log an event to the SEL. The sensor type of this event will be 2Bh, version change, offset 03h, “firmware incompatibility detected”.

Near the end of POST, the BIOS will look for a platform mismatch. If a mismatch is indicated, BIOS will display a warning message on the monitor.

4.2 Power System

The following sections discuss the BMC power control functions.

4.2.1 Power Supply Interface Signals

The BMC supports two power supply control signals: *Power On* and *Power Good*. The *Power On* signal connects to the chassis power subsystem and is used to request power state changes (asserted = request *Power On*). *Power Good* is a signal from the chassis power subsystem indicating current power state (asserted = power is on).

Figure 3 shows the power supply control signals and their sources. To turn the system on, the BMC asserts the *Power On* signal and waits for the *Power Good* signal to assert in response, indicating that DC power is on.

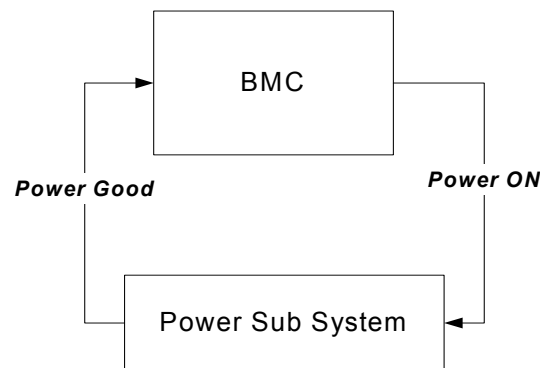


Figure 5. Power Supply Control Signals

The BMC uses the *Power Good* signal to monitor whether the power supply is on and operational, and to confirm whether the actual system power state matches the intended system on/off power state that was commanded with the *Power On* signal.

De-assertion of the *Power Good* signal generates an interrupt, which the BMC uses to detect either power subsystem failure or loss of AC power. If AC power is suddenly lost, the BMC:

1. Immediately asserts system reset.
2. Powers down the system.
3. Waits for configured system off time, then attempts to power the system back up (depending on configuration).

The BMC generally responds to the power loss interrupt within 1 ms, with the following exceptions:

- If the BMC is in Firmware Transfer mode, there will be no response until this mode is exited.
- If the BMC is writing an event message or SDR record, the response could take as long as 2 ms.

If the BMC asserts *Power On* but does not detect a *Power Good* assertion within the configured time (nominally 6 seconds), the BMC asserts a power fault condition. The *Power Good* signal should normally be asserted within 1.5 seconds in a system environment. The *Power Good* signal must remain stable and cannot glitch when being asserted.

If the BMC de-asserts *Power On* but power remains on, the BMC also asserts a power fault condition.

When system power is on, and a falling edge of the *Power Good* signal is detected, the BMC asserts a power fault condition. This can occur when the *Power Good* signal is de-asserted, or glitches, without the BMC de-asserting the *Power On* signal.

4.2.2 Power-up Sequence

When turning on the system power due to one of the event occurrences listed in Table 6, the BMC executes the following procedure:

1. The BMC asserts *Power On* and waits for the power subsystem to assert *Power Good*. The system is held in reset.
2. The BMC sends a *Set ACPI Power State* command indicating an S0 state to all management controllers whose SDR management device records indicate that they should receive the notification.
3. The BMC initializes all *sensors* to their *Power On* initialization state. The Init Agent is run.
4. The BMC attempts to boot the system by running the FRB algorithm. See section 4.6.2.

4.2.3 Power-down Sequence

To power down the server, the BMC effectively performs the sequence of power-up steps in reverse order. This occurs as follows:

1. The BMC asserts system reset (de-asserts *Power Good*).
2. If enabled, the BMC sends a *Set ACPI Power State* command indicating an S5 state to all management controllers whose SDR management device records indicate that they should receive the notification.
3. The BMC de-asserts the *Power On* signal.
4. The power subsystem turns off system power upon de-assertion of the *Power On* signal.

4.2.4 Power Control Sources

The sources listed in Table 6 can initiate power-up and/or power-down activity.

Table 6. Power Control Initiators

#	Source	External Signal Name or Internal Subsystem	Capabilities
1	Power Button	Front control panel power button	Turns power ON or OFF
2	BMC Watchdog Timer	Internal BMC timer	Turns power OFF, or power cycle
3	Platform Event Filtering	PEF	Turns power OFF, or power cycle
4	Command	Routed through command processor	Turns power ON or OFF, or power cycle
5	Power state retention	Implemented via BMC internal logic	Turns power ON when AC power returns
6	Chipset	Sleep S5	Turns power ON or OFF

4.2.4.1 Power Button

The *Power Button* signal is used to toggle system power. The *Power Button* signal to the BMC is activated by a momentary contact switch on the front panel assembly. The BMC de-bounces the signal. The de-bounce time is 50 ms (the signal must be in a constant state for 50 ms before it is treated as asserted). After de-bouncing the signal, the BMC routes it directly to the chipset via the *Chipset Power Button* signal. The chipset responds to the assertion of the signal. In other words, it reacts to the press of the switch, not the release. An exception to the above is when Secure Mode is enabled and active. See Section 4.7.6 for details on Secure Mode.

Note: If the BMC detects corrupted operational code when AC power is applied, then the BMC will automatically power the system up. While in firmware update mode the BMC does not respond to the *Power Button* so system power cannot be turned on or off when the BMC is in its firmware update mode.

In the case of simultaneous power and sleep button presses, the *Power Button* action takes priority. For example, if the sleep button is depressed for one second, then the *Power Button* is pressed and released, the system will power down. Due to the routing of the de-bounced *Power Button* signal to the chipset, the power signal action overrides the action of the other switch signals.

4.2.4.2 Chipset Sleep S5

The BMC expects a chipset signal *Sleep S5* to provide power state change requests. *Sleep S5*, when asserted, is taken as a power-off request and when de-asserted, is taken as a power-on request. The BMC requires *Sleep S5* to maintain its level for at least 15 ms to be recognized. This signal can change state as a result of the following events:

- Operating system request
- Real Time Clock alarm
- Chipset *Power Button* request response

4.2.4.3 Power State Retention

The BMC provides the ability to control the AC power-on behavior of the system. See section 4.2.2 for details.

4.2.5 Manual Power Control

Figure 6 shows the state of power control transitions from the user's point of view. There are two ACPI modes of power control operations, a one-button mode and a two-button mode. The one-button mode is used for platforms that do not implement a sleep button. Table 7 summarizes the possible transitions for 1- and 2-button modes.

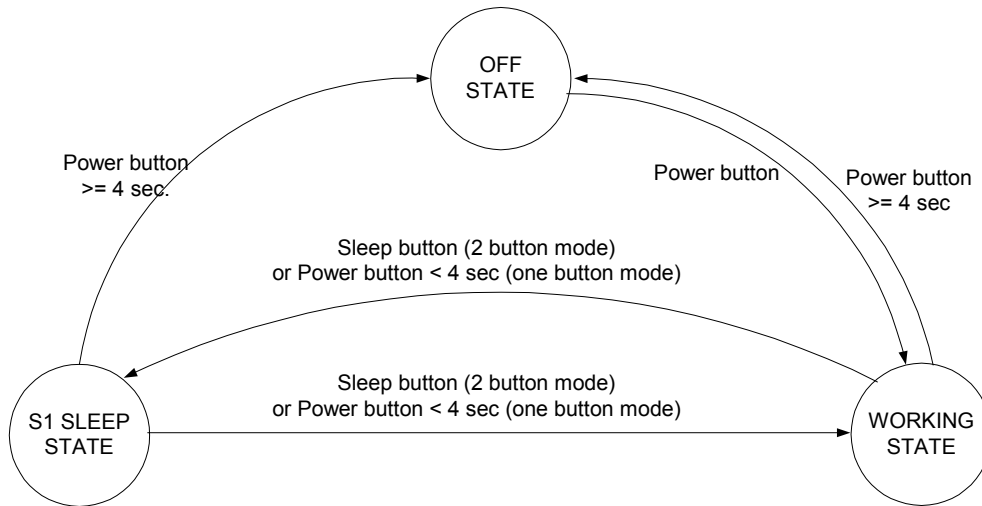


Figure 6. Power State Transitions

Table 7. Power State Transitions

From State	To State	Action Required	Mode
Off	Working	Press power button for at least 50 ms. The chipset should respond by deasserting the Sleep S5 signal.	1 and 2 button
Working	Off	Press power button for more than 4 seconds.	1 and 2 button
Working	Sleep	Press sleep button for at least 50 milliseconds or Press power button.	2 button 1 button
Sleep	Working	Press sleep button for at least 50 milliseconds or Press power button.	2 button 1 button
Sleep	Off	Press power button.	2 button

4.2.6 Power State Retention

The BMC persistently stores the most recent successfully attained commanded (via any of the sources in Table 7) power state. This feature is used to support the power state restoration feature. The power state is recorded as ON shortly after the BMC asserts *Power On* and sees that the *Power Good* signal indicates system power is present. The power state is recorded as OFF after the BMC de-asserts *Power On* and verifies that *Power Good* is no longer present.

4.2.7 AC Power Loss

When standby power returns after an AC power loss, the BMC can be configured via the *Set Power Restore Policy* command to restore the power state in one of three ways:

- Power Always Off: Leave power off when AC is restored
- Power Always On: Power system on when AC is restored
- Restore Power State: Restore power state to same as when AC was lost

4.3 Advanced Configuration and Power Interface (ACPI)

The BMC management features are designed to work in conjunction with the ACPI BIOS and hardware features of the server board. The following subsections illustrate these capabilities.

Table 8. ACPI Power States

State	Supported	Description
S0	X	Working
S1	X	Hardware context maintained; typically equates to processor/chipset clocks stopped
S2	—	Typically equates to stopped clocks with processor/cache context lost
S3	—	Typically equates to “suspend-to-RAM”
S4	X	Typically equates to “suspend-to-disk”
S5	X	Soft off

4.3.1 ACPI Power Control

The BMC works with the server board’s chipset to implement ACPI-compatible power control. The BMC is placed between the front panel power push button and the chipset so that it can implement the Secure Mode feature to disable the power button, and add additional power control sources to the system. The BMC passes power control requests through to the power push-button input of the chipset, allowing the ACPI-compatible power push-button logic in the chipset to be utilized. The BMC interprets the state of the chipset *Sleep S4* output signal as a power state request.

4.3.2 ACPI State Synchronization

The BMC is kept synchronized with the system ACPI state by the BIOS. The BIOS provides the ACPI state on power/sleep state transitions via the BMC ACPI/EC controller interface.

4.3.3 ACPI S1 Sleep Support

The following sequence of events occurs when the ACPI S1 state is entered. See the platform-specific EPS for additional details about ACPI sleep support.

- The front panel power LED blinks at a rate of 1 Hz with a 50% duty cycle (not controlled by the BMC).
- The front panel reset button is locked out by the BMC to prevent accidental system resets while in this mode.
- If enabled via the *Set ACPI Configuration Mode* command, the server board fans are set to sleep speed.
- The watchdog timer is stopped.

The BMC detects that the system has exited the ACPI S1 sleep state when it sees the *S1 Sleep* signal de-asserted. The BMC passes the state of the front-panel power button to the chipset during S1 mode. The chipset then de-asserts *S1 Sleep* when the button is pressed. Sleep state indication ceases whenever the system is powered down (S4 or S5).

4.3.4 ACPI S4 Sleep Support

Network Interface Chips hold the wake configuration state for Wake-on-LAN (WOL). This is typically configured by the operating system and is not cleared by system reset. WOL state needs to be cleared when going to S5 but not to S4. The BMC needs to know the difference so that it can assert a power-on reset to the Network Interface Chip that does clear the wake state configuration. The implementation for this is platform specific. See the platform-specific EPS documents for more information.

4.3.5 ACPI Power State Notify

If enabled through the *Set ACPI Configuration Mode* commands, the BMC sends the system's ACPI power state changes (S0, S1, S4/5) to other management controllers by sending the *Set ACPI Power State* command on the IPMB as indicated by their SDR management device records. The command is sent whenever there is a power state transition.

4.4 System Reset Control

The system has several different sources of reset. This section details these sources.

4.4.1 Reset Signal Output

The BMC pulses the *ICH Reset Button* signal on the server board to perform a system reset. The ICH performs the rest of the system reset process. The BMC cannot hold the system in reset and once started, the process is asynchronous with respect to BMC operation.

The reset portion of the power on process is automatically performed by the ICH and the BMC does not get involved in that part of the process.

4.4.2 Reset Control Sources

Table 9 shows the reset sources and the actions taken by the system.

Table 9. System Reset Sources and Actions

#	Reset Source	System Reset?	BMC Reset
1	Standby power comes up	No (no DC power)	Yes
2	Main system power comes up	Yes	No
3	Reset button or in-target probe (ITP) reset	Yes	No
4	Warm boot (DOS ctrl-alt-del, for example)	Yes	No
5	Command to reset the system	Yes	No
6	Set Processor State command	Yes	No
7	Watchdog timer configured for reset	Yes	No
8	FRB3 failure	Yes	No
9	PEF action	Optional	No
10	Exit BMC Firmware Update Mode	No	Yes

4.4.3 Front Panel System Reset

The reset button is a momentary contact button on the front panel. Its signal is routed through the front panel connector to the BMC, which monitors and de-bounces it. The signal must be stable for at least 50 ms before a state change is recognized.

If *Secure Mode* is enabled or the button is forced protected, the reset button does not reset the system, but instead a Platform Security Violation Attempt event message is generated. The reset button is disabled in sleep mode.

4.4.4 Warm Boot

This reset comes as a result of the BMC detecting that the *FRB3 Timer Halt* signal has become de-asserted after having previously been asserted by the BIOS to disable the FRB3 timer. This signal can be de-asserted by the BIOS. When the BIOS restarts after a warm boot (e.g., CF9 reset), it needs to translate that into a hard reset by sending a *Chassis Control* command to the BMC.

4.4.5 BMC Command to Cause System Reset

Chassis Control is the primary command used to reset the system. The *Set Processor State* command, which is used by the BIOS during POST, can also cause a system reset.

4.4.6 Watchdog Timer Expiration

The watchdog timer can be configured to cause a system reset upon its expiration. For more information, see the *Intelligent Platform Management Interface Specification, Version 2.0*.

4.4.7 FRB3 Failure

Simultaneous with pulsing ICH reset, the BMC starts its internal FRB3 timer. If the BIOS does not assert the *FRB3 Timer Halt* signal to cause the BMC to stop this timer, the BMC will reset the system when the timer expires.

4.5 BMC Reset Control

Table 10 shows all the sources of BMC reset and the actions by the System and the BMC.

Table 10. BMC Reset Sources and Actions

#	Reset Source	System Reset?	BMC Reset
1	Standby power comes up	No (system not up yet)	Yes
2	BMC exits Firmware Update mode	No	Yes

4.5.1 BMC Exits Firmware Update Mode

The BMC firmware can be updated using firmware transfer commands through the LPC interface. The BMC automatically enters Firmware Transfer Mode if it detects that the *Force Update* signal is asserted during initialization or if the operation code checksum validation fails. Upon exit from Firmware Transfer Mode, the BMC resets itself. The BMC will re-synchronize itself to the state of the processor and power control signals it finds when it initializes.

4.6 System Initialization

4.6.1 Processor Temperature and Voltage Threshold Setting

Some platforms extract processor temperature and voltage sensor thresholds from information provided by the Processor Information ROM (PIROM) devices. On those platforms, the SDR records for the processor temperature and voltage sensors should be configured to not initialize sensor thresholds. Processor temperature and voltage sensor critical thresholds are derived from the PIROM information.

4.6.2 Fault Resilient Booting (FRB)

Fault Resilient Booting (FRB) is a set of BIOS and BMC algorithms and hardware support that allow a multiprocessor system to boot in case of failure of the bootstrap processor (BSP) under certain conditions. The FRB algorithms detect BSP failure and take steps to disable that processor and reset the system so another processor will run as the BSP. For FRB3, the BMC relies on the BIOS to assert the *FRB3 Timer Halt* signal, which indicates to the BMC that the BSP is successfully running code.

4.6.2.1 Processor Sensor Offset Support

The BMC's internal sensor table retains a set of per-processor status sensors that may include information about the thermal trip, FRB2, FRB3, IERR, processor presence, terminator presence, and disable status for each processor. FRB uses the status derived from these sensors to determine whether a processor should be disabled.

These sensors are manual rearm processor status sensors, so that after an event state (sensor offset) has been asserted, it will remain asserted until one of the following happens:

- A *Rearm Sensor Events* command is executed for that processor status sensor
- A *Processor Retest* command is executed
- The Init Agent is re-run
- A system reset or power cycle occurs

Table 11. Requirements for Processor Status

Offset	Processor Status	Detected By	Persistent
0	IERR	BMC	—
1	Thermal trip	BMC	X
2	FRB1/BIST failure	BIOS	—
3	FRB2/Hang in POST failure	BIOS	—
4	FRB3/Processor startup/initialization failure (CPU fails to start)	BMC	—
5	Configuration error (for DMI)	BMC	—
6	SM BIOS uncorrectable CPU-complex error	Not Supported	—
7	Processor presence detected	BMC	—
8	Processor disabled	N/A	—
9	Terminator presence detected	Not Supported	—

The BMC is responsible for persistently retaining ThermTrip history for each processor. This history tracks whether the processor has had a ThermTrip since the last processor sensor re-arm or retest.

When the BMC detects an empty processor socket, it clears the persistent processor status and sets the disable bit for that processor. Note that on some platforms, determining whether a processor is in a processor socket cannot be done until DC power is on.

In certain systems, the BMC will check for processor population errors. If processor socket population rules are not followed, the BMC *may* produce a beep code through the system speaker and either hold all processor sockets in reset or refuse to power up. See the platform-specific BMC EPS for this behavior of the BMC. Beep codes are listed in Table 14.

4.6.2.2 Processor Disabling

To disable a processor, the BMC asserts the corresponding *Processor Disable* signal in conjunction with resetting the system. The signal used for this purpose is processor type specific.

Processors are selected for disabling as a result of FRB3 or ThermTrip failures or on command from the BIOS as a result of FRB2 errors.

4.6.2.3 BSP Identification

The BMC provides positive indication of disabled processors, not which processor is the BSP. Software that needs to identify the BSP should do so by using the multiprocessor specification tables (see the applicable *BIOS EPS*).

The BMC FRB3 algorithm assumes that a particular processor is the BSP in order to know which processor to disable, if there is an FRB3 timeout. How this is done is platform-specific.

4.6.2.4 Watchdog Timer Timeout Reason Bits

In order to implement FRB2, during POST, the BIOS determines whether a BMC watchdog timer timeout had occurred on the previous boot attempt, and if so, whether that timeout was an FRB2 timeout, a system management software (SMS) timeout, or an intentional timed hard reset. The BMC provides the IPMI *Get Watchdog Timer* command to facilitate this.

The timeout reason bits implemented by the BMC watchdog are maintained by the BMC across system resets and DC power cycles, but not AC power cycles.

4.6.2.5 FRB3

FRB3 refers to the FRB algorithm that detects whether the BSP is healthy enough to execute the BIOS code. The BMC starts an internal timer (FRB3 timer) when the system is powered up or hard reset. The BIOS stops this timer in Power-On Self Test (POST) by asserting the *FRB3 Timer Halt* signal to the BMC. This requires that the BSP actually runs BIOS code. If the timer is not stopped, then on timer expiration, the following occurs:

- The BMC disables the current BSP
- The BMC logs an FRB3 error event (if enabled via the processor status sensor SDR configuration)
- The BMC chooses another BSP (from the set of non-failed processors)
- The BMC resets the system.

This process repeats until either the system boots without an FRB3 timeout, or all of the processors have been disabled. At this point, if all the processors have been disabled, the BMC will attempt to boot the system on one processor at a time, irrespective of processor error history. This is called *Desperation Mode*.

Once all the processors that are present have been tried in Desperation Mode, the FRB3 algorithm is stopped and the first processor present is enabled and allowed to boot even though it fails to assert the *FRB3 Timer Halt* signal. An FRB3 failure beep code is sounded. This is called *Final Desperation Mode*.

FRB3 requires multiple processors. The BMC verifies that there are at least two processors installed in the system. If only one processor is present the FRB3 timer may or may not be started, depending on the configuration.

The default FRB3 timeout interval is platform-specific and is determined by BIOS requirements. This timeout value is not user settable.

4.6.2.6 FRB2

FRB2 refers to the FRB algorithm that provides for detection of system failures (hangs, etc) during the remaining portion of POST (after FRB3). The BIOS uses the BMC watchdog timer to back up its operation during POST. The BIOS configures the watchdog timer indicating that the BIOS is using the timer for the FRB2 phase of operation.

After the BIOS has identified the BSP and saved that information, it sets the watchdog timer FRB2 timer use bit and loads the watchdog timer with the new timeout interval. It then disables FRB3 by asserting the *FRB3 Timer Halt* signal. This sequence ensures that no gap exists in watchdog timer coverage between FRB3 and FRB2.

If the watchdog timer expires while the watchdog use bit is set to FRB2, the BMC (if so configured) logs a Watchdog expiration event showing an FRB2 timeout. The BMC then hard resets the system (assuming Reset was selected by the BIOS as the watchdog timeout action).

The BIOS is responsible for disabling the FRB2 timeout before initiating the option ROM scan or before displaying a request for a Boot Password. If the FRB2 timer expires (i.e., a processor has failed FRB2), the BMC resets the system. If it again passes FRB3, the system boots from the processor that just failed FRB2 (the BSP).

As part of its normal operation, the BIOS obtains the watchdog expiration status from the BMC. If this status shows an expiration with an FRB2 timer use, the BIOS generates a Processor FRB2 failure System Event Log (SEL) entry with the last POST code generated during the previous boot attempt in the OEM bytes of the event entry.

If the BIOS determines that the boot failure was processor related (e.g., BIST related), the BIOS then issues a *Set Processor State* command to the BMC, telling it to disable the BSP and reset the system. The BMC disables the processor that failed FRB2 and resets the system, causing a different processor to become the BSP.

4.6.2.7 FRB1

The FRB1 algorithm is implemented by the BIOS, which detects the failure of the BSP by examining a processor's Built-In Self Test (BIST) results. If a BIST failure is indicated, the BIOS indicates this to the BMC via the *Set Processor State* command, setting the FRB1 offset and indicating that the processor should be disabled and that the system should be reset. The following list describes the steps occurring when BIOS implements FRB1:

1. The BIOS reads processor BIST results, determines failure.
2. The BIOS saves the FRB1 (BIST) failure status for the processor to CMOS.
3. The BIOS attempts to identify the BSP and force a switch to another processor as the BSP. This is accomplished via a *Set Processor State* command to the BMC indicating the BSP had an FRB1 failure and that the BMC should disable the processor and reset the system. This command will cause the BMC to log an FRB1 failure event message (if so configured by the processor status sensor SDR configuration), disable the processor, and reset the system.
4. If the BIOS cannot successfully communicate with the BMC in step 3, the FRB3 timer times out and the BMC then disables the processor and resets the system as described in the FRB3 section above.

4.6.3 Boot Control Support

The BMC supports the IPMI 2.0 Boot Control feature. This feature allows the boot device and boot parameters to be managed remotely. Table 60 60 specifies the supported boot option parameters. The size of the Boot Initiator Mailbox is specified in the platform-specific BMC EPS.

4.7 Integrated Front Panel User Interface

The Intel® Management Module BMC incorporates the front panel interface functionality. The specific front panel model and supported features may be platform specific. Please refer to the platform specific board TPS for information on front panel features and LED behavior.

4.7.1 Chassis ID LED

The chassis ID LED is used to provide a visual indication of a system being serviced. The chassis ID LED is turned on and off by three methods:

- It is toggled by the chassis ID button
- It can be controlled by the *Chassis Identify* command (IPMI)
- It can be controlled by the *Chassis Identify LED* command (OEM)

Table 12. Chassis ID LED Indicator States

State	LED State
Identify active via button	Solid On
Identify active via command	~1 Hz blink
Off	OFF

There is no precedence or lock-out mechanism for the control sources. When a new request arrives, all previous requests are terminated. For example, if the chassis ID LED is blinking and the chassis ID button is pressed, then the chassis ID LED will change to solid on. If the button is pressed again, with no intervening commands, the chassis ID LED will turn off.

4.7.2 Front Panel / Chassis Inputs

The BMC monitors the front control panel switches and other chassis signals. The front control panel input buttons are momentary contact switches, which are de-bounced by the BMC processor firmware. The de-bounce time is 50 ms, which means the signal must be in a constant low state for 50 ms before it is treated as asserted.

4.7.3 Chassis Intrusion

Some platforms support chassis intrusion detection. On those platforms, the BMC monitors the state of the *Chassis Intrusion* signal and makes the status of the signal available via the *Get Chassis Status* command and *Physical Security* sensor state. If enabled, a chassis intrusion state change will cause the BMC to generate a *Physical Security* sensor event message with a *General Chassis Intrusion* offset (00h).

4.7.4 Diagnostic Interrupt (Front Panel NMI)

As stated in the *IPMI 2.0 Specification*, a Diagnostic Interrupt is a non-maskable interrupt or signal for generating diagnostic traces and 'core dumps' from the operating system. This is typically an NMI on IA-32 systems and an INIT on Itanium™-based systems.

The Diagnostic Interrupt button is connected to the BMC through the front panel connector. Pressing the Diagnostic Interrupt button will cause the BMC to do the following:

- Generate a *Critical Event* sensor event message with a *Front Panel NMI / Diagnostic Interrupt* offset (00h) (if enabled).
- Set the OEM 1 (40h) bit in the message flags. The message flags can be read with the *Get Message Flags* command and cleared with the *Clear Message Flags* command.
- Set bit 0 in NMI/INIT Source 1. This information can be retrieved with the *Get NMI/INIT Source* command.
- Generate a system NMI/INIT pulse whose duration is platform-specific and unrelated to the button press duration.

Once an NMI/INIT has been generated by the BMC, the BMC will not generate another until the system has been reset or powered down.

The BMC automatically clears the OEM 1 message flag and NMI/INIT sources whenever it detects a system reset, or is itself reset. The Diagnostic Interrupt button is not disabled or otherwise affected when the system is in Secure Mode, although the *Set Secure Mode Options* command can be used to block use of this button.

4.7.5 CMOS Clearing

The CMOS can be cleared in three ways, two of which the BMC participates in:

- CMOS Clear Jumper installed (primary method)
- *CMOS Clear* state asserted via BMC *CMOS Clear Options* command (allows remote clear operation)
- Front Panel CMOS clear sequence

The BMC provides the Front Panel CMOS Clear feature to allow CMOS to be cleared without accessing the CMOS Clear Jumper. The Front Panel CMOS Clear feature consists of a sequence of button presses that will both assert the BMC *CMOS Clear* state and power the system up. This feature can be enabled or disabled through the *CMOS Clear Options* command.

The following sequence of events must occur to invoke the Front Panel CMOS Clear feature.

1. Standby power must be on, the system power must be off, and the feature enabled.
2. The front panel reset button must be pressed and held for at least 4 seconds.
3. With the front panel reset button still pressed, the front panel power button must be pressed and then both buttons released.

The BMC produces a single beep through the system speaker to confirm the CMOS clear button sequence.

The BMC asserts the CMOS Clear hardware signal. The BIOS deasserts the BMC CMOS Clear state using the *CMOS Clear Options* command at each boot to avoid the possibility of repeated CMOS Clear operations. The BMC CMOS Clear state remains active until one of the following events occurs:

- It is forced off using a *CMOS Clear Options* command (this is the normal operational case, by BIOS).
- The reset button is pressed.
- The power button is pressed and released.
- The system is powered off.

4.7.6 Secure Mode Operation

The Secure Mode feature allows the front panel switches to be protected against unauthorized use/access. Secure Mode is enabled and controlled via the *Set Secure Mode Options* command.

If it is enabled, Secure Mode can be controlled via the *Secure Mode KB* signal from the keyboard controller. When Secure Mode is enabled and active, protected front panel switch presses will generate Secure Mode Violation events – specifically, assertions of the *Secure Mode Violation Attempt* offset (00h) of the BMC's *Platform Security Violation Attempt* sensor.

Secure Mode state is cleared whenever AC power or system power is applied, when a system reset occurs, or when a BMC reset occurs. The Secure Mode state includes the bits that specify which actions are to be taken when Secure Mode is active, as well as the *Force Secure Mode On* bit. In addition, the *Set Secure Mode Options* command allows specific front panel switches to be protected irrespective of Secure Mode state. See the command definition for details.

The set of buttons protected when Secure Mode is active varies depending on the system ACPI power state:

Table 13. Secure Mode vs. ACPI State

ACPI State	Power Switch	Sleep Switch	Reset Switch	Diagnostic Interrupt (FP NMI) Switch	ID Switch
S0	Protected	Protected	Protected	Unprotected	Unprotected
S1	Partial (see note)	Unprotected	Protected	Unprotected	Unprotected
S4/S5	Unprotected	Unprotected	Unprotected	Unprotected	Unprotected

Note: The partial support for the Power Switch in S1 means that system wake up due to the Power Switch is allowed, but the 4-second override is blocked.

Note: The Diagnostic Interrupt (Front Panel NMI) switch can be locked using the *Set Secure Mode Options* command but is never protected by Secure Mode. This allows a system to be recovered from a hung state when Secure Mode is active.

4.7.7 Set Fault Indication Command

The *Set Fault Indication* command can be used by satellite controllers and system management software to communicate fan, temperature, power, and drive fault states to the BMC. The BMC consolidates that state with its own platform state when determining how to set front panel indicator LED states and how to control other behavior such as fan boosting.

The command has a source field that allows the BMC to track the fault states of multiple sources. Each source must use a separate unique source ID. For example: Hot-swap controller 0 has ID 1 and Hot-swap controller 1 has ID 2.

The fault state of each source is tracked independently. Whenever a source sets the fault state for a particular fault type (e.g., fan or power), the new state overrides the previous state. The tracked fault state is cleared on system power up and system resets.

4.8 Private Management I²C Buses

The BMC can control multiple private I²C buses. The BMC is the sole master on these buses. External agents must use the BMC's *Master Write/Read I²C* command if they require direct communication with a device on any of these buses. In general, only FRU devices are accessible in this manner. Sensor devices should not be directly accessed by BMC clients.

4.9 Watchdog Timer

The BMC implements a fully IPMI 2.0 compatible watchdog. See the IPMI specifications for details. The NMI / Diagnostic Interrupt specified for IPMI 2.0 watchdog timer will be associated with an NMI.

4.10 System Event Log (SEL)

The BMC implements the logical System Event Log (SEL) device as specified in the *Intelligent Platform Management Interface Specification, Version 2.0*. The SEL is accessible via all communication transports. In this way, the SEL information can be accessed while the system is down by means of out-of-band interfaces. See Table 67 for optional SEL commands that are supported.

4.10.1 Servicing Events

Events may be received while the SEL is being cleared. The BMC implements an event message queue to avoid messages being lost. Messages are not overwritten once they are stored in the queue.

The BMC recognizes duplicate event messages by comparison of sequence number and message source. For more information, see the *IPMI 2.0 Specification*. Duplicate event messages are discarded (filtered) by the BMC after they are read from the event message queue. Therefore, the queue can contain duplicate messages.

4.10.2 SEL Entry Deletion

The *Delete SEL* command does not reclaim deleted SEL record space, but marks the records in such a way that causes them to be invisible via the IPMI *Get SEL Entry* command. In addition to the standard IPMI SEL commands, the BMC implements an extension command *Get All SEL Entry* that allows the caller to enumerate all the SEL records including those that have been “deleted” by the *Delete SEL* command.

Note: The *Get All SEL Entry* command is an OEM command and assumes a particular SEL implementation. If a software application that accesses the SEL is required to be IPMI compatible (e.g., it is expected to work with any IPMI compliant platform), then this command should not be used.

4.10.3 SEL Erasure

SEL erasure is a background process. After initiating erasure via the *Clear SEL* command, the client will have to execute more *Clear SEL* commands to get erasure status to determine when the SEL erasure has completed. This may take several seconds. SEL events that arrive during the erasure process will be queued until the erasure is complete and then committed to the SEL.

SEL erasure will generate an *Event Logging Disabled (Log Area Reset/Cleared offset)* sensor event and the BMC will log this event to the SEL after it is cleared.

4.10.4 Timestamp Clock

The BMC maintains a four-byte internal timestamp clock used by the SEL and SDR subsystems. This clock is incremented once per second and is read and set using the *Get SEL Time* and *Set SEL Time* commands, respectively. The *Get SDR Time* command can also be used to read the timestamp clock. These commands are specified in the *Intelligent Platform Management Interface Specification, Version 2.0*.

4.10.4.1 Real-Time Clock (RTC) Access

The Intel® Management Module has no way for the BMC to directly access the system RTC. The BIOS provides the time to the BMC during POST.

After a BMC reset, the BMC sets the initial value of the timestamp clock to 0x00000000. It is incremented once per second after that. A SEL event containing a timestamp from 0x00000000 to 0x140000000 has a timestamp value that is relative to BMC initialization.

During POST, the BIOS tells the BMC the current RTC time via the *Set SEL Time* command. The BMC will maintain this time, incrementing it once per second, until the BMC is reset or the time is changed via another *Set SEL Time* command.

If the RTC changes during system operation, SMS is responsible for keeping the BMC and system time synchronized.

4.10.4.2 Boot Event Messages

The BIOS downloads the Real Time Clock (RTC) system date and time to the BMC during POST and logs a pair of boot events into the SEL. The first event is logged just before the BIOS setting the BMC timestamp clock and the second is logged immediately after. These records do not indicate an error, and software that parses the event log should treat it as such. These records should be used by system software as a method to synchronize the RTC time of events that occurred before the BMC timestamp clock was set to the RTC.

4.11 Sensor Data Record (SDR) Repository

The BMC implements the logical Sensor Data Record repository device as specified in the *Intelligent Platform Management Interface Specification, Version 2.0*. The SDR Repository is accessible via all communication transports. In this way the SDR Repository information can be accessed while the system is down by means of out-of-band interfaces. See Table 92 for additional SDR command support.

4.11.1 SDR Repository Erasure

SDR Repository erasure is a background process. After initiating erasure via the *Clear SDR Repository* command, the client will have to execute more *Clear SDR Repository* commands to get erasure status to determine when the SDR Repository erasure has completed. This may take several seconds. The SDR Repository cannot be accessed or modified until the erasure is complete.

4.11.2 Initialization Agent

The BMC implements the internal sensor initialization agent functionality specified in the *Intelligent Platform Management Interface Specification, Version 2.0*. When the BMC initializes or on a system boot, it scans the SDR repository and configures the intelligent IPMB devices that have management controller records and the *Init Required* bit set in their SDR repository. This includes setting sensor thresholds, enabling/disabling sensor event message scanning, and enabling/disabling sensor event messages.

The initialization process causes those IPMB micro-controllers to rearm their event generation. In some cases, this causes a duplicate event to be sent to the BMC. The BMC's mechanism to detect and delete duplicate events should prevent any duplicate event messages from being logged.

4.12 Field Replaceable Unit (FRU) Inventory Device

The BMC implements the interface for logical FRU inventory devices as specified in the *Intelligent Platform Management Interface Specification, Version 2.0*. This functionality provides commands used for accessing and managing the FRU inventory information. These commands can be delivered via all interfaces.

The BMC provides FRU device command access to its own FRU device, as well as to the FRU devices throughout the system. The FRU device ID mapping is available via the Sensor Data Record (SDR) type 11 FRU Device Locator Records. The BMC controls the mapping of the FRU device ID to the physical device. Per the IPMI specification, FRU device 0 is always located on the server board. The Intel® Management Module BMC FRU will always be FRU device 1 and the GCM FRU will be device 2. All of these boards maintain onboard non-volatile storage to hold the FRU data.

4.12.1 BMC FRU Inventory Area Format

The BMC FRU inventory area format follows the Platform Management FRU Information Storage Definition. See *Platform Management FRU Information Storage Definition, Version 1.0* for details.

The BMC provides only low-level access to the FRU inventory area storage. It does not validate or interpret the data that are written. This includes the common header area. Applications cannot relocate or resize any FRU inventory areas.

Note: Fields in the internal use area are not for OEM use. Intel reserves the right to relocate and redefine these fields without prior notification. Definition of this area is part of the software design. The format in the internal use area may vary with different BMC firmware revisions.

4.13 Diagnostics and Beep Code Generation

The BMC may generate beep codes upon detection of the failure conditions. Beep codes are sounded each time the problem is discovered (for example, on each power up attempt) but are not sounded continuously. Codes common across platforms are listed in Table 14. Each digit in the code is represented by a sequence of beeps whose count is equal to the digit. Platform specific beep codes are defined and documented in the platform TPS.

Table 14. BMC Beep Codes

Code	Reason for Beep
1-5-1-1	FRB3 failure (processor failure)
1-5-2-1	CPU: Empty Slot
1-5-2-2	CPU: No Processors
1-5-2-3	CPU: Configuration Error (e.g., VID mismatch)
1-5-2-4	CPU: Configuration Error (e.g., BSEL mismatch)
1-5-4-2	Power fault: DC power unexpectedly lost (power control failures)
1-5-4-3	Chipset control failure
1-5-4-4	Power control fault

4.14 NMI

The BMC has specific monitoring and signal generation functionality in regards to the NMI (Non-maskable Interrupt) signal. This is similar for Itanium™-based platforms, which have and the INIT (Initialization) signal. These signals and BMC-related functionality are related but not equivalent. For purposes of this document, this signal pair will be referred to as NMI/INIT due to common command support in the IMM BMC for Itanium™-based platforms.

When a Diagnostic Interrupt (also referred to as FP Diagnostic Interrupt or NMI / Diagnostic Interrupt) is generated by the BMC, the actual signal that is pulsed is the NMI signal. A FP Diagnostic Interrupt sensor is used to log SEL events for assertion of the Diagnostic Interrupt.

4.14.1 Signal Generation

The BMC generates an NMI /INIT pulse under certain conditions. The BMC-generated NMI/INIT pulse duration is at least 30 ms. Once an NMI/INIT has been generated by the BMC, the BMC will not generate another until the system has been reset or powered down except that enabling NMI/INIT via an *NMI/INIT Enable/Disable* command will re-arm the NMI/INIT.

The BMC captures the NMI/INIT source(s) and makes that information available via a *Get NMI/INIT Source* command. Reading the NMI/INIT source information causes it to be cleared. The *Set NMI/INIT Source* command is available to other agents (e.g., BIOS SMI Handler) to register NMI/INIT sources when they detect NMI/INIT generating errors. OS NMI/INIT handlers that save system crash state can use the *Get NMI/INIT Source* command to determine and save the cause of the NMI/INIT.

BMC NMI/INIT generation can be disabled by the *NMI/INIT Enable/Disable* command. The default state is enabled and the enabled/disabled state is volatile (not saved across AC power cycles).

The following may cause the BMC to generate an NMI/INIT pulse:

- Receiving a *Chassis Control* command to pulse the Diagnostic Interrupt. Use of this command will not, however, cause an event to be logged in the SEL.
- Detecting that the front panel Diagnostic Interrupt button has been pressed. See section 4.7.4 for details.

- A PEF table entry matching an event where the filter entry has the Diagnostic Interrupt action indicated.
- A processor IERR (support for this is platform specific).
- Watchdog timer pre-timeout expiration with NMI/Diagnostic Interrupt pre-timeout action enabled.
- Receiving a *Set NMI/INIT Source* command issued from one of the command interfaces.

4.14.2 Signal State Monitoring

The BMC may monitor the state of the system NMI/INIT signal through the support of an NMI/INIT Signal State sensor. This capability is provided for diagnostic purposes.

4.15 SMI Generation

This section discusses Server Management Interrupt (SMI) generation. The BMC generates SMIs in IA-32 bit systems under the following conditions:

- Watchdog timer pre-timeout expiration with SMI pre-timeout interrupt specified
- Message received in event message buffer with *Event Message Buffer Full Interrupt Enabled* set in the global enables.

The SMI generation is software configurable. The above conditions may or may not be enabled to cause an SMI.

4.16 Processor Sensors

The BMC provides IPMI sensors for processors and associated components such as voltage regulators and fans. Most of these sensors are implemented on a per-processor basis.

Table 15. Processor Sensors

Sensor Type	Per Proc	Description
Processor Status	yes	Processor presence and fault state. See section 4.6.2 for details
Processor Temperature	yes	Temperature from processor itself
Processor VRD Temperature	yes	Temperature from processor voltage regulator
Processor Voltage	yes	Voltage from processor voltage regulator
Processor Fan Speed	yes	Speed of processor fan-sink (not always present)
Processor Thermal Control	yes	Percentage of time a processor is throttling due to thermal conditions
CPU Configuration Error	no	CPU type mismatch, improper socket population, etc

4.17 Fan Management

The BMC controls and monitors the system fans. For each fan, a fan speed sensor provides fan failure detection. Some platforms also provide fan presence detection that the BMC maps into per-fan presence sensors.

On some platforms, it is possible for the BMC to control the speed of some fans. Controllable fans are divided into fan domains in which there is a separate fan speed control for each domain and a separate fan control policy configurable for each domain. A fan domain can have a set of temperature and fan sensors associated with it, which are used to determine the current fan domain state. A fan domain has three states: Sleep, Nominal, and Boost. The Sleep and Boost states have fixed (but configurable – via OEM SDRs) fan speeds associated with them. The Nominal state has a variable speed determined by the fan domain policy (see section 4.17.1). An OEM SDR record is used to configure the fan domain policy. See TControl Fan Speed Control Record in the Appendix.

Fan domain state is controlled by several factors (states in order of precedence, high to low):

Boost

- Associated fan in a critical or non-recoverable state (non-redundant fan configuration).
- Fan domain is in an *insufficient resources* redundancy state (redundant fan configuration).
- Any temperature sensor in a critical or non-recoverable state.

Sleep

- No boost conditions, system in ACPI S1 sleep state, and BMC configured to transition fan domains to sleep state.

Nominal

- See section 4.17.1

4.17.1 Nominal Fan Speed

It is possible to configure a fan domain's Nominal fan speed to be either static (fixed value) or controlled by the state of one or more associated temperature sensors.

OEM SDR records are used to configure which temperature sensors are associated with which fan control domains as well as the relationship (algorithm) between the temperature and fan speed control (PWM) value. Multiple OEM SDRs may reference/control the same fan control domain and multiple OEM SDRs may reference the same temperature sensors.

The computed domain PWM value is the maximum of the values computed for that domain using one or more instances of the following algorithms, with the exception that a Stepwise Linear instance can be configured to provide the domain maximum at any time.

Two control algorithms are possible:

- **Stepwise Linear Control**

In this algorithm, a table mapping a temperature sensor's values to fan speed PWM values is provided for the temperature sensor. The PWM value is temperature related, but fixed for any particular temperature. A configurable hysteresis is implemented to avoid oscillatory behavior.

It is possible to configure an instance of this algorithm to provide the maximum PWM value for the fan control domain, meaning that even if other instances (e.g., a Clamp instance) referencing the same domain indicate that the PWM value should be higher, the domain's PWM value will not go higher than the value provided by this instance.

- **Temperature Clamp Control**

In this algorithm, a threshold is defined such that if the temperature exceeds this value, the fan speed contribution from the temperature sensor will increase over time until either the fan speed reaches saturation (maximum speed) or the temperature reduces to below the threshold. If the temperature is below the threshold, the sensor's contribution will be reduced over time until it goes to zero. The rate at which the sensor's contribution increases or decreases is configurable via the OEM SDR.

The temperature to clamp to can be configured to either be a fixed value (specified in the OEM SDR) or a CPU TControl value communicated via an OEM IPMI command Set Processor TControl.

The transition from one computed Nominal fan speed (PWM value) to a new one is ramped over time to avoid audible transitions. The ramp rate is configurable via the OEM SDR. Note that multiple Additive and Clamp Controls can be defined for each fan domain and used simultaneously.

4.17.2 Sleep State Fan Control

Using the *Set ACPI Configuration Mode* command, the BMC may be configured to set the fans to a fixed sleep state speed when the system is in the S1 Sleep state.

4.17.3 Fan Redundancy Detection

The BMC supports redundant fan monitoring and implements fan redundancy sensors. A fan redundancy sensor generates events when its associated set of fans transitions between redundant and non-redundant states, as determined by the number and health of the component fans. The definition of fan redundancy is configuration dependent. The BMC allows redundancy to be configured on a per fan-redundancy sensor basis via OEM SDR records (See the Appendix).

A single fan failure, or removal of a single fan in a chassis that supports hot swap fans, in a redundant fan configuration is a non-critical failure and will be reflected in the front panel status as such.

4.17.4 Hot Swap Fan Support

Some chassis and server boards provide support for hot swap fans. These fans can be removed and replaced while the system is operating normally. The BMC implements fan presence sensors (sensor type *Slot/Connector (21h)*, event/reading type *Sensor Specific (6Fh)*) for each hot swappable fan. When a fan is not present, the associated fan speed sensor is put into an *Init In Progress* state. In addition, when a fan is replaced, and it is part of a fan control domain, the fan speed for the domain is temporarily set to high (kick-start) to ensure proper fan starting.

4.18 Power Unit Management

The BMC always implements a Power Unit sensor (event/reading type 09h). This sensor provides power state and power unit fault state. The following offsets are always supported:

- Power Off/Down: Asserted when system power is off, Deasserted when power is on.
- Power Cycle: Asserted and then deasserted as a result of a Power Cycle request.
- AC Lost: Asserted on BMC power on if the previous system power state was on.
- Power Unit Failure: Asserted if power unit changes power state unexpectedly.

Other offsets may be implemented on a platform specific basis depending on power unit capabilities.

The BMC also sends power supply state information to Hot Swap controllers by via the *Set Power Supply State* command.

4.18.1 Power Supply Status Sensors

Platforms that support power units with multiple power supplies (e.g., redundant power units) implement a Power Supply Status sensor (event/reading type 08h) for each power supply. The *presence* and *power supply failure detected* offsets are always supported. Other offsets are supported based on power supply capabilities and are described in the platform-specific board TPS.

4.18.2 Power Gauge/Nozzle

Some chassis provide power supplies that implement power/current consumption monitoring. The BMCs on platforms that use those chassis implement the following sensors:

- Per-power supply input current sensors (power gauge)
 - Sensor Type: Current (03h)
 - Event/reading Type: Threshold (01h)
 - Entity ID: Power Supply (0Ah)
 - Units: Amps (5)

These sensors provide the current values that each power supply is drawing from the input power cord (AC). These sensors can be told from the per-power supply output current sensors by looking at the Sensor Direction field of the Accuracy byte of the Full Sensor SDR.

- Per-power supply output power sensors – aggregate per supply (power gauge)
 - Sensor Type: Other Units (0Bh)
 - Event/Reading Type: Threshold (01h)
 - Entity ID: Power Supply (0Ah)
 - Units Watts (6)

These sensors provide the aggregate power being used by the system as monitored by each power supply.

- Per-power supply output current sensors (per monitored output voltage) (power gauge).
 - Sensor Type: Current (03h)
 - Event/Reading Type: Threshold (01h)
 - Entity ID: Power Supply (0Ah)
 - Units: Amps (5)

These sensors provide the current being provided by each monitored power supply voltage (from each supply). These sensors can be told from the per-power supply input current sensors by looking at the Sensor Direction field of the Accuracy byte of the Full Sensor SDR.

These sensors are associated with particular power supply entities. If the power supply is not present (as indicated by the associated Power Supply Status sensor), then these sensors should be ignored by software. Note that these power units generally support power supply hot plug and so the power supply presence may change dynamically.

4.18.3 Power Unit Redundancy

The BMC supports redundant power subsystems and implements a Power Unit Redundancy sensor per platform. A Power Unit Redundancy sensor is of sensor type Power Unit (09h) and reading type Availability Status (0Bh). This sensor generates events when a power subsystem transitions between redundant and non-redundant states, as determined by the number and health of the power subsystem's component power supplies. The definition of redundancy is power subsystem dependent and sometimes even configuration dependent. The BMC allows redundancy to be configured on a per power-unit-redundancy sensor basis via the OEM SDR records.

4.19 System Memory RAS and Bus Error Monitoring

System Memory and Bus Error monitoring is done by system BIOS. The BIOS updates the status of RAS configuration at startup and later at run time. BMC monitors and logs SEL events based on the SDR definitions. In addition, the BIOS helps the BMC maintain the current DIMM presence and failure state and current memory RAS configuration (e.g., sparing, mirroring, RAID).

Some platforms provide support for monitoring errors on system buses such as front side bus (FSB) errors and PCI bus errors. These are monitored by BIOS, which generates Critical Interrupt sensor SEL events when the errors are detected.

The supported sensors are described below.

4.19.1 SMI Timeout Sensor

The Intel® Management Module BMC supports an SMI Timeout Sensor (sensor type OEM (F3h), event type Discrete (03h)) that asserts if the SMI signal has been asserted for longer than a fixed, platform dependent time period (nominally 90 seconds). A continuously asserted SMI signal is an indication that BIOS cannot service the condition that caused the SMI. This is usually because that condition prevents BIOS from running.

When an SMI Timeout occurs, the BMC will do an after-crash (post-mortem) system scan for uncorrectable memory and front-side bus errors. Any errors detected will be logged against either a Memory or Critical Interrupt sensor, as appropriate.

After the SMI timeout occurs and a time stamped SEL entry is logged indicating that an uncorrectable FSB and/or memory error has occurred, then 20 additional OEM type E0 SEL entries are also logged. These entries represent a dump of the MCH DRAM Controller Error Reporting registers (0x00 – 0xEF). Each SEL entry contains 12 bytes of register information.

The BMC supports sensors for reporting post-mortem System Memory errors and for DIMM presence, disabled state, and failure. The System Memory error monitoring and DIMM sensor features interact since, for example, if the error is isolatable to a DIMM, a BMC detected uncorrectable memory error will cause the DIMM to be marked failed.

4.19.2 Memory Sensor

The BMC supports one *Memory* type (0Ch) sensor supported that is event only. The sensor is only logged against by BMC detected errors (post-mortem). Events will be event type specific (reading code 6Fh). The supported sensor offsets is 01h, Uncorrectable ECC.

The format of the event data 2 and event data 3 bytes are as specified in section 6 of this document. The syndrome byte (event data 2) will be provided if supported by the hardware. Event data 3 will represent a DIMM ID or DIMM bank depending on what can be determined based on the hardware support and error type.

4.19.3 Critical Interrupt Sensor

The BMC implements a *Critical Interrupt* (13h) sensor for reporting the following conditions/events:

- Bus Uncorrectable Error: Only sensed after an SMI timeout (post-mortem)
- Front Panel NMI/Diagnostic Interrupt: Monitored during normal system operation

4.19.4 Memory Board Sensors

On server boards that use memory boards, the BMC implements a sensor for each board slot/connector that provides memory board inventory state. These sensors are IPMI sensor type *Slot/Connector* (21h) and event/reading type *Sensor Specific* (6Fh). The supported offset is 02h, Device Installed.

The sensor entity ID in the SDRs allows software to associate this sensor with the memory boards. These sensors may not be readable when the system is in a power-off state (they may be in an *init-in-progress* state) depending on whether board presence is determinable in that state.

4.19.5 DIMM Sensors

One DIMM sensor is available for each DIMM slot. These sensors are IPMI sensor type *Slot/Connector* (21h) and event/reading type *Sensor Specific* (6Fh). The supported offsets are:

- 00h: Fault Status Asserted
- 02h: Device Installed
- 08h: Device Disabled
- 09h: Slot Holds Spare Device

The BMC detects DIMM presence and sets the DIMM sensor *Device Installed* state at system power up time if enabled by the platform firmware. It does this by looking for the DIMM SPD FRU devices on the SMBus.

The DIMM fault state will be asserted by BIOS for the following conditions:

- The BIOS detects an uncorrectable ECC error
- The BIOS issues a *Set DIMM State* command indicating that a DIMM's fault status should be asserted

The DIMM disabled state will be asserted by the following:

- The BIOS issues a *Set DIMM State* command indicating that a DIMM's disabled status should be asserted

The DIMM fault and disabled state will be de-asserted (reset) by the following:

- A DIMM slot becoming empty
- A *ReArm Sensor* command being executed for that DIMM sensor
- A *ReArm DIMMs* command being executed

The DIMM Sparing indication is informational and used by the memory redundancy features to determine redundancy state.

The Intel® Management Module BMC does not persistently store DIMM fault and disabled status.

Note: If a DIMM is on a separate memory board, the memory board presence sensor provides an entity presence function that indicates whether the DIMM sensor should be ignored. This depends on the board presence.

4.19.6 System Memory Redundancy Monitoring

Some chipsets support memory redundancy features that go beyond single-bit error correction, allowing failing or failed DIMMs to be managed on-line without affecting normal system operation. BMC support for these is indicated below.

4.19.6.1 DIMM Sparing

If a platform supports DIMM sparing, the BMC will implement two sensors per DIMM sparing domain (the set of DIMMs which share a spare set of DIMMs). Each sparing domain will have an associated unique Entity ID. Both sensors will belong to that entity. Some server boards implement a single sparing domain, others implement multiple.

- DIMM Sparing Redundancy Sensor

This sensor is of type *Availability Status* (0Bh) and indicates whether the domain is redundant or not (i.e., whether there are spare DIMM(s) available for use). The supported offsets are:

- 00h: Fully Redundant
Both operational and enabled DIMMs and spares in domain
- 01h: Redundancy Lost: Sufficient Resources, from Redundant
Operational and enabled DIMMs in domain and no spares
- 05h: Non-Redundant: Insufficient Resources
No operational or disabled DIMMs in domain

- DIMM Sparing Enabled

This sensor is used to communicate the enabled state of the sparing feature for the associated domain. The sensor is of type *Entity Presence* (25h). This state of this sensor indicates whether the DIMM Sparing Redundancy sensor should be ignored (according to IPMI Entity Association rules about detection of entities – section 34.2 of the IPMI 2.0 specification). The supported offset is:

- 00h: Entity Present
If asserted, indicates that the DIMM sparing feature for this domain is enabled and the associated DIMM Sparing Redundancy Sensor state is valid. If not asserted, then the DIMM Sparing Redundancy Sensor should be ignored.
- 01h: Entity Absent
If asserted, indicates that the DIMM sparing feature for this domain is not available and the associated DIMM Sparing Redundancy Sensor state is invalid. This offset is mutually exclusive with offset 00h - Entity Present.

4.19.6.2 Memory Mirroring

If a server board supports Memory Mirroring, the BMC will implement two sensors per Memory Mirroring domain (the DIMMs which form a mirrored set). Each mirroring domain will have an associated unique Entity ID. Both sensors will belong to that Entity. Some platforms implement a single mirroring domain, others implement multiple.

- Memory Mirroring Redundancy Sensor

This sensor is of type *Availability Status* (0Bh) and indicates whether the domain is redundant or not (i.e., whether all mirrored DIMM(s) are available for use). The supported offsets are:

- 00h: Fully Redundant
All DIMMs in mirrored domain operational and enabled
- 01h: Redundancy Lost : Sufficient Resources
One or more failed DIMMs in one of the mirror pairs
- 05h: Non-Redundant : Insufficient Resources
Non-operational DIMMs in both of the mirror pairs

- Memory Mirroring Enabled

This sensor is used to communicate the enabled state of the mirroring feature for the associated domain. The sensor is of type *Entity Presence* (25h). This state of this sensor indicates whether the Memory Mirroring Redundancy sensor should be ignored (according to IPMI Entity Association rules about detection of entities – section 34.2 of the IPMI 2.0 specification). The supported offset is:

- 00h: Entity Present
If asserted, indicates that the Memory Mirroring feature for this domain is enabled and the associated Memory Mirroring Redundancy sensor state is valid. If not asserted, then the Memory Mirroring Redundancy sensor should be ignored.
- 01h: Entity Absent
If asserted, indicates that the Memory Mirroring feature for this domain is not available and the associated Memory Mirroring Redundancy Sensor state is invalid. This offset is mutually exclusive with offset 00h - Entity Present.

4.19.6.3 Memory RAID

If a server board supports Memory RAID, the BMC will implement two sensors per Memory RAID domain (the DIMMs which form a RAID set). Each RAID domain will have an associated unique Entity ID. Both sensors will belong to that Entity. Platforms that implement this feature only implement a single RAID domain.

- Memory RAID Redundancy Sensor

This sensor is of type *Availability Status* (0Bh) and indicates whether the domain is redundant or not (i.e., whether all DIMM(s) are available for use). The supported offsets are:

- 00h: Fully Redundant
All DIMMs in the RAID domain operational and enabled.
- 01h: Redundancy Lost : Sufficient Resources
One or more failed DIMMs in one of the mirror pairs
- 05h: Non-Redundant : Insufficient Resources
Non-operational DIMMs in both of the RAID sets

- Memory RAID Enabled

This sensor is used to communicate the enabled state of the RAID feature for the associated domain. The sensor is of type *Entity Presence* (25h). This state of this sensor indicates whether the Memory RAID Redundancy sensor should be ignored (according to IPMI Entity Association rules about detection of entities – section 34.2 of the IPMI 2.0 specification). The supported offset is:

- 00h: Entity Present
If asserted, indicates that the Memory RAID feature for this domain is enabled and the associated Memory RAID Redundancy sensor state is valid. If not asserted, then the Memory RAID Redundancy sensor should be ignored.
- 01h: Entity Absent
If asserted, indicates that the Memory RAID feature for this domain is not available and the associated Memory RAID Redundancy Sensor state is invalid. This offset is mutually exclusive with offset 00h - Entity Present.

4.19.7 Memory Hot Plug

On some systems, memory DIMMs can be removed or added while the system is operational (hot plug). The BMC needs to be kept up-to-date with the inventory changes and their implications on the Redundancy and entity presence sensors state.

Systems that currently support memory hot plug only support memory board hot plug, not individual DIMM hot plug. Each board supports multiple DIMMs and those DIMMs will then come and go as a set. Each memory board has an associated memory board presence sensor that provides inventory state for the board. That sensor also plays the role of the entity presence sensor that determines if the associated DIMM sensors should be ignored or not.

The BIOS manages the memory hot plug process. It updates the BMC on both memory board and DIMM population and status changes due to hot plug operations. Memory board presence changes are communicated to the BMC via the *Set DIMM State* command. A change in a memory board presence state of *present* to *not present* will put the associated DIMM sensors into an *init-in-progress* state. After updating memory board presence state, the BIOS will communicate the DIMM presence states for that board to the BMC (only on memory board insertions).

4.19.8 BIOS/BMC Interface for Memory RAS

The BMC relies on the BIOS to communicate Memory RAS state to the BMC.

4.19.8.1 Get DIMM State Command

System software can use the *Get DIMM State* command to determine the state of the Memory RAS features of the system. The states supported are:

- DIMM presence
- DIMM disabled state (DIMM not in use)
- DIMM failure state (had an uncorrectable ECC or other non-recoverable error)
- DIMM sparing state (DIMM is marked as spare)

DIMMs are grouped into DIMM Groups. Systems with onboard DIMM sockets, there will be only one DIMM Group (assuming the server board supports eight or fewer DIMMs). Systems with memory boards will have one DIMM Group per memory board. The *Get DIMM State* command supports being able to communicate DIMM Group (memory board) presence state. This will be used during a memory board hot plug operation.

The BMC logs SEL events for assertion and deassertion of all DIMM states except for deassertion of Fault offset.

4.19.8.2 Memory Board State

On platforms that have hot-pluggable memory boards, the BIOS will communicate the memory board presence state to the BMC. No other memory board state is supported. The BMC will log SEL events for assertion and deassertion of offsets associated with memory boards.

4.19.8.3 RAS State Commands

The BIOS communicates to the BMC during POST the enabled state of the various RAS domains (sparing, mirroring, RAID). This allows the BMC to enable or disable the appropriate sensor(s) that monitor memory redundancy.

Changes to Memory RAS domain redundancy state is communicated by the BIOS to the BMC. The BIOS detects changes in DIMM and memory board configuration and population and implements the policy determining whether a memory domain is redundant or not. The BMC just reports that state.

The BMC will log SEL events for assertion events of Redundancy state per event type 0x0B. See the sensor table in the appropriate platform-specific EPS for details on SEL logging associated with this sensor.

4.19.9 Single-bit ECC Error Throttling Prevention

The system detects, corrects, and logs correctable errors. As long as these errors occur infrequently, the system should continue to operate without a problem.

Occasionally, correctable errors are caused by a persistent failure of a single component. For example, a broken data line on a DIMM would exhibit repeated errors until replaced. Although these errors are correctable, continual calls to the error logger can throttle the system, preventing any further useful work.

For this reason, the system counts certain types of correctable errors and disables reporting if they occur too frequently. Correction remains enabled but calls to the error handler are disabled. This allows the system to continue running, despite a persistent correctable failure. The BIOS adds an entry to the event log to indicate that logging for that type of error has been disabled. Such an entry can indicate a serious hardware problem that should be repaired at the earliest possible time.

The system BIOS implements this feature for two types of errors, correctable memory errors and correctable bus errors. If ten errors occur in a single wall-clock hour, the corresponding error handler disables further reporting of that type of error. A unique counter is used for each type of error; i.e., an overrun of memory errors does not affect bus error reporting.

The BIOS re-enables logging and SMIs the next time the system is rebooted.

4.20 Hot Plug PCI Support

Some chassis and server boards provide support for one or more hot plug PCI slots. Cards in these slots can be removed and replaced while the system is operating normally. The BMC implements Hot Plug PCI sensors (sensor type *Slot/Connector (21h)*, event/reading type *Sensor Specific (6Fh)*) for each hot pluggable slot.

The supported sensor offsets are:

- 00h: Fault status asserted
- 02h: Slot/connector device installed/attached
- 03h: Slot/connector ready for device installation
- 04h: Slot/connector ready for device removal
- 05h: Slot power is off

The format of the event data 2 and event data 3 bytes are as specified in the IPMI specification. Event data 2 will be encoded as slot/connector type PCI slot (0). Event data 3 will hold the slot number. Slot numbers are server board specific; the numbers used by the BMC will match the server board silkscreen markings.

4.21 Event Logging Disabled Sensor

The BMC implements an *Event Logging Disabled* type (10h) sensor that is event only. The supported sensor offsets (dependent on server board) are:

- 00h: Correctable Memory Error Logging Disabled (server board dependent)
- 02h: Log Area (SEL) Reset/Clear (always supported)

4.22 Event Message Generation and Reception

The BMC cannot be configured to act as an event generator on the IPMB, so the BMC does not accept the *Set Event Receiver* command. The BMC does respond to the *Get Event Receiver* command.

4.23 BMC Self Test

The BMC performs various tests as part of its initialization. If a failure is determined (e.g., corrupt BMC SDR), the BMC stores the error internally. BMC or BMC sub-system failures detected during regular BMC operation may also be stored internally. Two commands may be used to retrieve the detected errors. The *IPMI 2.0 Get Self Test Results* command can be used to return the first error detected. The *Read Self Test* command can be used to sequentially read all the accumulated self test errors.

The following table shows self-test errors that may be posted.

Note: Not all tests are performed as part of the BMC initialization.

Table 16. BMC Self Test Results

First Byte	Second Byte	Description
55h	00h	No error
57h	01h	BMC operational code corrupted
57h	02h	BMC boot/firmware update code corrupted
57h	04h	BMC FRU internal use area corrupted
57h	08h	SDR repository empty
57h	10h	IPMB Signal Error
57h	20h	BMC FRU device inaccessible
57h	40h	BMC SDR repository inaccessible
57h	80h	BMC SEL device inaccessible
58h	00h	BMC RAM test error
58h	01h	BMC Fatal hardware error
C0h	XXh	Management controller XXh not responding
C1h	XXh	Private I ² C bus not responding (second byte xx indicates bus, bus0 = bit0)
C2h	XXh	BMC internal exception, XXh is the exception number
C3h	00h	BMC A/D timeout error
C3h	01h	SDR repository corrupt
C3h	02h	SEL corrupt
C3h	03h	PIA corrupted
C4h	XXh	I ² C Bus initialization error, XXh is the BMC physical bus number
C5h	00h	Platform type mismatch (i.e.; Server Board SE7520JR2 firmware running on IMM installed in the Server Board SE7520AF2)
EEh	XXh	Unhandled ARM Exception, XXh is the exception number: 01h = Undefined Instruction 03h = Prefetch Abort 04h = Data Abort 05h = ARM reserved exception 06h = Unhandled IRQ 07h = Unhandled FIQ

4.24 BMC and BIOS interaction with Error Logging

This section defines how errors are handled by the system BIOS. Also discussed is the role of the BIOS in error handling and the interaction between the BIOS, platform hardware, and server management firmware with regard to error handling. In addition, error-logging techniques are described and beep codes for errors are defined.

Chapter 6 of this document includes detailed information on error logging and information on event translations.

One of the major requirements of server management is to correctly and consistently handle system errors. System error sources can be categorized as follows:

- PCI bus
- Memory multi-bit errors (single-bit errors are not logged)
- Sensors
- Processor internal errors, bus/address errors, thermal trip errors, temperatures and voltages, and GTL voltage levels
- Errors detected during POST, logged as POST errors

When the Intel® Management Module is installed, the sensors are managed by the BMC. The BMC is capable of receiving event messages from individual sensors and logging system events

4.24.1 SMI Handler

The SMI handler handles and logs system-level events that are not visible to the server management firmware. If SEL error logging is disabled in the BIOS Setup utility, no SMI signals are generated on system errors. If error logging is enabled, the SMI handler preprocesses all system errors, even those that are normally considered to generate an NMI.

The SMI handler sends a command to the BMC to log the event and provides the data to be logged. For example, The BIOS programs the hardware to generate an SMI on a single-bit memory error and logs the location of the failed DIMM in the system event log.

4.24.2 PCI Bus Error

The PCI bus defines two error pins, PERR# and SERR#, for reporting PCI parity errors and system errors, respectively. The BIOS can be instructed to enable or disable reporting the PERR# and SERR# through NMI. Disabling NMI for PERR# and/or SERR# also disables logging of the corresponding event. In the case of PERR#, the PCI bus master has the option to retry the offending transaction, or to report it using SERR#. All other PCI-related errors are reported by SERR#. All the PCI-to-PCI bridges are configured so that they generate a SERR# on the primary interface whenever there is a SERR# on the secondary side, if SERR# has been enabled through Setup. The same is true for PERR#.

4.24.3 Processor Bus Error

If the chipset supports ECC on the processor bus then the BIOS enables the error correction and detection capabilities of the processors by setting appropriate bits in the processor model specific register (MSR) and appropriate bits inside the chipset.

In the case of unrecoverable errors on the host processor bus, proper execution of the asynchronous error handler (usually SMI) cannot be guaranteed and the handler cannot be relied upon to log such conditions. The handler will record the error to the SEL only if the system has not experienced a catastrophic failure that compromises the integrity of the handler.

4.24.4 Memory Bus Error

The hardware is programmed to generate an SMI on single-bit data errors in the memory array if ECC memory is installed. The SMI handler records the error and the DIMM location to the system event log. Double-bit errors in the memory array are mapped to the SMI because the mBMC cannot determine the location of the bad DIMM. The double-bit errors may have corrupted the contents of SMRAM. The SMI handler will log the failing DIMM number to the mBMC if the SMRAM contents are still valid. The ability to isolate the failure down to a single DIMM may not be available on certain platforms, and/or during early POST.

4.24.5 System Limit Error

The BMC monitors system operational limits. It manages the A/D converter, defining voltage and temperature limits as well as fan sensors and chassis intrusion. Any sensor values outside of specified limits are fully handled by the BMC. The BIOS does not generate an SMI to the host processor for these types of system events.

4.24.6 Processor Failure

The BIOS detects any processor BIST failures and logs the event. The failed processor can be identified by the first OEM data byte field in the log. For example, if processor 0 fails, the first OEM data byte will be 0. The BIOS depends upon the BMC to log the watchdog timer reset event.

If an operating system device driver is using the watchdog timer to detect software or hardware failures and that timer expires, an Asynchronous Reset (ASR) is generated, which is equivalent to a hard reset. The POST portion of the BIOS can query the BMC for a watchdog reset event as the system reboots, and then log this event in the SEL.

4.25 Light Guided Diagnostics

Intel® Server Boards and Platforms provide system fault/status LEDs in many areas of the platform. These can include, but are not limited to, fault LEDs for each DIMM slot and for each processor, and status LEDs for 5-volt stand-by and system state. Operation of some of these LEDs is dependant upon whether an Intel® Management Module is installed or not. The BMC works with the server BIOS to provide this LED support. LED functions supported by the Intel® Management Module and their behaviors are described below.

Note: Some platforms do not support these LED functions.

- The CPU 1 or CPU 2 led is lit to indicate the processor is disabled. DC-Off or AC cycle will cause the LED to turn off.
- CPU 1 and 2 LEDs are both lit to indicate the server board hardware has discovered a configuration error. If processor mis-population is detected when using standard onboard instrumentation, server board hardware will illuminate both processor error LEDs. The BMC will generate a series of beep codes when this condition is detected and will illuminate the processor 1 fault LED. An AC cycle will cause the LEDs to turn off.
- Fan fault LEDs are lit whenever the BMC detects a fan operating out of threshold.
- The 5-Volt stand-by LED is always lit when 5-volt stand-by is present.
- The Status LED displays the state of the system. It mirrors the state of the Control Panel Status LED. Valid states include: Solid Green, Blinking Green, Blinking Amber, Solid Amber, and Off.

To allow for system diagnostics, some LEDs are persistent and will stay on even after the condition that triggered the LED has passed. This includes A/C cycling the system. These persistent LEDs are cleared when the RESET button on the system control panel is pressed.

4.26 On-line Firmware Update

In addition to the Immediate Firmware Transfer Mode Update, the BMC also supports an Online Update feature. This feature allows a new BMC image to be copied into a special region of flash while the BMC continues to operate normally. At the next system reset, the BMC copies the downloaded image over to its execution region and then executes the new image.

The BMC also saves the previous firmware before updating itself with the new firmware. If the update fails the BMC can automatically roll back to the previous version. It can be commanded to rollback manually as well.

Operational code, PIA, and the SDR firmware areas can all be updated. Boot code updates are not permitted online. Configuration settings are not currently updated or rolled back. Any one or all of the supported firmware area types (Operational, PIA, and SDR) may be updated in a single update operation.

The BMC switches between the new and old images by storing three update images and copying firmware images between them. The images are:

- Execution Image: Contains the code currently being executed
- Staging Image: Where new code is downloaded
- Rollback Image: Where old code is saved in case a rollback is needed

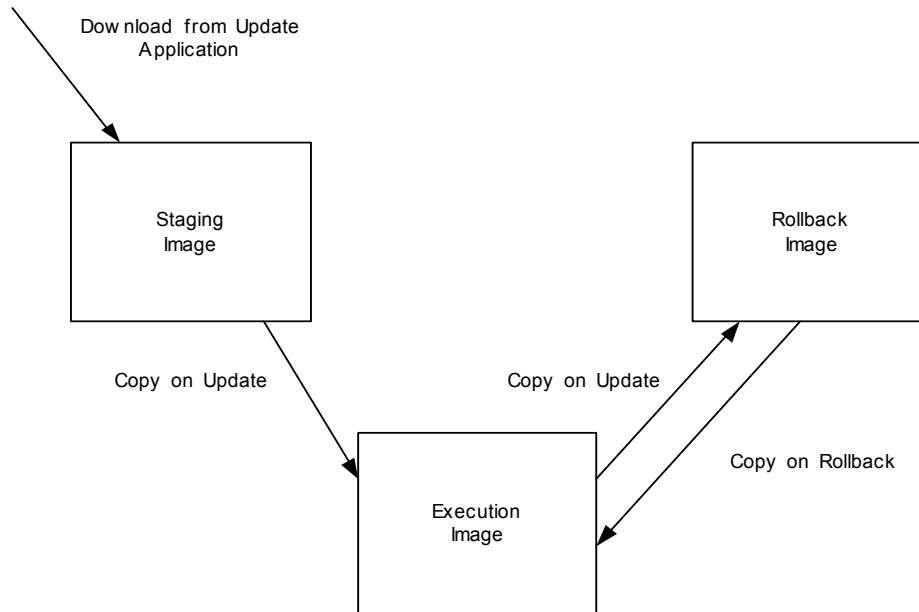


Figure 7. BMC Online Update Images

4.26.1 Online Update Flow

The sequence of events that occurs during a full update with rollback is as follows:

1. The online update application sends a command causing the BMC to re-initialize the Staging Image and enter the downloading state.
2. The online update application sends a command telling the BMC to create an area in the Staging Image and prepare to receive download data of the specified type.
3. The online update application updates the Staging Image.
4. The online update application sends a command (if doing a rollbackable update) to tell the BMC to save the current contents of the Execution Image.
5. The online update application sends a command to indicate the download is complete and is ready to be enacted. The BMC enters the Pending state.
6. A system reset occurs (potentially much later).

7. The BMC copies the current state of all areas in the Execution Image to the Rollback Image, regardless of which ones are actually being updated. This allows a self-consistent known good image to rollback to. It then copies the areas being updated from the Staging Image to the Execution Image.
8. The BMC validates the new firmware in the Execution Image and begins using it.
9. If the validation fails, the BMC copies the Rollback Image contents to the Execution Image and uses that firmware.

Note: System power cycles are treated the same as system resets. If A/C power is lost any time before the actual update copying process starts, the registered update is forgotten and when A/C power is eventually reapplied, the BMC comes up in the idle condition. If A/C power is lost during or after the copy process is started, it is resumed once A/C power is restored.

There can be one and only one area of each type in a single update sequence. Partial updates are not permitted – for example, if any PIA data is to be loaded the entire PIA must be loaded. Before each area is copied to the Execution Image, that entire region in the Execution Image is first erased. When an area is copied to the Rollback Image, the entire region is always copied.

4.26.2 Update Related SEL Logging

The BMC generates SEL entries whenever an Online Update is performed. The events are generated with the IPMI standard sensor type System Event, using the System Reconfigured offset (offset 0x00). The specific Online Update event is captured in the Event Data 2 byte of the SEL entry. If the BIOS and the BMC are updated together, two separate SEL events are still generated. A successful update followed by a manual rollback will generate two SEL entries (updated, then rolled back) whereas a failed update followed by an automatic rollback will generate a single SEL entry (rolled back).

Table 17. Online Update SEL Events

System Event Sensor, System Reconfigured Offset	
Event Description	Event Data 2 Value
BIOS Updated Successfully	0x01
BIOS Rolled back	0x02
BIOS Update Failed	0x03
BMC Updated Successfully	0x04
BMC Rolled back	0x05
BMC Update Failed	0x06

4.26.3 SDR Update Component of Online Update Process

The online update process for SDRs consists of the following steps.

Note: These actions are part of the whole Online Update process, which includes Operational and PIA updates as well.

1. The update application reads the current SDRs from the BMC.
2. These records are read using a sequence of standard Get SDR IPMI commands and stored in utility memory. As it reads the records in, the utility looks for the OEM SDR Tag records holding the tags describing the platform configuration and builds an internal database of tags.
3. The update application reads and edits the new SDR file provided in the update package.
4. The SDR file contains a sequence of SDR records and associated tags (if any). The update application discards SDR records whose tags do not match any in the internal tag database. Untagged records are kept.
5. The update application utility loads the newly generated set of SDRs into the BMC.
6. The SDR region of the FW update staged area is formatted as a contiguous set of binary SDR records with no gaps or alignment. The update application sends the new online update commands described above to send this contiguous set of binary SDR records to the BMC. It would also send down other areas being updated (Operational code and PIA) at this time.
7. The update application schedules the update.
8. A command is sent to the BMC to schedule the update.
9. On the next system reset, the BMC begins using the new SDRs.

4.27 Messaging Interfaces

This section describes the supported BMC communication interfaces:

- Host SMS Interface via Low Pin Count (LPC)/Keyboard Controller Style (KCS) interface
- Host SMM Interface via Low Pin Count (LPC)/Keyboard Controller Style (KCS) interface
- ACPI EC Interface via Low Pin Count (LPC)/Keyboard Controller Style (KCS) interface
- Intelligent Platform Management Bus (IPMB) I²C interface
- PCI SMBus
- Emergency Management Port (EMP) using the IPMI over Serial/Modem protocols for serial remote access
- LAN interface using the IPMI over LAN protocols
- Intelligent Chassis Management Bus using the IPMI ICMB protocol

These specifications are defined in the following subsections.

4.27.1 Channel Management

Every messaging interface is assigned an IPMI channel ID by *IPMI 2.0*. There are commands to configure each channel for privilege levels and access modes. The following table shows the standard channel assignments:

Table 18. Standard Channel Assignments

Channel Id	Interface	Supports Sessions
0	IPMB	No
1	LAN 1	Yes
2	LAN 2 1	Yes
3	LAN 31	Yes
4	EMP (Basic/PPP)	Yes
5	ICMB	No
6	PCI SMBus 1	No
7	SMM	No
0Eh	Self2	-
0Fh	SMS / Receive Message Queue	No

¹ If supported by the server board or platform

² Refers to the actual channel used to send the request

4.27.2 User Model

The BMC supports the IPMI 2.0 user model including *User ID 1* support. The BMC supports a maximum of four users.

4.27.3 Media Bridging

The BMC supports bridging between the EMP and IPMB interfaces and between the LAN and IPMB interfaces. This allows the state of other intelligent controllers in the chassis to be queried by remote console software. Requests may be directed to controllers on the IPMB, but requests originating on the IPMB cannot be directed to the EMP or LAN interfaces.

4.27.4 Request/Response Protocol

All of the protocols used in the above mentioned interfaces are Request/Response protocols. A Request Message is issued to an intelligent device, to which the device responds with a separate Response Message. For example, with respect to the IPMB interface, both Request Messages and Response Messages are transmitted on the bus using I²C Master Write transfers. That is, a Request Message is issued from an intelligent device acting as an I²C master, and is received by an intelligent device as an I²C slave. The corresponding Response Message is issued from the responding intelligent device as an I²C master, and is received by the request originator as an I²C slave.

4.27.5 Native CLI

The native CLI provides a command line interface in EMP and LAN channels to communicate with the BMC through scripting. Native CLI syntax is described in section 5.5.2.

4.27.6 Host to BMC Communication Interface

4.27.6.1 LPC/KCS Interface Overview

The Intel® Management Module BMC interfaces to its host system via the low pin count (LPC) bus. The BMC incorporates three 8042* keyboard controller style (KCS) ports. Only one of these ports is available for use by system software. The remaining two ports are used for the system management mode interface, and the ACPI embedded controller interface.

The KCS ports provided by the BMC reside at specific I/O addresses on the host's LPC bus. These ports are accessible only by the LPC host. Other LPC masters should not access them.

These interfaces are assigned the following uses:

Interface#	Name	Use
0	SMS Interface	SMS, BIOS POST, and utility access
1	SMM Interface	SMI handling for error logging
2	ACPI Embedded Controller (EC) Interface	Power management communications

The nominal I/O addresses for these interfaces are specified in Table 19. The SMBIOS tables may be used to locate the system interface as described in the IPMI 2.0 specification.

Table 19. KCS Interface Addresses

Interface Name	Address
SMS	0CA2h – 0CA3h
SMM	0CA4h – 0CA5h
ACPI EC	0CA6h – 0CA7h

The BMC gives higher priority to transfers occurring through the SMM interface. This provides minimum latency during SMI accesses. The BMC acts as a bridge between the SMS and the IPMB interfaces. Interface registers provide a mechanism for communications between the BMC and the host system.

4.27.6.2 Receive Message Queue

The Receive Message Queue is only accessible via the SMS interface since that interface is the BMC's host/system interface. The queue size is two entries.

4.27.6.3 SMS/SMM Status Register

Bits in the status register provide interface and protocol state information. As an extension to the IPMI 2.0 KCS interface definition, the OEM1 and OEM2 bits in the SMS and SMM interfaces have been defined to provide BMC status information. Table 20 summarizes the functions of the Status Register bits. Read/Write is from the perspective of the host interface. All status register bits are read-only to the host.

Table 20. SMS/SMM Status Register Bits

Bit	Name	Description
7	S1	Bits 7 and 6 indicate the current state of this KCS interface. The host software examines these bits to verify that they are in sync with the BMC. For more information on these bits, see the IPMI 2.0 specification.
6	S0	
5	BMC State 1 (OEM2)	These bits provide a status indication of BMC health: 00b – BMC Ready 01b – BMC Hardware Error (e.g., BMC memory test error) 10b – BMC FW Checksum Error 11b – BMC is Not Ready
4	BMC State 0 (OEM1)	
3	C/D#	
2	SMS_ATN / SMM_ATN	When status register is used for SMS interface, SMS_ATN bit indicates that the BMC has a message for the SMS. When status register is used for SMM interface, SMM_ATN bit indicates that the BMC has a message for the SMI handler. Set to 1 when the BMC has a message for the SMS/SMI handler. See sections 4.27.6.4 and 4.27.6.5 for more details on these flag bits.
1	IBF	Input buffer full. Set to 1 when either the associated command or Data_In register has been written by system-side software. Cleared to 0 by the BMC reading the data register.
0	OBF	Output buffer full. Set to 1 when the associated Data_Out register is written by the BMC. Cleared to 0 by the host reading the data register.

Note: When the BMC is reset (from power on or a hard reset), the protocol state bits (S0,S1) are initialized to 11b, Error State, and the BMC state bits (BMC State 0/1) are initialized to 00b, BMC Ready. This allows host software to detect that the BMC has been reset and that the BMC has terminated any in-process messages.

The BMC state bits will be set to 11b, BMC Not Ready, if the BMC is busy; such as during SEL or SDR erasure or while the Initialization Agent is running.

4.27.6.4 SMS Interface

The SMS interface is the BMC host interface. The BMC implements the SMS KCS interface as described in the IPMI 2.0 specification. The BMC implements the optional *Get Status/Abort* transaction on this interface. Only LUN 0 is supported on this interface. In addition the status register OEM1/2 bits have been defined as specified in section 4.27.6.3.

If so configured via the *Set BMC Global Enables* command, the BMC can generate an interrupt requesting attention when setting the SMS_ATN bit in the status register.

The SMS_ATN bit being set indicates one or more of the following:

- At least one message is in the BMC Receive Message Queue
- There is an event in the Event Message Buffer
- Watchdog Pre-timeout interrupt flag set

All conditions must be cleared (and all BMC to SMS messages must be flushed) for the SMS_ATN bit to be cleared.

The host I/O address of the SMS interface is nominally *0CA2h – 0CA3h*, but this address assignment may be overridden by the server board BMC EPS.

The operation of the SMS interface is described in detail in the *Intelligent Platform Management Interface Specification* in the chapter titled Keyboard Controller Style (KCS) Interface.

4.27.6.4.1 Canceling In-progress Commands

It is allowable for software to cancel an in-progress transaction by issuing a new WRITE_START command to the interface. However, there are cases where the BMC has accepted the command already and queued it for execution. These commands will be executed even if the transaction has been canceled.

Since the SMS interface is single-threaded, the BMC will not accept a new command until the current canceled in-progress command has completed execution. Until that time, any new command sent via the SMS interface will be responded to with a NODE_BUSY completion code. When the current canceled in-progress command has completed, the BMC discards the response and the SMS interface will again accept commands for execution.

4.27.6.5 SMM Interface

The SMM interface is a KCS interface that is used by the BIOS when interface response time is a concern, e.g., in the BIOS SMI Handler. The BMC gives this interface priority over other communication interfaces.

4.27.6.6 ACPI Embedded Controller Interface

The ACPI/EC is used by the BIOS to communicate ACPI states to the BMC.

4.27.7 IPMB Communication Interface

The IPMB is a communication protocol that utilizes the 100 KB/s version of an I²C bus as its physical medium. The IPMB implementation in the BMC is compliant with the *IPMB v1.0, revision 1.0*.

The BMC IPMB slave address is 20h.

The BMC both sends and receives IPMB messages over the IPMB interface. Non-IPMB messages received via the IPMB interface are discarded.

Messages sent by the BMC can be either originated by the BMC (e.g., due to Init Agent operation) or on behalf of another source (e.g., due to a *Send Message* command with IPMB channel number issued by SMS).

For BMC originated IPMB request messages, the BMC implements a response timeout interval of 60 ms and a retry count of 3.

4.27.7.1 PCI System Management Bus (SMBus)

Some platforms support BMC access to the PCI SMBus. IPMB messaging is supported on the PCI SMBus. However, the packet format has been modified to help to disambiguate between IPMB and SMBus traffic. If a server board supports PCI SMBus access, then it must support command access as on the IPMB.

4.27.7.2 BMC as I²C Master Controller on IPMB

The BMC allows access to devices on the IPMB as an I²C master. The following commands are supported:

- *Send message*
This command writes data to an I²C device as master.
- *Master Write-Read I²C*
This command allows the following actions:
 - Writing data to an I²C device as a master
 - Reading data from an I²C device as a master
 - Writing data to I²C device as a master, issue an I²C Repeated Start, and reading a specified number of bytes from I²C device as a master. Errors in I²C transmission or reception are communicated via completion codes in the command response.

These functions support the most common operations for an I²C master controller. This includes access to common non-intelligent I²C devices like EEPROMs. The *Send Message* command is normally used to send IPMB messages to intelligent devices that utilize the IPMB protocol.

4.27.7.3 IPMB LUN Routing

The BMC can receive either request or response IPMB messages. The treatment of these messages depends on the destination Logical Unit Number (LUN) in the IPMB message. For IPMB request messages, the destination LUN is the responder's LUN. For IPMB response messages, the destination LUN is the requester's LUN. The disposition of these messages is described in Table 21. The BMC accepts LUN 00b and LUN 10b.

IPMB messages can be up to 32 bytes long.

Table 21. BMC IPMB LUN Routing

LUN	Name	Message Disposition
00b	BMC	Request messages with this LUN are passed to the BMC command handler for execution. Response messages with this LUN are compared with outstanding BMC originated requests. If there is a match, the BMC subsystem that sent the request is notified. Otherwise the message is discarded.
01b	Reserved	Reserved – All messages arriving with this LUN are discarded.
10b	SMS	All messages arriving with this destination LUN are placed in the Receive Message Queue. If that buffer is full, the message is discarded. No further processing or response is done.
11b	Reserved	Reserved – All messages arriving with this LUN are discarded.

Figure 8 shows a logical block diagram of the BMC receiving IPMB messages.

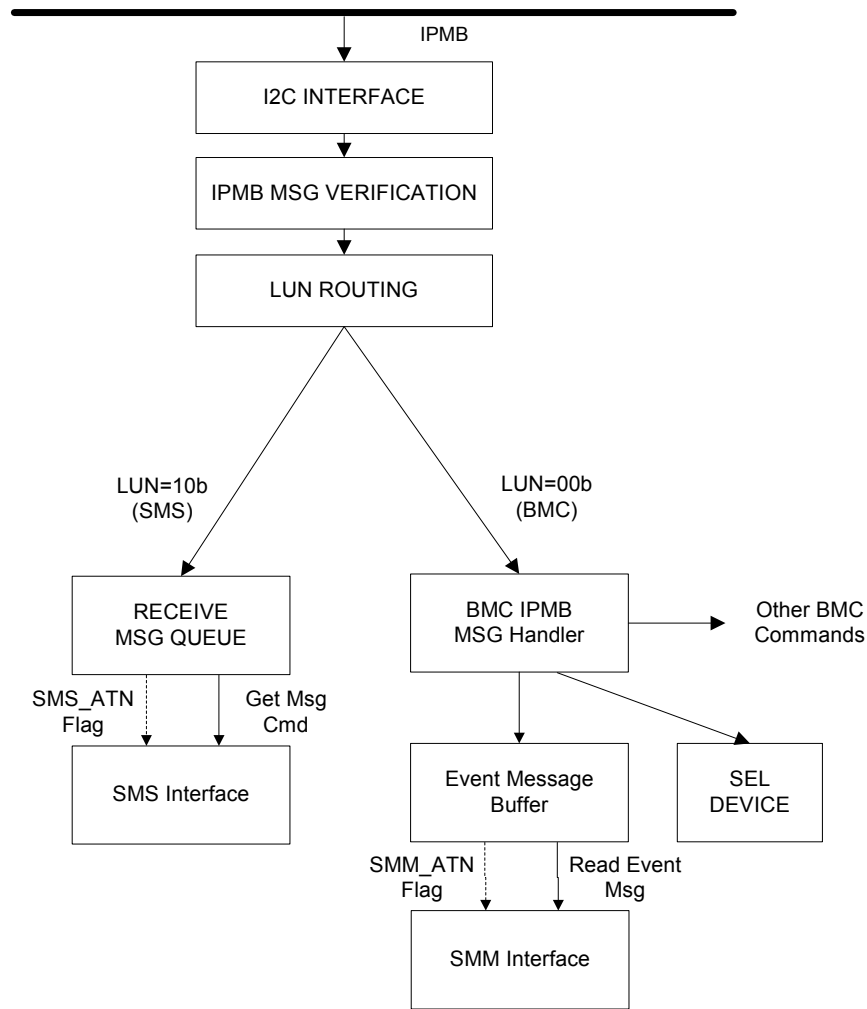


Figure 8. BMC IPMB Message Reception

4.27.8 EMP Interface

The EMP interface is the Intel implementation of the IPMI 2.0 IPMI over Serial/Modem feature.

The out-of-band RS232 connection provides system administrators with the ability to access low-level server management firmware functions by using commonly available tools. To make it easy to use and provide high compatibility with LAN and IPMB protocols, this protocol design adopts some features of both protocols.

The Intel implementation shares EMP function with the server board's COM2 interface. The BMC has control over which agent (BMC or System) has access to COM2. Hardware handshaking is supported as is the *Ring Indicate* and *Data Carrier Detect* signals.

The Basic and PPP/UDP with PPP/UDP Proxy modes of IPMI over Serial/Modem are supported and are available whether the system has DC power on or not. Both Basic and PPP/UDP modes support the callback feature, providing another level of security.

Additional details on the IPMI over Serial/Modem interface can be obtained from the *IPMI 2.0 Specification*.

4.27.8.1 COM2 Port Switching

As specified in the *IPMI 2.0 Specification*, if EMP is enabled, the BMC will watch the serial traffic when the COM2 port is owned by the system. This is done to respond to in-band port switching requests. Serial traffic snooping is done both in Basic and PPP/UDP modes, although the exact sequences recognized in the two modes is different.

4.27.8.2 Callback

The BMC supports the callback feature and its associated configuration parameters. Callback is initiated by the *Callback* command. Normally, one user is configured with only callback privileges and another is configured to require a callback to be active. After phoning in, the first user ID is used to trigger the callback. Then, once the callback connection is established, the remote side establishes a session using the second user ID.

In order to support this feature, the BMC implements the following Serial/Modem parameters. See the server board BMC EPS for more information.

- Number of Dial Strings (shared with Dial Page Alerting)
- Number of Serial/Modem Alert Destinations (shared with Dial Page Alerting)
- Number of Serial/Modem Destination IP Addresses (PPP/UDP mode only)
- Number of PPP Accounts (PPP/UDP mode only)

4.27.8.3 Basic Mode

Basic mode uses a protocol where IPMI requests are transmitted over the serial port with minimal framing, providing a low overhead implementation. Command receipt by the BMC is acknowledged on a per-packet basis.

Basic mode minimally supports *None* (un-authenticated) and *Straight Password/Key* authentication types. Basic mode supports Callback.

4.27.8.4 PPP/UDP Mode

PPP/UDP mode allows standard PPP protocol sessions to be established with the BMC via the EMP port. Once a PPP session is established, communication over the interface works just like the LAN interface with UDP packets being delivered and interpreted as described in the *IPMI 2.0 specification* IPMI LAN Interface section.

The BMC supports *None* (no authentication), *MD2*, and *MD5* authentication types over the PPP/UDP interface.

Callback is supported in PPP/UDP mode.

4.27.8.5 PPP Proxy

The BMC provides commands that allow host side applications to send and receive UDP packets via an established BMC PPP connection. See the *IPMI 2.0 specification* for more details.

4.27.8.6 Terminal Mode

The BMC supports Terminal Mode as specified in the *IPMI 2.0 specification*. Terminal Mode provides a printable ASCII text-based mechanism for delivering IPMI messages to the BMC over the serial channel or any packet-based interface. Messages can be delivered in two forms:

- Via hex-ASCII pair encoded IPMI commands
- Via text “SYS” commands

4.27.8.6.1 **Input Restrictions**

- **Maximum Input Length**
The BMC supports a maximum of 122 characters per line. The BMC will stop accepting new characters and stop echoing input when the 122 character limit is reached. Some characters will still be accepted and handled appropriately even after the character limit is reached. These are the <ESC>, <backspace>/<delete>, illegal, and input <newline> characters.
- **Maximum IPMI Message Length**
The terminal mode interface supports a maximum IPMI message length of 40 bytes.
- **Line Continuation Character**
The line continuation character is supported over the serial channel in terminal mode only. The line continuation character is supported for both hex-ASCII and text commands.

4.27.8.6.2 **Command Support**

- **Text Commands:**
The BMC supports all the text commands described in the IPMI 2.0 specification as well as the OEM text commands described in Table 73.
- **Text Command Privilege Levels**
The BMC supports the privilege level scheme for terminal mode text commands as specified in Table 72.

4.27.8.6.3 **Bridging Support**

The BMC utilized the optional bridging functionality described in the IPMI 2.0 specification as part of the native CLI command support. Bridging support is limited to communicating requests for graceful operating system shutdown from the BMC to an operating system resident agent in Intel® Server Management 8.

4.27.8.7 **Native CLI over Serial**

It is possible to use the Native CLI command set over EMP by configuring the channel to interpret the text commands in Native CLI syntax. This is done by setting the Serial Channel *CLI Syntax* parameter (# 193) described in Table 71.

4.27.8.8 **Invalid Password Handling**

If three successive invalid *Activate Session* commands are received on the EMP interface, the BMC will send the 'hang-up' sequence if in modem mode, and delay 30 seconds before accepting another *Activate Session* command. The BMC will also log an 'Out-of-band Access Password Violation' event to the System Event Log each time an invalid *Activate Session* command is received.

4.27.8.9 Serial Ping Message Behavior

The IPMI 2.0 specification *Serial/Modem Connection Active* command, or “serial ping”, is output by the BMC over the serial connection at a rate of once every two seconds when Basic Mode is enabled in the EMP configuration settings. However, serial ping message behavior can be different depending on the types of operating modes enabled.

4.27.8.9.1 Operating Mode (Connection Mode Enable) Auto-Detection

If Basic Mode and at least one other operating mode are enabled, and the BMC is currently auto-detecting the operating mode, the serial ping will be output by the BMC until an operating mode is detected. If the operating mode eventually detected is not Basic Mode, the serial pings will be automatically disabled. Upon connection loss, session loss, or session closure, the serial pings will be automatically re-enabled based upon the EMP configuration settings.

4.27.8.9.2 Terminal Mode

If Terminal Mode is the only operating mode enabled in the EMP configuration settings, the serial ping is automatically always disabled regardless of the EMP configuration settings. If the EMP configuration changes, the serial ping message behavior will automatically be re-evaluated as necessary.

4.27.9 LAN Interface

The BMC implements both the IPMI 1.5 and IPMI 2.0 messaging models. These provide out-of-band Local Area Network (LAN) communication between the BMC and the external world. The BMC uses the TCO (Total Cost of Ownership) management port on one or more of the server board Ethernet interfaces or dedicated NICs.

See the *IPMI 2.0 Specification* for details on the IPMI over LAN protocol and the server board BMC EPS for details on the LAN interface implementation for that server board, which should detail at least the following:

- The number of supported LAN interfaces
- Communication UDP port numbers
- ARP/DHCP support
- Channel assignments
- Number of supported sessions

Run-time determination of LAN channel capabilities can be determined by both standard IPMI defined mechanisms as well as an OEM configuration parameter that defines advanced feature support.

4.27.9.1 IPMI 1.5 Messaging

The communication protocol packet format consists of IPMI requests and responses encapsulated in an IPMI session wrapper (for authentication) wrapped in an RMCP packet, which is wrapped in an IP/UDP packet. Although authentication is provided, no encryption is provided, so administrating some settings such as user passwords via this interface is not advised.

Session establishment commands are IPMI commands that don't require authentication or an associated session.

The BMC supports *None* (no authentication), *Straight Password/Key*, *MD2*, and *MD5* authentication types over the LAN interface.

4.27.9.2 IPMI 2.0 Messaging

IPMI 2.0 messaging is built over RMCP+ and has a different session establishment protocol. The session commands are defined by RSSP and implemented at the RMCP+ level, not IPMI commands. Authentication is implemented at the RMCP+ level. RMCP+ provides link payload encryption, so it is possible to communicate private/sensitive data (confidentiality).

The BMC supports the following Cipher Suites:

Table 22. Supported RMCP+ Cipher Suites

ID	Authentication Algorithm	Integrity Algorithm(s)	Confidentiality Algorithm(s)
0	RAKP-none	None	None
1	RAKP-HMAC-SHA1	None	None
2	RAKP-HMAC-SHA1	HMAC-SHA1-96	None
3	RAKP-HMAC-SHA1	HMAC-SHA1-96	AES-CBC-128
6	RAKP-HMAC-MD5	None	None
7	RAKP-HMAC-MD5	HMAC-MD5-128	None
8	RAKP-HMAC-MD5	HMAC-MD5-128	AES-CBC-128
11	RAKP-HMAC-MD5	MD5-128	None
12	RAKP-HMAC-MD5	MD5-128	AES-CBC-128

For user authentication, the BMC can be configured with 'null' user names, whereby password/key lookup is done based on 'privilege level only', or with non-null user names, where the key lookup for the session is determined according to the user name.

IPMI 2.0 messaging introduces the concept of Payload Types and Payload IDs. This allows other types of data to be transferred than IPMI commands. IPMI 2.0 Serial Over LAN is implemented as a Payload Type. The Intel proprietary Keyboard/Video/Mouse redirection feature of the Intel® Management Module - Advanced Edition is implemented as both OEM Payload types and specific Payload Ids as OEM Explicit.

Table 23. Supported RMCP+ Payload Types

Payload Type	Feature	IANA
00h	IPMI Message	N/A
01h	Serial Over LAN	N/A
02h	OEM Explicit	Intel (343)
10h – 15h	Session Setup	N/A
20h ¹	Intel OEM - Video Redirection	N/A
21h ¹	Intel OEM - Keyboard/Mouse Redirection	N/A

1. Advanced Version only

4.27.9.3 RMCP/ASF Messaging

The BMC supports the 'RMCP Ping Discovery' mechanism where it responds with a 'Pong' message for a RMCP/ASF ping request. This is implemented as per the *IPMI 2.0* specification,

4.27.9.4 Shared LAN Channels

Some LAN channels use NICs that are shared with the system, meaning that the NIC MAC and IP addresses are shared with the Operating System.

These channels provide access to the BMC firmware over the enterprise network without requiring a dedicated server management Ethernet interface.

For these channels, the BMC supports IPMI over LAN, but cannot implement ARP, DHCP, or any other protocol other than RMCP over IP. ARP support is implemented via Gratuitous ARP support (sending out of periodic unrequested ARP responses).

On most implementations of Shared LAN Channels, the BMC cannot read the NIC MAC addresses. These must be provided by the BIOS during POST.

If DHCP configuration of the BMC LAN Channel IP address is required, the BIOS must perform the DHCP operation and set the LAN Channel configuration parameters appropriately based on the results. These parameters are:

- IP address
- Subnet Mask
- Default Gateway IP Address
- Default Gateway MAC Address

The BIOS can tell whether to do DHCP on a LAN Channel by looking at the *IP Address Source* (4) LAN configuration parameter for each channel. The BIOS should do DHCP if the value of the parameter is *Address loaded by BIOS or system software* (3h). A set of OEM parameters are implemented per-channel to aid in BIOS/BMC DHCP.

The BMC supports three OEM LAN configuration parameters:

- DHCP Server IP Address
- DHCP Server MAC Address
- DHCP Enable

These values are stored by the BMC for use by the BIOS and are not interpreted by the BMC. They are documented in Table 69.

Neither the BIOS nor the BMC can keep a DHCP acquired IP address renewed during system operation. An operating system-based client is necessary to synchronize the BMC with IP address changes.

4.27.9.5 Dedicated LAN Channels

In some configurations, the BMC has one or more dedicated LAN channels. This is either via a NIC that is purely used by the BMC (invisible to the BIOS and operating system), or via a system NIC that provides a separate MAC address for server management use. The BMC has access to the NIC MAC address, so the BIOS is not required to provide it.

This type of channel supports all of the advanced features. In addition, they support ARP and DHCP.

Changing the IP address of these dedicated LAN channels while advanced features are currently active is not supported.

4.27.9.5.1 ARP

Dedicated LAN Channels support responding to received ARP requests. In addition, they support ARP resolution of configured IP addresses. If the Default Gateway or Alert Destination MAC addresses in the LAN configuration are set to 00:00:00:00:00:00, the BMC will use ARP to resolve the associate IP address. The BMC does not issue Gratuitous ARPs over dedicated LAN channels.

4.27.9.5.2 DHCP

Dedicated LAN Channels support BMC implemented DHCP. If the *IP Address Source (4) LAN* configuration parameter is set to *Address obtained by BMC running DHCP (2h)*, the BMC will run DHCP and set the channel *IP address*, *Subnet Mask*, and *Default Gateway IP Address* configuration parameters.

4.27.9.6 VLAN Support

The Intel® Management Module does not implement the IPMI 2.0 method for encapsulating Server Management LAN traffic in a Virtual LAN (VLAN) over a physical LAN as defined by the IEEE 802.1q specification.

4.27.9.7 Serial Over LAN (SOL)

The BMC supports both the previous generation Intel proprietary SOL (now known as SOL 1.0) as well as the IPMI 2.0 defined SOL feature. SOL redirection performance may be impacted when BMC network bandwidth is not available i.e. when a KVM session is active

4.27.9.7.1 SOL 1.0

The BMC supports the Intel proprietary Serial over LAN (SOL 1.0) features. This is included for legacy software support. The SOL feature provides for bi-directional transport of system COM2 serial data encapsulated in IPMI over LAN packets. This provides out-of-band LAN access to BIOS console redirection, service partition application communication, or operating system console interaction (e.g., Linux/Unix) without the BIOS or software being LAN enabled or aware of other than normal serial port interaction.

The SOL feature includes commands to configure the feature (*Get/Set SOL 1.0 Configuration Parameters*) and activate SOL for a LAN session (*Activate SOL*). Table 85 specifies the supported configuration parameters.

4.27.9.7.2 SOL 2.0

IPMI 2.0 introduces a standard Serial Over LAN feature. This feature is implemented as a standard Payload type (01h) over RMCP+. System software developers should develop their software applications based on the IPMI 2.0 SOL feature.

Three commands are implemented for SOL 2.0 configuration.

- *Get/Set SOL 2.0 Configuration Parameters*
These commands are used to get and set the values of the SOL configuration parameters. The parameters are implemented on a per-channel basis. See Table 85 for details of supported SOL parameters.
- *Activating SOL*
This command is not accepted by the BMC, but sent by the BMC over the Serial channel if there is an active session when SOL is activated, to notify a remote client of the switch to SOL.

4.27.10 ICMB Interface

The Intelligent Chassis Management Bus (ICMB) defines a character-level transport for inter-chassis communications between intelligent chassis. This includes the ability to use the ICMB to bridge messages from the IPMB in one chassis to the IPMB in another, as shown in Figure 4.

Physically, the ICMB is a multi-drop, multi-master, 2-wire, half-duplex, differential drive bus, utilizing RS-485 transceivers. At any given time, only one chassis can be driving the bus. Each must arbitrate to gain control of the bus when it has something to send. ICMB messages are IPMI compatible with an implicit Net Function of Bridge. See the *Intelligent Chassis Management Bus, Version 1.0, Revision 1.20* for the definition of commands and responses.

Note: Some server systems provide the ICMB interface as a standard feature of the board set. Other systems may offer similar functionality, but with an add-in transceiver card.

On previous platforms, the ICMB feature had been implemented in a standalone Chassis Bridge Controller (CBC) or as part of a secondary controller such as the Front Panel Controller (FPC). It is now integrated directly into the Intel® Management Module BMC. In this implementation, the ICMB Bridge Device functionality appears to system software as if there was a separate ICMB Bridge Controller on a physical IPMB. The BMC implements the ICMB Bridge Device functions internally on a 'virtual IPMB'. This provides backward-compatibility with software that was designed to work with a non-integrated ICMB Bridge Device.

The BMC reports that the Chassis Bridge Device is not part of the BMC by returning an address in the *Get Chassis Capabilities* command (0x28) that is different than the BMC address (0x20). This indicates to software that it needs to access the Bridge Device function by using *Send Message* commands to deliver messages to the Bridge Device via the primary IPMB. The BMC then monitors the *Send Message* command for messages directed to the Bridge Device address.

When the BMC sees *Send Message* commands to the Bridge Device address, it handles them internally instead of routing them out to a physical IPMB. Responses from the virtual Bridge Device are placed into the *Receive Message Queue* as if they were received from the IPMB. The Virtual IPMB can only be accessed using *Send Message/Receive Message* commands. If commands are sent over the physical IPMB to the Bridge Device address (0x28), the command request will be NAK'ed.

To maintain compatibility with system software that treats the bridge device as a separate entity, in addition to the bridge commands defined in the ICMB specification, the Integrated ICMB module responds to some standard IPMI commands. These are *Get Device Id*, *Get Self Test Results*, and *Set ACPI Power State*.

The *Get Self Test Results* command always returns OK since the Integrated ICMB module has no self-tests of its own. The *Set ACPI Power State* command causes the Integrated ICMB module to generate an ICMB event message informing external management software of the chassis's power state change.

4.27.10.1 ICMB Addresses

Every ICMB bridge node on the bus has a unique two-byte address. The ICMB address is dynamically assigned by the ICMB data-link protocol on installation or if an address conflict is detected. The bridge can pick any address except some reserved ones, such as the broadcast address (0000h). After the bridge address is assigned, it is saved in Non-Volatile Store. The bridge will re-use the same address whenever it comes on-line until it detects an address conflict or the address is changed with the *Set ICMB Address* command. The address is a ConfigAPI-maintained data item as well, so the address will be preserved across firmware updates.

4.27.10.2 ICMB Bridge Response Packet Format

The Integrated ICMB function implements the ICMB v1.0 protocol. Backwards compatibility is also maintained to a degree with the older v0.9 as described below.

The format of an ICMB response packet (on the ICMB bus) to a *Bridge Request* command is different between ICMB v1.0 and ICMB v0.9.

The v1.0 response data format provides the remote bridge's completion code, IPMB responder's slave address, IPMB net function/LUN, and IPMB command in the response packet data. The v0.9 format only provides the IPMB net function/LUN and command.

Table 24. ICMB Bridge Request Response Data Packet Format

ICMB v0.9	X	X	IPMB Nf/LUN	IPMB Cmd	IPMB Comp code	IPMB Resp Data
ICMB v1.0	Bridge Comp code	IPMB RsSA	IPMB Nf/LUN	IPMB Cmd	IPMB Comp code	IPMB Resp Data

The returned IPMB net function in the v1.0 response is converted into the appropriate response net function. In the v0.9 response, the original request net function is echoed back.

This Integrated ICMB module has been written to accept v0.9 bridge request responses allowing a system with v1.0 firmware to manage a system with a v0.9 CBC. However, the v0.9 CBC will not be able to correctly interpret the v1.0 *Bridge Request* responses and so cannot be used to manage a system with v1.0 firmware.

4.27.10.3 Virtual IPMB Bridge Response Format

The format of an IPMB response packet to a bridged *Bridge Request* command is different between ICMB v1.0 and ICMB v0.9.

In ICMB v0.9, the response data consists of the local bridge's completion code, the remote IPMB net function / LUN, the remote IPMB command, the remote IPMB completion code, and the remote IPMB response data.

ICMB v1.0 deletes the remote IPMB net function / LUN and remote IPMB command, and adds the remote bridge's completion code.

Table 25. IPMB Bridge Request Response Data Format

ICMB v0.9	Local Bridge Comp code	X	Remote IPMB Nf/LUN	Remote IPMB Cmd	Remote IPMB Comp code	Remote IPMB Resp Data
ICMB v1.0	Local Bridge Comp code	Remote Bridge Comp code	X	X	Remote IPMB Comp code	Remote IPMB Resp Data

Software communicating with ICMB bridges via IPMB will have to interpret bridged *Bridge Request* responses differently depending on the IPMI revision level of the controller.

4.27.10.4 Chassis Bridge LUNs

The Integrated ICMB supports LUNs (Logical Units) of 00b and 01b. In essence, LUNs allow multiple logical intelligent nodes to be implemented behind a single slave address. Each LUN can have its own unique command and NetFns that it responds to. A request to an invalid LUN will be rejected.

Table 26. LUN Table

LUN	Name	Description
00b	Bridge	This LUN is the normal bridging request messages for accessing the Integrated ICMB functions.
01b	ICMB Proxy	Remote IPMB Device Proxy (ICMB Proxy).
10b	-	Not valid
11b	-	Not valid

The ICMB Proxy LUN on the Bridge Device implements an IPMB bridge proxy device allowing maximum-length IPMB messages to be bridged to a remote chassis, something not possible with the bridging commands described below because of request encapsulation. Once configured with the remote chassis address and remote IPMB address any command sent to LUN 1 on the Bridge Device will be bridged to the specified destination IPMB node.

Note: The ICMB Proxy LUN is only used on requests that are sent to the Bridge Device address (28h). Requests sent to the Chassis Device address (20h) with a LUN of 1 are routed to the SMS Receive Message Queue. See the ICMB specification for details of the format of proxy-ed messages.

4.28 Event Filtering and Alerting

The BMC implements most of the IPMI 2.0 alerting features. The following features are supported:

- PEF
- Dial Paging
- Alert over LAN
- Alert over Serial/PPP

4.28.1 Platform Event Filtering (PEF)

The BMC monitors server board health and logs failure events into the SEL. The Platform Event Filtering (PEF) feature provides a configurable mechanism to allow events to trigger alert actions. PEF provides a flexible, general mechanism that enables the BMC to perform selectable actions triggered by a configurable set of platform events. The BMC supports the following PEF actions:

- Power Off
- Power Cycle
- Reset
- Diagnostic Interrupt
- OEM Action
- Alerts

The following PEF resource sizes are platform-specific. See the server board BMC EPS for the values.

- Event Filter Table Entry Count
- Alert Policy Table Entry Count

The PEF startup delay disable feature is not supported.

The processing of alerts and non-alert actions after power loss is not supported (see sections 15.13.1 and 15.13.2 of the *IPMI 1.5 Specification*, Revision 1.1).

See the *IPMI 2.0 Specification* for additional details of PEF operation.

4.28.2 OEM Action

OEM actions are performed after the IPMI PEF actions and alerts.

Table 27. OEM PEF Action Priorities

Action	Priority	Additional Information
OEM - Shutdown	6	This action is performed only when powered on.

PEF can be configured to perform OEM actions using the OEM Event Filter Table, shown below. The table contains the same number of PEF entries as in the PEF table.

Table 28. OEM Event Filter Table

Byte	Field	Description
1	OEM Action	[7:1] reserved [0] 1b = Shutdown 0b = no Shutdown
2	Shutdown	Used to request a graceful shutdown operation using an OS agent. It is a no-op when the OS agent is not running, but it logs a SEL. The SEL log created by this OEM action will not perform further PEF OEM shutdown operation. [7:1] reserved [0] 1b = OS Shutdown and restart 0b = OS Shutdown and no restart

4.28.3 Dial Page Alerting

Dial Paging operates using an external modem connected to the system's onboard EMP serial connection on COM2. With Dial Paging, the system can be configured to automatically dial up a paging service when a platform event occurs. Dial Paging is an alert type supported by Platform Event Filtering (PEF). A Dial Page can be initiated by the arrival of an event that triggers the PEF Dial Page action or it can be initiated by an *Alert Immediate* command with appropriate parameters.

The following Dial Page resource sizes are platform-specific. See the server board BMC EPS for their values.

- Alert String Count
- Dial String Count (shared with Alert over Serial/PPP)
- Serial/Modem Alert Destination Count (shared with Alert over Serial/PPP)

See the *IPMI 1.5 Specification* for additional details on this feature.

4.28.4 Alert over LAN

Two types of Alerts are supported over LAN.

- Platform Event Trap (PET) alerts – Standard and Advanced
- SMTP Alerts – Advanced (Dedicated NIC only)

The Alert over LAN feature is used to notify remote system management application of PEF selected events regardless of the state of the host's operating system. LAN alerts may be sent over any of the LAN channels supported by a server board, modulo the specific channel capabilities. The BMC implements three OEM PEF parameters associated with PET Alerts over LAN. The parameter data formats are described in Table 63.

- PET OEM String (parameter 96)
This string, if defined, is included as part of the PET packet OEM data field.
- Infinite Retry Alert Behavior (parameter 97)
A byte value where if equal to 1, indicates that the Alert over LAN feature should retry Alerts until they succeed.
- UTC Offset (parameter 98)
This parameter provides a value for the UTC Offset field in the PET packet.

The following Alert over LAN resource sizes are server board-specific. See the server board BMC EPS for their values.

- LAN Alert Destination Count

See *IPMI 2.0 Specification* for additional details on PET Alerts feature. See section 5.4 for more details on SMTP Alerting.

4.28.5 Alert over Serial/PPP

Alert over Serial/PPP uses the same IP/UDP packet encapsulation as Alert over LAN, but allows alerts to be delivered via modem to a PPP enabled destination. An alert can trigger a dial out to one or more destinations as specified in the Alert Policy table, Serial/Modem Destination, and PPP Account parameters.

The following Alert over Serial/PPP resource sizes are server board-specific. See the server board BMC EPS for their values.

- Alert String Count
- Alert String Size
- Dial String Count (shared with Dial Page Alerting)
- Serial/Modem Alert Destination Count (shared with Dial Page Alerting)
- Serial/Modem Destination IP Address Count
- PPP Account Count

The Microsoft CBCP (Callback Control Protocol) is not supported. See the *IPMI 2.0 Specification* for additional details on this feature.

5. Advanced Edition Features

Advanced features are network-based features that require dedicated LAN Channels and potentially feature specific hardware support. This is accomplished through the use of the Generic Communication Module (GCM) as a dedicated LAN channel. The GCM has an integrated Intel® 82551QM network controller.

5.1 Feature Support Detection

Some or all of the Advanced features may be provided by a particular BMC. There are two ways to detect Advanced feature support.

- The *Get Platform Information* command provides a *Feature Support* parameter that indicates the overall BMC support for the advanced features, on a per-feature basis.
- Each LAN channel provides an *OEM Feature Support* parameter that indicates the advanced feature support for that channel.

5.2 Feature Configuration Model

The advanced features have up to three different types of configuration:

- Per-Feature: implemented as *Get/Set XX Configuration Parameter* OEM IPMI commands, where XX represents the feature of interest. Currently, only the HTTP Server implements this type of configuration.
- Per-Channel: implemented via feature specific *Get/Set XX Channel Configuration Parameter* OEM IPMI commands, where XX represents the feature of interest. The commands have a channel number selector to determine what channel the parameter applies to.
- Per-User: implemented via *Get/Set User Feature Configuration Parameter* OEM IPMI commands for Telnet and HTTP/HTTPS. KVM user access is controlled via IPMI 2.0 payload access commands.

5.3 HTTP Server

The HTTP server provides web pages allowing viewing of server board state (e.g., health, SEL) and secure platform power and reset control.

The HTTP server is only accessible on LAN channels that support the TCP protocol. This means that LAN channels that are shared with the system (have the same IP address) cannot support HTTP. Support for the feature for a particular LAN channel can be determined by the LAN Channel *OEM Feature Support* parameter.

The HTTP feature has server-global configuration parameters managed via the *Get/Set HTTP Server Configuration Parameter* commands.

The HTTP feature can be enabled or disabled via an HTTP configuration parameter using the *Set HTTP Channel Configuration Parameter* command. In addition, each user can be enabled or disabled for HTTP use via the *Set User Feature Configuration Parameter* command.

The HTTP module accepts requests via HTTP on port 80. By default, the web server will listen on port 80 for HTTP connections and port 443 for HTTPS connections. Some sites may change these values in order to hide access to the system. The commands to change these are defined in Section 7.3.

There are two styles of interface supported via the network interface:

- Browser interface
- Program interface

The “browser interface” style presents web pages to the user, who may click buttons or enter text values in order to interact with the server.

The “program interface” style accepts requests using a more traditional HTTP GET/POST mechanism for presenting requests. Responses are returned wrapped in XML and are designed for ease of parsing by software – not for necessarily for ease of display.

5.3.1 Browser Interaction

This style of network interaction presents a web browser style of interaction. Users may click buttons, fill out web forms, and move from page to page interacting with the BMC.

The webserver content consists of an outer frame, with minimal information and navigation buttons, and a set of inner pages. In addition there are a set of built-in pages largely provided for development which are independently accessible, and which display as a full screen page.

The set of inner pages include the following:

Table 29. Initial Pages

Page Name	Summary Description
System Summary	Displays static information as well as selected sensor values
System Event Log	Provides a high level view of the Event Log entries
SEL Details	Provides raw data and extended Event Log display
Power	Display power state, and permit power control of the system.
IPMI Command	Execute CLI commands or IPMI hex strings and display the result.
Configuration	A page that allows configuration of persistent storage values that are used by dynamic web pages.
Help	Shows help information for the system. May be configured by a vendor
Vendor	Shows vendor-specific information The default value is www.intel.com. May be configured by a vendor.

5.3.1.1 Outer Frame

The outer frame includes the following:

- The button that invokes the vendor website.
- The button that invokes the help screen.
- The button that invokes the about box.

Note: Although this may be rewritten by vendors, copyright notices are contained in the code. These notices must be retained.

- System identification: host name, domain name, IP address, and firmware version. This information is not dynamically updated, i.e. the information is really correct at the time the outer page is loaded.
- Buttons to dispatch web pages: System Summary, System Event Log, SEL Details, Power Control, IPMI Commands, and Configuration.
- Graphic images to fill in the unused areas.

5.3.1.2 System Summary Page

The system summary page displays sensor information about the processors, memory, attached devices, and static system information from the FRU/SDR package. The sensor information is updated every 60 seconds by re-reading the sensor values from the webserver. The FRU/SDR data is only read when the page is loaded, since it is considered unchanging. Should changes be made (e.g. by reloading information via the FWPIAUPD program or by running the System Configuration Wizard) the browser page must be killed and a new one started to view the changed data. This is because the browser caches this information.

This page is the default page – that is it is loaded when the web pages are started. In version 1 of the webserver, this page is also reloaded when the page is refreshed via browser button or the <F5> key.

As shown in the following figure, sensor data should be as descriptive as possible. Where bit-levels are involved, the English definition of the bits are displayed. Where sensor readings are involved, units are displayed if practical, e.g. RPM. Sensor readings that are outside of alarm limits are displayed in bold red; outside of warning but inside of alarm limits are displayed in bold yellow.

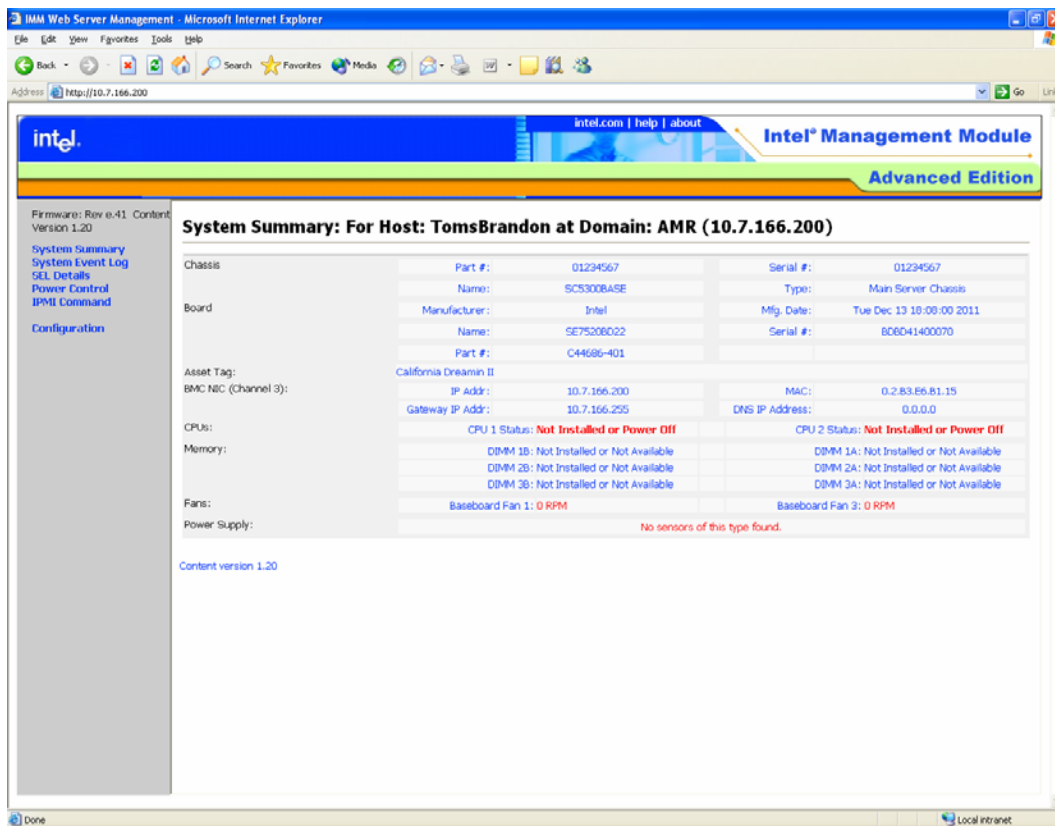


Figure 9. System Summary Web Page

5.3.1.3 System Event Log Page

The SEL page presents the user with the total number of SEL entries and displays selected records. By default it displays the first ten records, but that can be changed by the user. Buttons are provided to allow the user to request the next group of records, the previous group of records, the first group of records, or the last group of records. All of these groups default to ten records, but the next and previous groups will display the number of records selected by the user.

Edit boxes are provided to allow the user to set the starting and ending record numbers. This is the method to change the number of displayed records in the SEL view. A button is provided to request the records selected by this method.

A button is provided to allow the user to clear the all of the records from the System Event List. When this occurs, the display will be updated to show the remaining records.

The display includes the Record number, the date and time (if available), and a description of the event in English.

The page updates the total number of records every 30 seconds. When the number of records is changed (for example by clearing the SEL via an IPMI command) the display is updated. This update is suppressed if the user has set a record range that is still available (specifically the starting record number is greater than 1 and less than the last record number available) it is assumed that the user wishes to keep the same records displayed and the display is not updated.

Similarly, if the user is in the process of entering the start or end values the update is suppressed until the user has selected a button to display the records selected.

The records are always displayed in order. If the user requests a starting record number that is less than the ending record the two are interchanged. The time to display records depends on the number of records selected. About one record per second is processed, so 120 records may take up to two minutes to display.

The web server will display a message if the SEL is empty.

In addition to the number of records, the display includes the remaining number of records and the percentage of the total records used. This is displayed to the nearest percent; there may be times when records are in the event log but the percent indicated is zero.

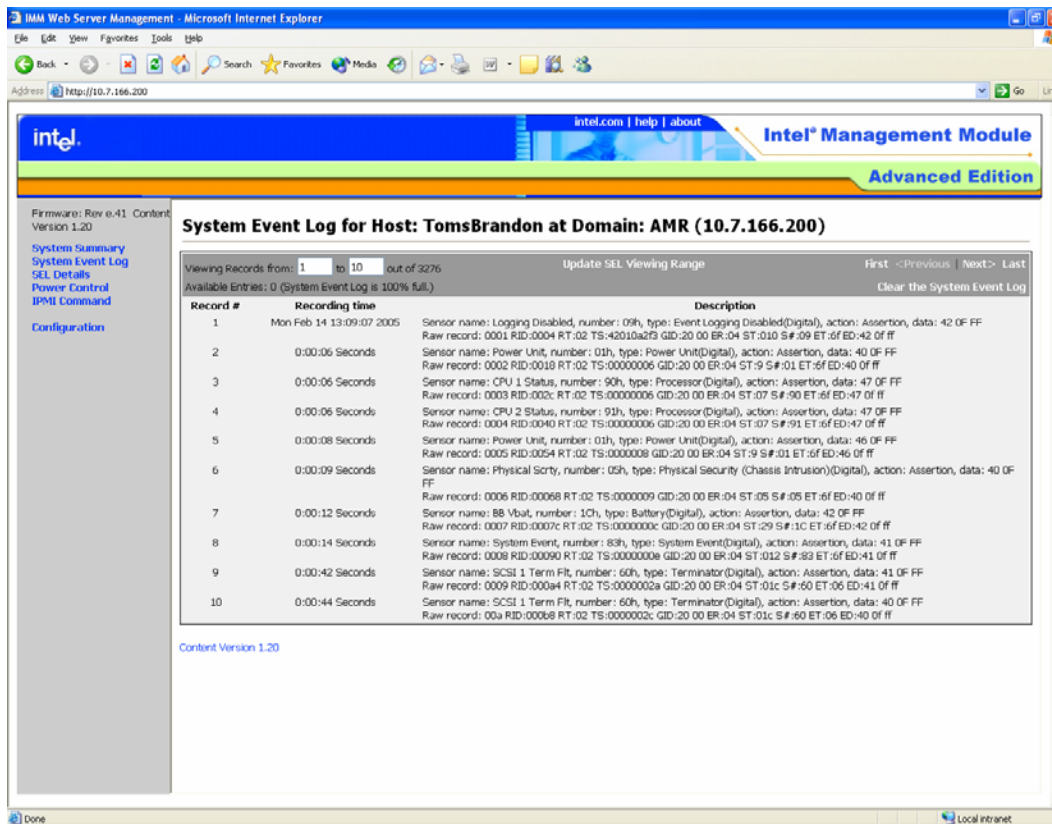


Figure 10. System Event Log Web Page

5.3.1.4 Power Control Page

The Power page allows the user to view the current power state of the host, to control the power, and to reboot the system.

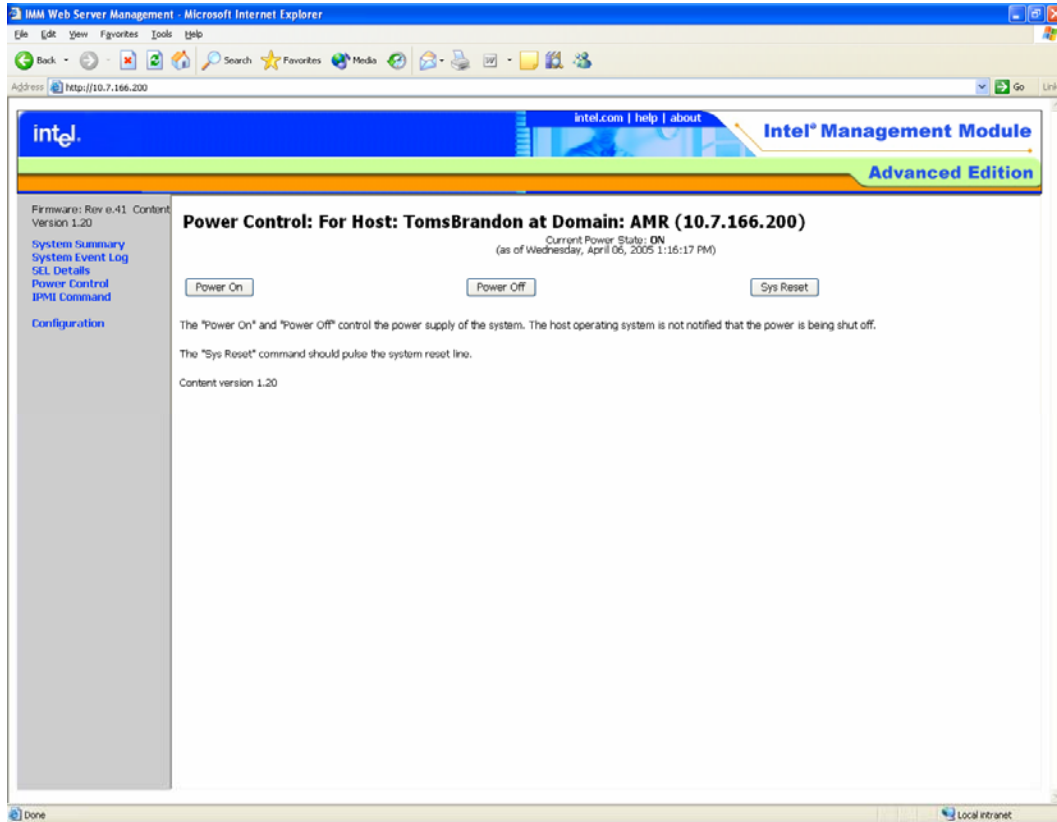


Figure 11. Power Control Web Page

5.3.1.5 IPMI Command Page

The IPMI Command page supports IPMI commands represented as hex strings. The result of the command will be displayed at the bottom of the page. See Figure 12. The command is issued when the execute button is clicked or when the return key is pressed.

The user must enter the IPMI commands in their full hex values. '20.18.01' is acceptable, but '20.18.1' will result in an error.

Error codes are detailed in Table 88.

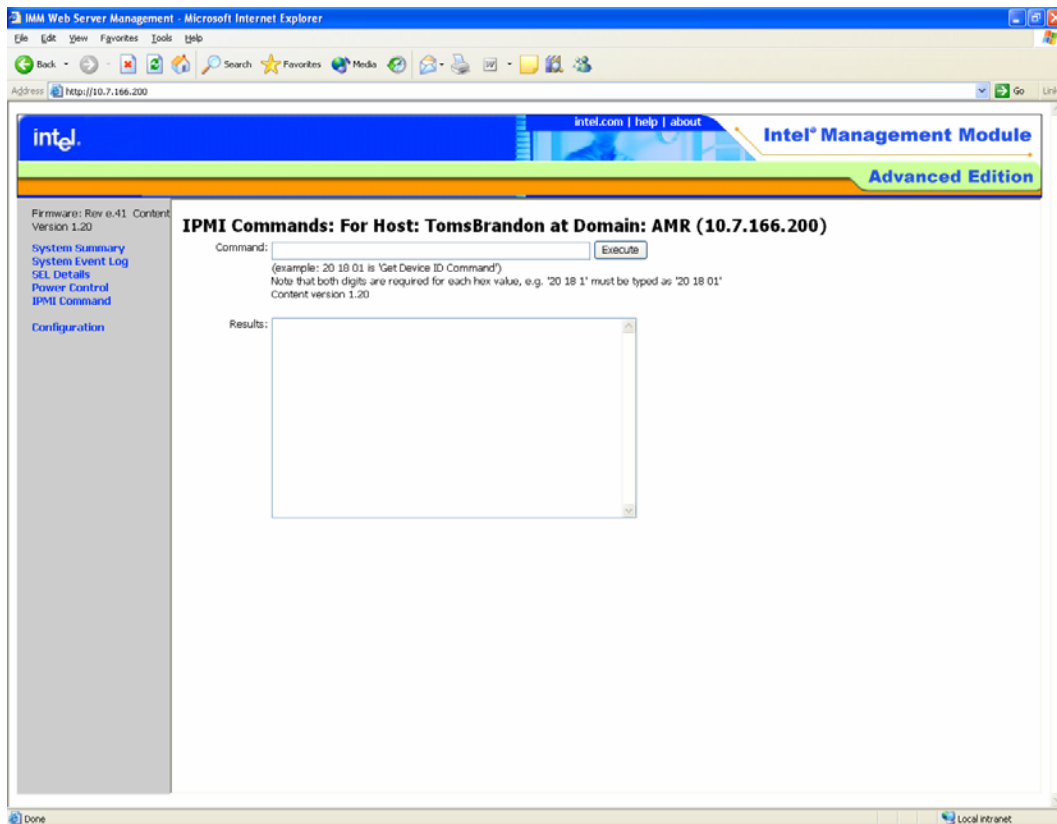


Figure 12. IPMI Commands Web Page

If the user enters a legal command but that command has an incorrect context (e.g. requesting information from a non-existent channel) an appropriate IPMI completion code is returned and displayed. See Figure 13.

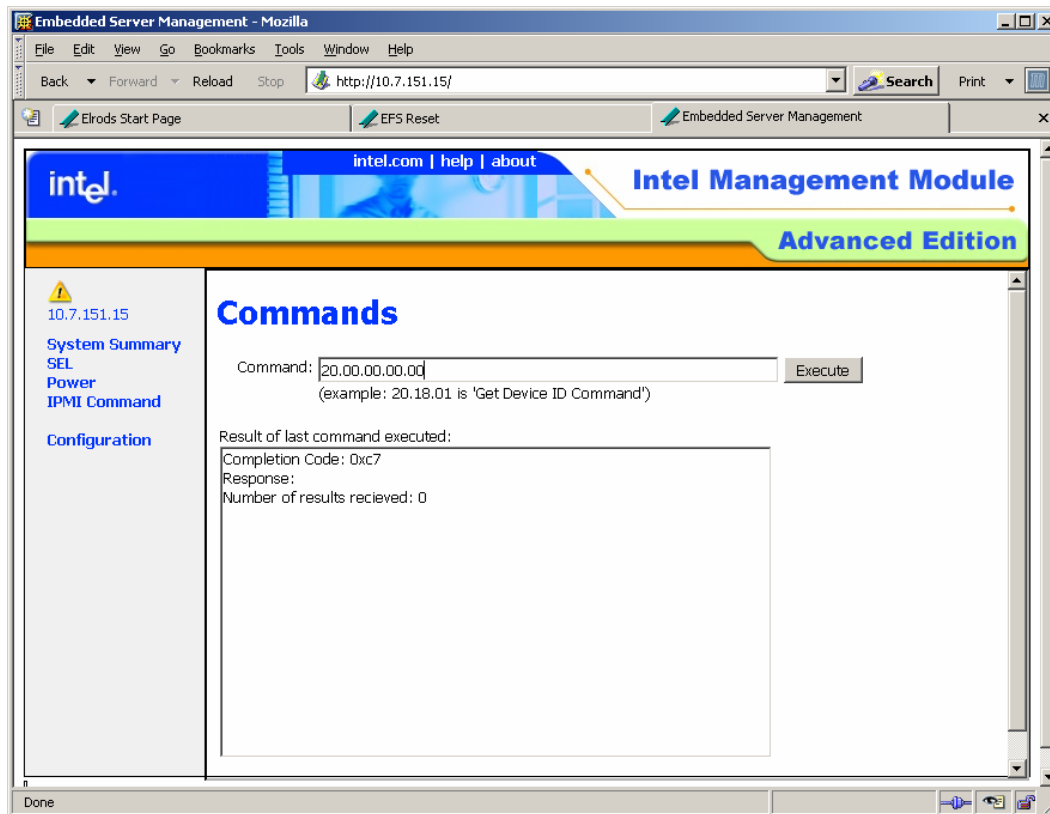


Figure 13. IPMI Commands Web Page With IPMI Error

If the user enters an illegal command, an alert box is displayed. See Figure 14. In this case the return code and data are invalid and should be ignored.

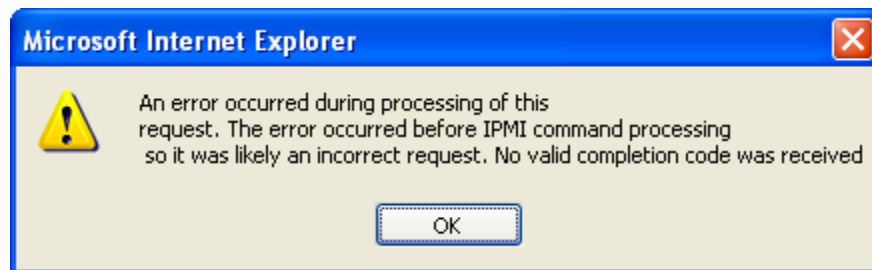


Figure 14. IPMI Commands Web Page with Invalid Command

5.3.1.6 Embedded Web Server Configuration

The Configuration page allows users to set most of the web configuration values. See Figure 15 and Figure 16.

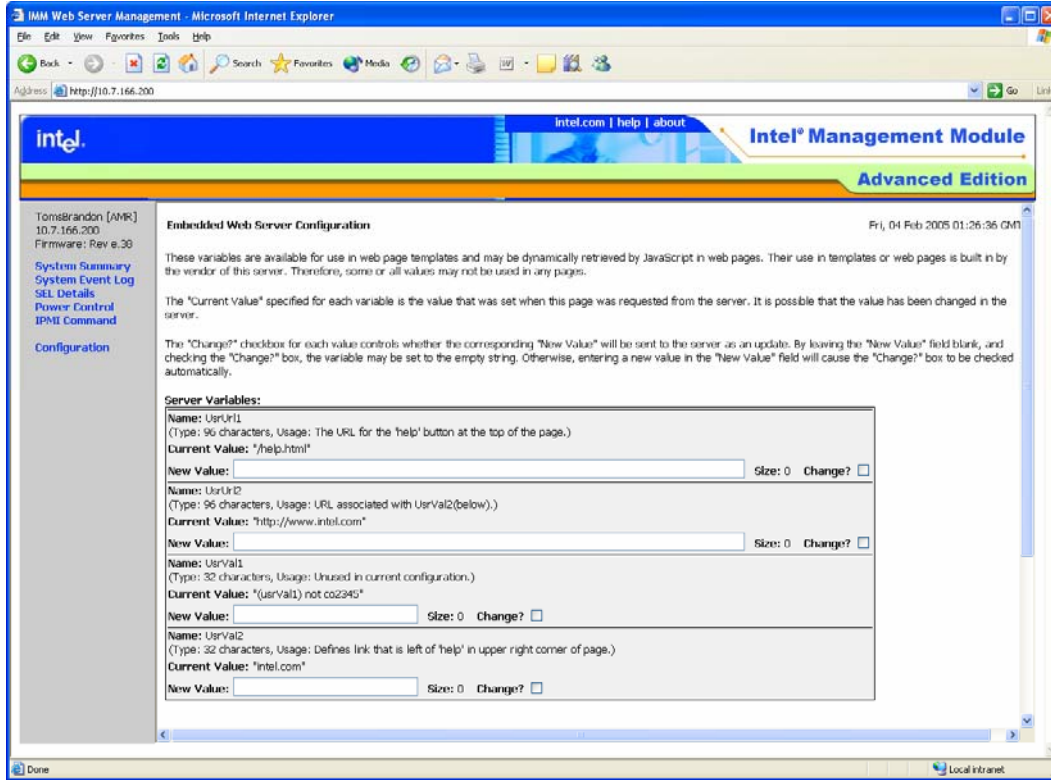


Figure 15. Web Server Configuration - Top Part

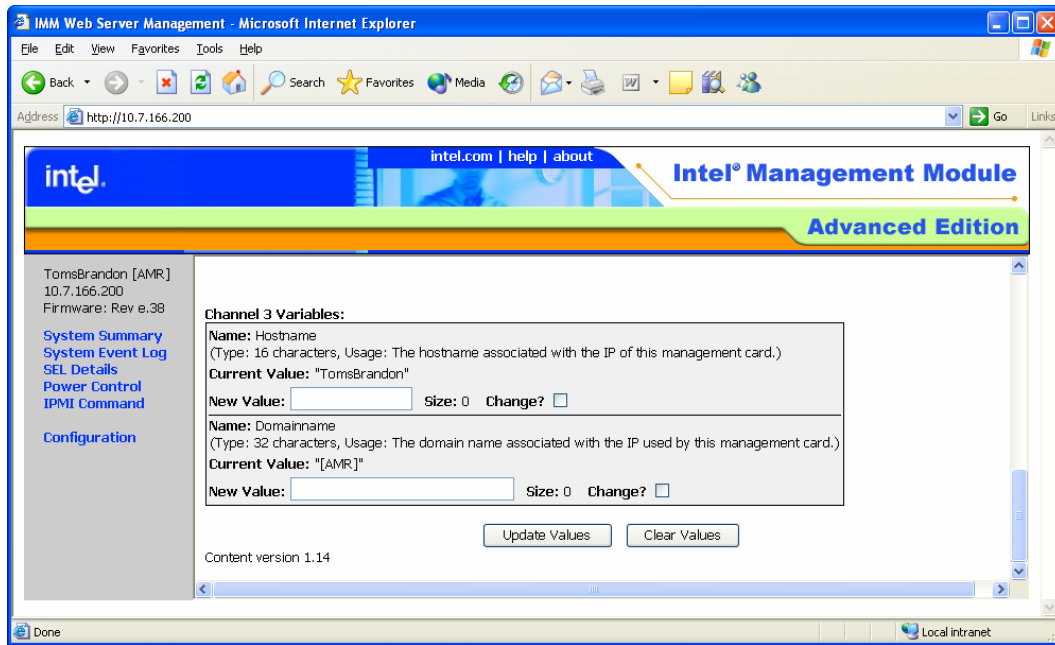


Figure 16. Web Server Configuration - Lower Part

The uses of the various fields are as follows:

- UsrUrl1: The URL which is accessed when the **help** button on the outer frame is clicked.
- UsrUrl2: The URL which is accessed when the vendor website button on the outer frame is clicked. The default label for this button is **intel.com**, but it may be changed (See UsrVal2, below).
- UsrVal1: Not used in the default web pages.
- UsrVal2: Label for the vendor website button.
- Hostname: For information only. This shows the host name associated with the server on which the website is located.
- Domain Name: For information only. This shows the domain name associated with the server on which the website is located.

The URLs above are processed by the webserver before being returned to the browser, so they must be in the following format:

- Internal (contained in EFS): /<page name>, e.g. **/help**
- External (outside of the webserver machine): http://<full URL>, e.g. **http://www.intel.com**

Note: When the URLs are changed, other instances of the browser are not updated. To reflect changes that are displayed (such as the hostname) the user must reload the page.

5.3.1.7 Security

As requests for information, web pages or to execute IPMI commands are received, the web server verifies that the appropriate authentication has been carried out.

5.3.1.7.1 User Security

User security will permit or deny access to the system based on the authentication of the user. The following access levels are used:

Table 30. User Access Levels

User	Description
Administrator	Administrator level users are allowed. No restriction on allowed commands
Operator	Operator and administrator level users are allowed. Commands that change BMC configuration are restricted
User	User, operator and administrator level users are allowed. Commands are restricted to read-only access of system information.

Each action such as page retrieval or command execution has a privilege level associated with it. When the user attempts to access a page or execute a command, the web server will verify that the appropriate level of authentication has been carried out. When a user first attempts to utilize a user resource, they will be prompted for a user name and password.

The user name and password are settings that need to be configured for each system. All commands will be carried out using privileges of the username and password provided by the user. If the user enters a “User” level username and password and attempts an “Operator” level action (e.g. power down), then they will be prompted again for a username and password.

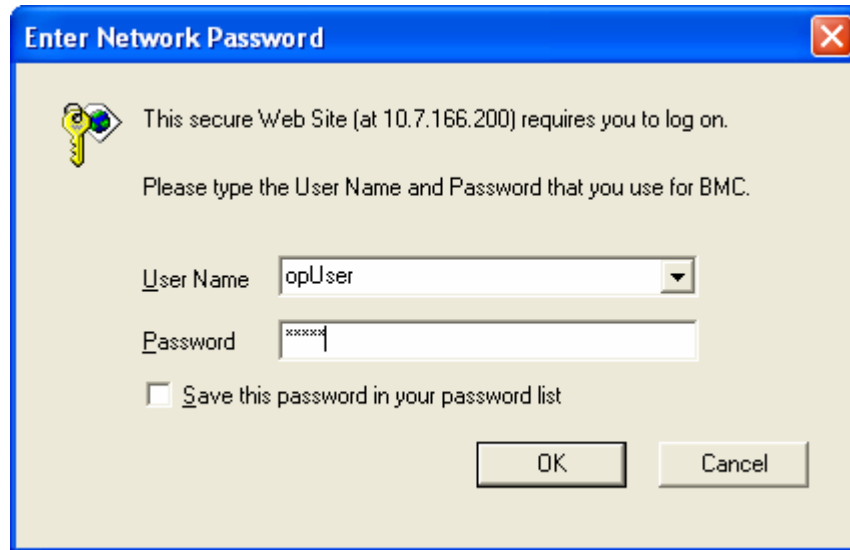


Figure 17. Authentication Prompt

If the user enters an administrative username and password, then all actions will be carried out as administrator and they wouldn't be prompted for a password again.

Customers have varying security requirements. A small company may have a server sitting on the internet and would thus want maximum security. A company that performs large computations may have a server room with a dedicated management network that has no outside access and thus requires no per-system security.

When a user first requests the web page, they have entered no username or password. The web server will attempt to validate this access using the NULL login and a NULL password. If the NULL user is permitted to log on with a NULL password, then "anonymous" access is allowed. If the NULL user is permitted to log in with administrative access, there will never be any name/password dialog presented. This is intentional, since it conforms to one user model which must be supported.

The presentation of the name and password dialog is under the control of the browser, rather than being under programmatic control. Internet Explorer* 6.0 will by default cache user and password dialog contents. When the name and password are cached, a dialog may be displayed with the user name and password filled in or no dialog may be displayed. This can be turned off but some amount of "mini-caching" occurs in a given browser instance. If, for example, a connection has been made with a given account and password, and the validity of that account and password are changed, the connection will be continued with the original account/password, and will fail due to authentication errors.

Mozilla* functions slightly differently. If an account and password is accepted on one instance it is maintained on other instances as well.

5.3.1.7.2 Channel Security

In addition to the user level security, security is provided on a per channel basis.

Each IP address available to the web server has two port settings, an HTTP port and an HTTPS port. The HTTP port is the port number on that IP address which the web server listens to for incoming connections. Traditionally, the HTTP port is port "80". If this value is set to "0", then the port is disabled. The HTTPS port is the port the web server listens to for SSL connections. Traditionally, the HTTPS port is port "443". Even on platforms that support SSL, the impact on the system is significant in terms of both memory and processing power.

5.3.2 Program Interaction

Two programmatic interfaces will be supported for scripting and external software access to the system. Both interfaces submit commands as form arguments (either GET or POST format). One format returns an XML message with the response data, the second format returns an HTML web page. The second format is more useful in web page design, while the first format may be easier for scripts and non-browser programs to parse.

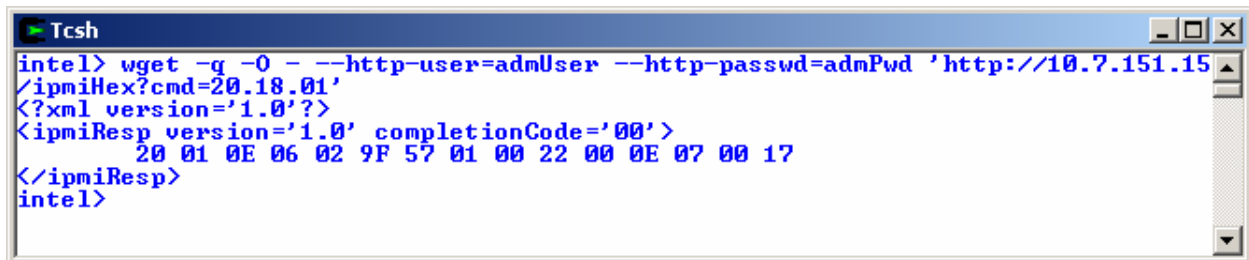
5.3.2.1 Simple HTTP Request

Scripts and non-browser programs which wish to make IPMI requests will find it easiest to use the “simple request” format. This format allows the request to be submitted as part of the URL. The format is:

```
http://hostname/ipmiHex?cmd=XX.XX.XX...
```

This form of IPMI request is easy to use from web access software built into Perl*, Java*, Python*, or similar tools, or in programs such as wget (on Unix or Windows* / Cygwin*).

If a username and password are required and no acceptable username/password combination was provided, then an HTTP authorization failure will be returned. This will typically cause the browser to prompt the user for a username and password. Command-line tools must be able to perform digest authentication (as wget does) in order to properly communicate the authentication data to the server.



```

Tcsh
intel> wget -q -O - --http-user=adminUser --http-passwd=adminPwd 'http://10.7.151.15
/ipmiHex?cmd=20.18.01'
<?xml version='1.0'?>
<ipmiResp version='1.0' completionCode='00'>
      20 01 0E 06 02 9F 57 01 00 22 00 0E 07 00 17
</ipmiResp>
intel>

```

Figure 18. Example 'wget' Command-line IPMI Request

5.3.2.2 Simple HTTP Response

The response to simple requests is an XML response. The format of the response is:

```
<ipmiResp completionCode="XX">XX-XX-XX...</ipmiResp>
```

The inner text of the element may be empty, but the completion code will always be provided.

If insufficient authentication was provided, the completionCode will be set to “D4” as specified by the IPMI specification.

5.3.2.3 HTML Page Request

Web page designers may prefer to receive the response data as a web page which calls an “onload()” JavaScript handler when it loads.

The HTML Page Request command mechanism implements a hidden frame (or hidden iframe) on the web page which can be loaded by the main page in response to events such as a timer or user interaction. The Web server will load a web page into the hidden frame, and have that web page call a JavaScript function in the main page from its “onload” event handler (in the <body> element). An example of this request has the format:

```
http://hostname/ipmiPage?cmd=XX.XX.XX...&onload=functionName
```

where *hostname* is the hostname of the system, and *functionName* is the name of a JavaScript function in the parent.

The arguments to the *functionName* function specify the return information as defined in the following table.

Table 31. "Onload" Function Arguments

Web Page Requested	Response Values
.../ipmiPage	Argument 1: Completion code Argument 2: The command that was executed Argument 3: The command response
.../webCmdPage	The requested data is returned. Argument 1: An error code 0 or 1 indicate no error. Argument 2: An error string associated with the code Argument 3: The response data (if any) See Table 32 for more information about web commands.

Using this technique, dynamic HTML pages can change themselves without reloading, providing a more interactive experience. This mechanism can also be used to get updates of sensor values such as system health or to monitor values such as temperatures.

If the command is “webCmdPage”, then the following subcommands may be entered. A request has the format:

`http://hostname/webCmdPage?cmd=XX.XX.XX...&onload=functionName`

Table 32. webCmdPage Sub-commands

webCmdPage Sub-Command	Action and Response Values																												
getSEL	<p>Returns the requested SEL data.</p> <p>This command requires the following additional parameters, which are provided as either GET or POST format form fields.</p> <table border="1" data-bbox="383 579 1360 751"> <thead> <tr> <th data-bbox="383 579 553 611">name</th> <th data-bbox="560 579 1360 611">use</th> </tr> </thead> <tbody> <tr> <td data-bbox="383 619 553 651">first</td> <td data-bbox="560 619 1360 651">The number (1-based) of the first SEL entry to be retrieved.</td> </tr> <tr> <td data-bbox="383 659 553 690">last</td> <td data-bbox="560 659 1360 690">The number (1-based) of the last SEL entry to be retrieved.</td> </tr> <tr> <td data-bbox="383 699 553 730">onload</td> <td data-bbox="560 699 1360 751">The name of the “parent window” function to be called from the JavaScript onload function.</td> </tr> </tbody> </table> <p>The JavaScript returned will call the parent window from its onload handler. The parent window's function will be passed the following arguments: Argument 1: Completion code Argument 2: The command that was executed Argument 3: An array of SEL objects. Each object has computer friendly fields and human readable fields which are easy to display.</p>	name	use	first	The number (1-based) of the first SEL entry to be retrieved.	last	The number (1-based) of the last SEL entry to be retrieved.	onload	The name of the “parent window” function to be called from the JavaScript onload function.																				
name	use																												
first	The number (1-based) of the first SEL entry to be retrieved.																												
last	The number (1-based) of the last SEL entry to be retrieved.																												
onload	The name of the “parent window” function to be called from the JavaScript onload function.																												
clearSEL	<p>Will clear the SEL.</p> <p>The JavaScript returned will call the parent window from its onload handler. The parent window's function will be passed the following arguments: Argument 1: completion code Argument 2: the command that was executed</p>																												
changeVars	<p>Will set the web server variables specified as arguments.</p> <p>Possible values are:</p> <table border="1" data-bbox="492 1209 1321 1728"> <thead> <tr> <th data-bbox="492 1209 756 1241">Name</th> <th data-bbox="763 1209 1321 1241">Use</th> </tr> </thead> <tbody> <tr> <td data-bbox="492 1249 756 1276">HealthyUpdateRate</td> <td data-bbox="763 1249 1321 1276">Changes the healthy system update rate.</td> </tr> <tr> <td data-bbox="492 1285 756 1312">WarningUpdateRate</td> <td data-bbox="763 1285 1321 1312">Changes the warning level system update rate.</td> </tr> <tr> <td data-bbox="492 1320 756 1348">CriticalUpdateRate</td> <td data-bbox="763 1320 1321 1348">Changes the critical level system update rate.</td> </tr> <tr> <td data-bbox="492 1356 756 1383">EfsUpdatePath</td> <td data-bbox="763 1356 1321 1383">Changes the EFS Update URL path component.</td> </tr> <tr> <td data-bbox="492 1392 756 1419">EFSInfoPath</td> <td data-bbox="763 1392 1321 1419">Changes the EFS Info URL path component.</td> </tr> <tr> <td data-bbox="492 1428 756 1455">UsrUrl1</td> <td data-bbox="763 1428 1321 1455">Sets the 'UsrUrl1' value.</td> </tr> <tr> <td data-bbox="492 1463 756 1491">UsrUrl2</td> <td data-bbox="763 1463 1321 1491">Sets the 'UsrUrl2' value.</td> </tr> <tr> <td data-bbox="492 1499 756 1526">UsrVal1</td> <td data-bbox="763 1499 1321 1526">Sets the 'UsrVal1' value.</td> </tr> <tr> <td data-bbox="492 1535 756 1562">UsrVal2</td> <td data-bbox="763 1535 1321 1562">Sets the 'UsrVal2' value.</td> </tr> <tr> <td data-bbox="492 1570 756 1598">3:Hostname</td> <td data-bbox="763 1570 1321 1598">Sets the hostname for IPMI channel 3.</td> </tr> <tr> <td data-bbox="492 1606 756 1633">3:Domainname</td> <td data-bbox="763 1606 1321 1633">Sets the domain name for IPMI channel 3.</td> </tr> <tr> <td data-bbox="492 1642 756 1669">3:TelnetUrl</td> <td data-bbox="763 1642 1321 1669">Sets the telnet URL for IPMI channel 3.</td> </tr> <tr> <td data-bbox="492 1677 756 1705">3:KvmUrl</td> <td data-bbox="763 1677 1321 1705">Sets the KVM URL for IPMI channel 3.</td> </tr> </tbody> </table> <p>You should only send those values which should be changed. For example, to change the CriticalUpdateRate to 600 you might use the following: <code>http://.../webCmdPage?cmd=changeVars&CriticalUpdateRate=600</code></p>	Name	Use	HealthyUpdateRate	Changes the healthy system update rate.	WarningUpdateRate	Changes the warning level system update rate.	CriticalUpdateRate	Changes the critical level system update rate.	EfsUpdatePath	Changes the EFS Update URL path component.	EFSInfoPath	Changes the EFS Info URL path component.	UsrUrl1	Sets the 'UsrUrl1' value.	UsrUrl2	Sets the 'UsrUrl2' value.	UsrVal1	Sets the 'UsrVal1' value.	UsrVal2	Sets the 'UsrVal2' value.	3:Hostname	Sets the hostname for IPMI channel 3.	3:Domainname	Sets the domain name for IPMI channel 3.	3:TelnetUrl	Sets the telnet URL for IPMI channel 3.	3:KvmUrl	Sets the KVM URL for IPMI channel 3.
Name	Use																												
HealthyUpdateRate	Changes the healthy system update rate.																												
WarningUpdateRate	Changes the warning level system update rate.																												
CriticalUpdateRate	Changes the critical level system update rate.																												
EfsUpdatePath	Changes the EFS Update URL path component.																												
EFSInfoPath	Changes the EFS Info URL path component.																												
UsrUrl1	Sets the 'UsrUrl1' value.																												
UsrUrl2	Sets the 'UsrUrl2' value.																												
UsrVal1	Sets the 'UsrVal1' value.																												
UsrVal2	Sets the 'UsrVal2' value.																												
3:Hostname	Sets the hostname for IPMI channel 3.																												
3:Domainname	Sets the domain name for IPMI channel 3.																												
3:TelnetUrl	Sets the telnet URL for IPMI channel 3.																												
3:KvmUrl	Sets the KVM URL for IPMI channel 3.																												

5.3.2.4 Security

Both simple and page request mechanisms may require that a username and password be sent to the server in order to authenticate the user and carry out the request.

The both mechanisms, using HTTP as its transport protocol, will pass the username in clear text, but use MD5 in order to hide the password using Digest Authentication.

Each IPMI command that is executed has a minimum privilege level that is required as defined by the IPMI specification. The web server enforces this requirement on a per-command basis. If the user does not have sufficient privilege to execute the command, the web server will send an Authentication Failure HTTP message back. Most browsers will prompt the user for username/password information and then reattempt to access the page.

In addition, each web page is stored in the EFS has a minimum user level associated with it. This mechanism is a bit redundant, since each command must be authenticated, but it aids the designer in presenting a smoother user experience. If the page designer knows that operator privilege is required to perform any action on the page, then the page can be protected and the user must authenticate before seeing any data.

5.3.3 Auto Refresh

The following fields will be refreshed automatically:

- Summary Page: All sensor readings will be updated every 60 seconds. The displayed values will be updated after each reading
- Power Control Page: The power status will be updated every 30 seconds, and the displayed status updated after each reading

SEL and SEL Details Pages: The total record count will be updated every 60 seconds. If there is a change in the record count affecting the number of displayed records, the records will be re-read and the display will be updated. The number of records displayed will be reduced if a smaller number of records exist, however every attempt will be made to maintain the users number of records chosen. For example, if ten records are displayed, and the SEL is cleared, only one record will remain, and that one record will be displayed.

5.3.4 Authentication and Encryption

The authentication of users and the security of messages sent to and from the web server is a growing concern in the management of all systems. If the IMM module's dedicated NIC is on a private "management" network, then authentication and encryption may not be needed and should be avoided to maximize performance.

5.3.4.1 Authentication

HTTP's Digest authentication provides protection of the password. Per RFC 2617, Digest Authentication is used to protect the user's password. The 128 bit MD5 digest that is used by default, creates a hash using a server supplied nonce and the password the user types in.

The Digest scheme works as follows:

1. The browser sends a request for a page.
2. The server replies with an “unauthorized” 401 error, and specifies MD5 digest as the preferred authentication scheme. It also provides a realm name, along with a nonce and other values.
3. The client creates a hash from the user’s login, the realm, the password and the nonce.
Note: This hash may include additional data, such as the URL of the requested page.
4. The client sends this 128-bit hash to the server in the header of the request, along with the nonce and user’s unencrypted login.
5. The server verifies that the nonce received from the client is one previously sent and that it is still valid.
6. The server constructs a hash using the user’s login, the realm, the locally stored password and the nonce.
7. If the two hashes match, then access to the resource is permitted.

If the nonce is old or bad, then the request will be rejected and the browser must reattempt the connection.

The server may calculate a new nonce and provide it in the header of its response. This can permit “one time use” of a nonce, with minimal overhead in the exchange between client and server.

Note: Microsoft’s Internet Explorer and Mozilla* have interpreted the Digest RFC differently. This web server supports Digest Authentication on both Mozilla and Internet Explorer.

5.3.4.2 Encryption

HTTPS connections may be enabled in the web server. SSL encryption protects the username, password and all data in a transaction.

5.3.5 Customization and Internationalization

Customizing embedded web server pages is possible, but the process is not documented here. Contact your field representative for information on customizing the Intel® Management Module – Advanced Edition web pages.

5.4 SMTP Alerting

The BMC Email Alerting feature provides a way for IPMI Alerts to be delivered via Email as well as the IPMI standard alerting methods. This allows Email Alerting in an operating system independent and operating system absent (e.g., pre-OS and OS-hung) fashion.

The BMC provides support for sending mail via SMTP, the Simple Mail Transport Protocol as defined in Internet RC 821.

SMTP alerting is implemented as an OEM alert type (OEM1) for LAN channels. This alert type is Unacknowledged only. Each LAN Alert Destination can be configured as an SMTP alert. A separate LAN Channel OEM parameter defines the specific SMTP alert configuration associated with a specific LAN Alert Destination. SMTP alerting is only available over the GCM interface.

Email Alerting does not support alert acknowledgement. Once the mail is accepted by the SMTP server, the alert is assumed to be delivered.

The BMC supports a set of email alerting configurations independent of the channels. A pair of *OEM Get/Set* configuration commands are implemented to manage this configuration data. Each email configuration includes the following:

- Sender Machine Name
- Email From Address
- Email To address
- Email Subject

The LAN and Serial channel Alert Destination configurations provide the destination IP address and gateway selector for the SMTP server and reference a specific email alert configuration, via a LAN or Serial channel OEM configuration parameter. Email alerts are always sent to the standard SMTP TCP port (25).

The email message text consists of any Alert String associated with the Alert by the Alert Policy and a notice that this is an automatically generated message and the receiver should not reply. If the alert string does not exist, the text “Alert string unspecified, check server status” is used.

SMTP alert configurations are accessed/defined via the *Get/Set SMTP Alert Configuration Parameter* commands. The supported parameters include:

- Number of supported alert configurations
- Email *From*: name (per configuration)
- Email *To*: name (per configuration)
- Email *Subject*: line (per configuration)
- Alerting machine name (identifies the managed server to the SMTP server, it is used with the SMTP *HELO* command)

Support for the feature for a particular LAN channel can be determined by the LAN Channel *OEM Feature Support* parameter.

The Message Content is a human readable version of the SEL event that triggered the alert.

5.4.1 Email Alerting Configuration Parameters

A pair of email Alerting Configuration commands provide email alerting configuration parameter access. The following string types are provided.

- **From Address:** To carry out any email transactions, a valid "from address" must be supplied to the SMTP server. This address must be supplied even though the server is not capable of receiving emails. The SMTP server will check the validity of this address and can reject mail with an invalid address. This string is limited to 64 bytes.
- **To Address:** The "to address" must be a valid recipient's email address. This is the email address to which alert emails will be sent. This string is limited to 64 bytes.
- **Subject:** This string indicates the message is an alert. This string is limited to 64 bytes.
- **Sender Machine Name:** This string identifies the managed server to the SMTP server; it is used with the SMTP *HELO* command. This string is limited to 64 bytes.

5.4.2 Initialization and Setup

When the BMC is reset, the strings are read from persistent storage. If any data is not present or is corrupted, the following defaults will be used:

- From Address: No default
- To Address, No default
- Subject: "Intel Server Alert"
- Sender Machine: "BMC"

If the From or To Addresses are not set, no email is sent. No errors are logged to the SEL or the BMC self test data.

5.5 Telnet

The Telnet server provides text-based access to the BMC Native CLI (NCLI) command interpreter. See section 5.5.2 for the commands provided by native CLI.

User/password protection is provided, but the password is sent as clear text. Clients should use IPMI messaging if a stronger authentication mechanism is required and IPMI messaging over RMCP+ (IPMI 2.0) if data encryption is required (e.g., for setting user passwords remotely).

The Telnet server is only accessible on LAN channels that support the TCP protocol. This means that LAN channels that are shared with the system (have the same IP address) cannot support Telnet. Support for the feature for a particular LAN channel can be determined by software using the LAN Channel *OEM Feature Support* parameter.

The Telnet feature can be enabled or disabled independently for each supported LAN channel via a Telnet configuration parameter using the *Set Telnet Configuration Parameter* command. In addition, each user can be enabled or disabled for Telnet use on a per-channel basis via the *Set User Feature Configuration Parameter* command.

The Telnet server accepts connections on TCP port 23 (default) on all Telnet enabled channels. The maximum number of simultaneous Telnet sessions is four (4). See section **Error! Reference source not found.** for details on supported commands and parameters.

5.5.1 Common CLI

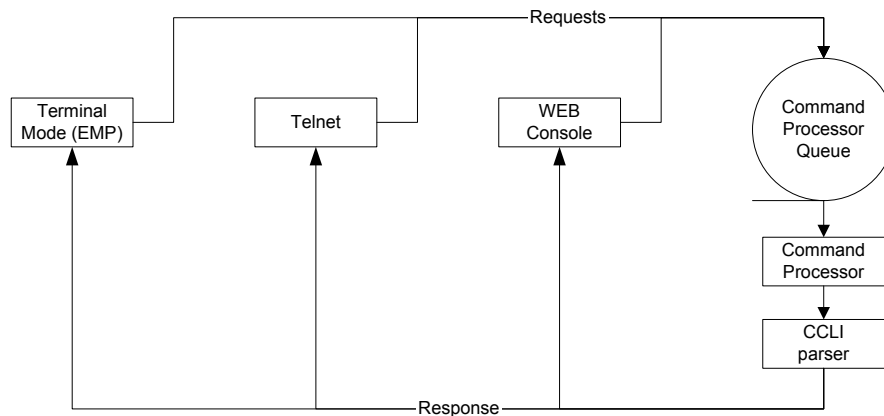
The Common CLI provides a common syntax for command line interface commands across the channels and platforms to communicate with the BMC through scripting.

The Intel® Management Module BMC can be configured using an Intel OEM channel configuration command to select the CLI syntax between Common CLI (CCLI) and Terminal Mode CLI (TMCLI) for channels supporting both syntaxes. Both CCLI and TMCLI command modes are mutually exclusive. When the CCLI is enabled, TMCLI commands are disabled automatically.

CCLI is a command parser/processor module, which can be used by different software modules including Terminal Mode console, Telnet console and WEB (HTTP) console.

This Common CLI performs the following operations:

- Receives text commands from the Terminal Mode, Telnet and WEB console.
- Checks the syntax for incoming text commands.
- Process the commands per the supported CCLI specification.
- Send response for the commands back to the Terminal Mode, Telnet or WEB console.



5.5.2 Native CLI Command Interface

The following table details the supported Native CLI command set. In the table, each command may provide the following output in addition to the output shown in the “Output” column:

- OK
- Unsupported command
- Unrecognized command
- Command time-out
- Insufficient privilege
- Session in progress

Table 33.Native CLI Command Set

Cmd Set	Cmd	Privilege Level	Option(s)	Short Description	Output
Built-in	help	Callback	[no switches]	Display list of valid commands.	IPMI Rev: x.x FW Rev: xx.xx Commands ..
			[-C <command>]	Returns a brief description of command	Command: xx

Cmd Set	Cmd	Privilege Level	Option(s)	Short Description	Output
Common	health	User	[no switches]	Display the health of the target.	<p>PWR:zzz H:xx T:xx V:xx PS:xx C:xx D:xx S:xx O:xx</p> <p>Where:</p> <ul style="list-style-type: none"> PWR = system power state H = overall Health T = Temperature V = Voltage PS = power supply subsystem F = cooling subsystem (Fans) D = Hard Drive / RAID Subsystem S = physical Security O = Other (OEM) <p>zzz is: "ON", "OFF" (soft-off or mechanical off), "SLP" (sleep - used when can't distinguish sleep level), "S4", "S3", "S2", "S1", "??" (unknown)</p> <p>and xx is: ok, nc, cr, nr, uf, or ??</p> <p>Where:</p> <ul style="list-style-type: none"> "ok" = OK (monitored parameters within normal operating ranges) "nc" = non-critical ('warning': hardware outside normal operating range) "cr" = critical ('fatal': hardware exceeding specified ratings) "nr" = non-recoverable ('potential damage': system hardware in jeopardy or damaged) "uf" = unspecified fault (fault detected, but severity unspecified) "??" = status not available/unknown (typically because system power is OFF)

Cmd Set	Cmd	Privilege Level	Option(s)	Short Description	Output
			-f	- display "full" health status (default is "short" summarized health status)	Health :xx Temperature :xx Voltage :xx ...
Computer System	ID	Callback	[no switches]	Display the unique identifier for this target (UUID)	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
	boot	Operator	[no switches]	Initiate a system reset : Attempts to reboot the system using OS shutdown by default	
			-f	Force boot, do not attempt OS shutdown	
			-console	During boot, switch to system console output	
	reset	Operator	-console	During reset, switch to system console output	
			[no switches]	Power off by issuing an orderly shutdown to the OS	
Operating System	shutdown	Operator	[no switches]	Issue an orderly shutdown to the OS	
			-f	Force OS shutdown. (Switch power off)	
Power Control	power	Operator	-on	Switch power on	
			-console	Set session to system console for power-on	
			-off	Switch power off (no OS shutdown)	
			-cycle	Switch power off, then on (no OS shutdown)	
			-state	Display target's current power state: On or Off End current session	On Off
Session	console (Only Telnet allows this command set)	Callback	[no switches]	Switch the session to system console. "~." switches to CLI mode.	
	exit (quit)	Callback	[no switches]	End current session	

Cmd Set	Cmd	Privilege Level	Option(s)	Short Description	Output
	logon (Only Terminal Mode allows this command set)	Callback	-u <username> -n -p <password>	Start a session	
Extended	network	Operator	[no switches]	Displays the current network channel configuration of the BMC. If not LAN channel, it uses the primary NIC channel	IP address: x.x.x.x IP address Source: (Static DHCP BIOS Other) MAC address: xx:xx:xx:xx:xx:xx Subnet Mask: x.x.x.x Gateway: x.x.x.x
			-p <params>	Display the requested parameter (mac ip subnet gateway).	
	interrupt	Operator	[no switches]	Causes the BMC to generate an IPMI diagnostic interrupt (NMI for IA-32 systems)	
			-console	Switches to console mode and issues the interrupt	
	version	Callback	[no switches]	Displays the Native CLI version number	Native CLI version x.x.
	displaylog ¹	Operator	[no switches]	Displays SEL records	Hex characters for each record, separated by linefeed characters.
			- s <first><last>	Set of SEL records	
	clearlog ¹	Operator	[no switches]	Clears SEL records	
			-state	Displays SEL erasure status	Erasure in Progress Erasure Completed
	Alarm Device	alarm ¹	Operator	[no switches]	Manage system alarms
-q Syntax: {-g <id> {-o <id>} {-a <id>}} {-l <severity>} {-p} }				Query current Alarm state	AlarmGenID=id AlarmSW=[Y N] AlarmSWID=[id NA] AlarmID=id AlarmSev=severity AlarmPWR=[Y N]

¹ The displaylog, clearlog and alarm commands are not yet implemented and are listed for informational purposes only.

Cmd Set	Cmd	Privilege Level	Option(s)	Short Description	Output		
			-s Syntax: alarm -s -a <id> -l <severity>	Set or add an alarm with -a id -l severity. The generator id will always be 47h and the software originator id will be 0.	New alarm added to BMC TAM alarm database. Request alarm matches existing BMC TAM alarm database record. Request alarm updated an existing BMC TAM alarm database record. BMC TAM alarm database is full. Request alarm record bumped because of lower priority. BMC TAM alarm database is full. Request alarm record bumped existing record.		
			-c syntax: {{-g <id> {-o <id>} {-a <id>}} {-l <severity>}} -all	Clear alarm record(s)	Alarm ID id cleared (Generator ID id)		
			-g <generator id>	identifying alarm generator			
			-o <sw originator id>	identifying alarm originator, if SW			
			-a <alarm id>	identifying numeric index of alarm			
			-l <severity>	Alarm severity level CRT(critical) MJR (major) MNR(minor)			
			-p	alarms that are related to power			
			-all	All alarm records			
			identify	Operator	[no switches]	Activate a local indicator to identify target	
					-on [#]	Switch indicator on; turn off in # seconds	
			-off	Switch indicator off			
			-state	Display current state of indicator	On (Application) On (Button) Off		

Performing a graceful operating system shutdown requires an ISM operating system agent be present. If this agent is not present or is unable to/does not respond after 7 seconds, an error message will be displayed and the command will terminate. No reset or power off will be performed. Graceful shutdown commands will not perform hard resets or power off if operating system shutdown does not complete.

The BMC does not verify completion of a graceful shutdown and that the power off/reset operation was successful within the operating system.

5.5.3 Configuration

An OEM configuration parameter is defined for configuring the Serial channel to use the CLI syntax in Terminal Mode. Based on this parameter value (CCLI/TMCLI) of the serial channel, the CLI command processor processes the terminal mode text commands differently by looking into its command table.

5.5.4 Terminal Mode and CLI

Terminal mode in the serial (EMP) channel uses the CLI syntax parameter in the OEM command configuration. The handshake for Terminal Mode is defined in the IPMI specification and remains the same between the IPMI TMCLI text commands and CCLI text commands. Only Terminal mode supports both CLI syntaxes.

5.5.5 Telnet and CCLI

Telnet protocol in the net stack always uses the Common CLI command syntax for its text commands. It creates an IPMI session with the input provided at the login/password prompts. Once the session is created, it allows executing the CCLI text commands. It closes the IPMI session when it receives “exit/logout” commands. Table 34 provides details about the I/O sequence.

5.5.6 Web Console and CCLI

HTTP protocol in the net stack always uses the Common CLI command syntax for its text commands. It also, supports IPMI hex commands. It creates an IPMI session with the input provided at the login/password prompts. It creates a session and executes a single command. The session is terminated after each response is built for return to the user. Table 34 provides details about the I/O sequence.

Table 34. Input and Output Sequence

		Terminal Mode		Telnet	HTTP
		TMCLI	CCLI	CCLI	CCLI
Input	Start	[
	Stop]			
	Input Token	SYS			
	Abort Command		TBD	TBD	none
	Console-to-Command	Esc+(Esc+(~.	TBD
Output	Start	[
	Stop]			
	Welcome Screen			Welcome to BMC-Telnet Server	
	Prompt		CCLI>	CCLI>	none
	Login prompt		Login	Login	provided by browser
	Password prompt		Password	Password	provided by browser
	Clear Screen		Esc[2J	Esc[2J	none

5.5.7 Scripting Support

Common CLI and Telnet are designed for use with scripts. Telnet provides a prompt for controlling the input for CLI text commands. A script should wait for the prompt to appear after each command before sending the next command to ensure proper execution.

5.6 SNMP

The Intel® Management Module – Advanced Edition SNMP feature provides for basic server management information through Platform Event Traps (PET). The BMC provides support for SNMP v2c.

5.6.1 PET Trap Enhancements

The v1.0 PET format was enhanced in the Advanced Edition BMC to support SNMP. The following enhancements were made. This is in addition to the standard PET format, which is also supported.

- Creation of an alternate PET format in which the variable bindings are presented in separate varbind entries (as opposed to the singular varbind field of the v1.0 PET format)
- Reading each of the separate variable bindings fields through conveniently mapped variables

The alternate PET format is selectable through an OEM parameter of the PEF configuration commands. The variable bindings associated with the most recent PET trap received are accessible via MIB browser examination of a table of PET varbinds. The user is directed with MIB descriptions to specify an instance value equal to 0 to access these variables (or just use a GETNEXT operation).

Also, the PET trap may be configured to append the SDR ID String field of the record with the matching sensor number to the end of the trap. When configured to place each variable binding field into its own varbind, the SDR name is the seventeenth varbind.

5.7 Keyboard/Video/Mouse (KVM) Redirection

The Remote KVM (keyboard, video, mouse) feature provides a means to capture and redirect managed server video graphics to a remote console (running KVM software) over the network. Video memory from the managed server is compressed, encrypted and sent over the network using TCP/IP to a remote client. In addition, keyboard and mouse activity from the remote console can be received for input to the managed server.

Specific details about the use of KVM are available in the ISM Install and Users Guide.

The KVM feature is implemented as OEM payload types over RMCP+ as well as OEM Explicit payload types with specific payload ids. Authentication and session setup is handled per the IPMI 2.0 specification. The payload types are:

- 20h Video Redirection
- 21h PS/2 Keyboard and Mouse Redirection

The OEM Explicit payload ids (IANA: Intel (343))

- 0100h Video Redirection
- 0200h PS/2 Keyboard and Mouse Redirection

Support for the feature for a particular LAN channel can be determined by software using the IPMI 2.0 *Get Channel Payload Support* command using the above defined payload types. The KVM feature can be enabled or disabled independently for each LAN channel via a KVM configuration parameter using the *Set KVM Configuration Parameter* command. Per-user ability to use the feature can be configured via the IPMI 2.0 *Get/Set User Payload Access* commands.

5.7.1 KVM Data Encryption

The BMC supports encryption of the KVM data using the RC4 encryption protocol. The encryption is limited to keyboard and video data. This encryption can be enabled or disabled by configuration of the KVM viewer. To enable/disable the encryption of KVM data, the KVMConf.properties file in the KVM install directory must be edited.

By default, the Keyboard data is encrypted and Video data is not encrypted.

The following option in the KVMConf.properties file enables/disables the Keyboard encryption:

```
KEYBOARD_ENCRYPTION_ENABLED = TRUE
```

Setting this option to 'FALSE' will disable keyboard data encryption.

The following option in the KVMConf.properties file enables/disables the video encryption.

```
VIDEO_ENCRYPTION_ENABLED = FALSE
```

Setting this option to 'TRUE' will enable video encryption.


5.7.2 Intel Advanced Remote Server Control

Intel Advanced Remote Server Control (IARSC) is a server management tool that enables a virtual presence solution that provides a remote administrator full control over the managed server's display, keyboard and mouse. Intel Advanced Remote Server Control connects to the Advanced Module and is a component of Intel Server Management 8. Intel Server Management 8 is fully documented in the Intel Server Management Install and Users guide that ships with that software product.

Note: Mouse acceleration must be disabled when used with KVM (required for mouse pointer synchronization between client and server; enabled by default in Windows* and Linux* operating systems; see operating system documentation for information on disabling mouse acceleration or refer to the ISM install and users guide).

5.7.2.1 Local User Activity

If the local user at the target server location takes control of the video, the Intel Advanced Remote Server Control will get a notification. The viewer will then show a “Local User Active” message on the status bar. The remote user will be able to see the changes happening on the local system on the remote viewer. But the remote user will not be able to take any action on the screen. This will be evident in the following ways:

- The menus on the menu pane will be grayed out.
- The macros on the macro pane will be grayed out. Clicking on them will not show any change on the presentation pane.
- The mouse alignment will go away. Moving the remote mouse will not show a corresponding move of the local mouse. The remote mouse will be changed to an Hourglass or a “Working in Background” icon ()

The viewer will wait about 20 seconds before changing the status back. The “Local User Active” message on the status bar will disappear. The mouse icon will be changed back to the one set by the remote user originally. The remote mouse pointer may not be aligned with the local mouse pointer. The menus and macros will be enabled again.

6. Error Reporting and Handling

The Intel® Management Module conforms to the Intelligent Platform Management Specification (IPMI) 2.0 error reporting format. Event entries are logged in the BMC's non-volatile System Event Log (SEL).

6.1 SEL Event Format

The standard SEL record format is defined in section 32.1 of the IPMI 2.0 specification, but is reproduced here for convenience. See Table 35 for the standard SEL Event format.

Table 35. SEL Event Records

Byte	Field	Description
1 2	Record ID	ID used for SEL Record access. The Record ID values 0000h and FFFFh have special meaning in the Event Access commands and must not be used as Record ID values for stored SEL Event Records.
3	Record Type	[7:0] Record Type 02h = system event record C0h-DFh = OEM timestamped, bytes 8-16 OEM defined E0h-FFh = OEM non-timestamped, bytes 4-16 OEM defined
4 5 6 7	Timestamp	Time when event was logged. LS byte first.
8 9	Generator ID	RqSA & LUN if event was generated from IPMB. Software ID if event was generated from system software. Byte 1 [7:1] 7-bit I2C . Slave address, or 7-bit system software ID [0] 0b = ID is IPMB slave address 1b = system software ID Byte 2 [7:4] Channel number. Channel that event message was received over. 0h if the event message was received via the system interface, primary IPMB, or internally generated by the BMC. (New for IPMI v1.5. These bits were reserved in IPMI v1.0) [3:2] reserved. Write as 00b. [1:0] IPMB device LUN if byte 1 holds slave address. 00b otherwise.
10	EvM Rev	Event Message format version (=04h for events in this specification, 03h for IPMI v1.0 Event Messages.)
11	Sensor Type	Sensor Type Code for sensor that generated the event
12	Sensor #	Number of sensor that generated the event

Byte	Field	Description
13	Event Dir Event Type	Event Dir [7] 0b = Assertion event 1b = Deassertion event Event Type Type of trigger for the event, e.g. critical threshold going high, state asserted, etc. Also indicates class of the event. E.g. discrete, threshold, or OEM. The Event Type field is encoded using the Event/Reading Type Code. See section 6.3.5. [6:0] Event Type Code
14	Event Data 1	See section 6.3.6
15	Event Data 2	See section 6.3.6
16	Event Data 3	See section 6.3.6

6.1.1 OEM Event Types

The IPMI 2.0 specification also defines two OEM event types, defined in section 32.2 and 32.3 of the IPMI 2.0 specification. The Intel® Management Module does not use these OEM event types for any type of event.

6.2 Timestamp Format

The BMC timestamp is an unsigned 32-bit value representing the local time as the number of seconds from 00:00:00, January 1, 1970. This format is sufficient to maintain timestamping with 1-second resolution past the year 2100. This is based on a long standing UNIX-based standard for time keeping, which represents time as the number of seconds from 00:00:00, January 1, 1970 GMT.

The BMC timestamp clock behavior is described in detail in section 4.10.4.

6.3 SEL Translation Process

This section will discuss the process for decoding a 16 byte HEX SEL entry. The primary data bytes used for translating SEL messages are event data bytes 8-16.

6.3.1 Generator ID

The generator ID field is used to indicate the source of the SEL message. The Intel® Management Module BMC uses a generator ID of 20 for all internally generated events. See Table 36 for the generator IDs used by the server board BIOS.

Table 36. BMC Generator IDs

Generator ID #	Used By	Description
20	BMC	Used for standard events logged by BMC
21	BMC	Used by BMC in conjunction with <i>Platform Event Message</i> command sent by system software.

6.3.2 EvM Rev (Event Message Revision)

The EvM Rev field is 04h for all events logged by the Intel® Management Module, per the IPMI 2.0 specification.

6.3.3 Sensor Type

The IPMI 2.0 specification lists all standard sensor types in table 42-3 of that specification. This table is reproduced here as Table 37 for your convenience. When translating a SEL entry, compare byte 11 of the SEL record to the second column of Table 37 to determine the type of sensor that generated the event.

Table 37. Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
Reserved	00h	—	reserved
Temperature	01h	—	Temperature
Voltage	02h	—	Voltage
Current	03h	—	Current
Fan	04h	—	Fan
Physical Security (Chassis Intrusion)	05h	00h	00h: General Chassis Intrusion
		01h	01h: Drive Bay intrusion
		02h	02h: I/O card area intrusion
		03h	03h: Processor area intrusion
		04h	04h: LAN Leash Lost (system is unplugged from LAN) The Event Data 2 field can be used to identify which network controller the leash was lost on where 00h corresponds to the first (or only) network controller.
		05h	05h: Unauthorized dock/undock
		06h	06h: Fan area intrusion (supports detection of hot plug fan tampering)
Platform Security Violation Attempt	06h	00h	00h: Secure Mode (Front Panel Lockout) Violation attempt
		01h	01h: Pre-boot Password Violation - user password
		02h	02h: Pre-boot Password Violation attempt - setup password
		03h	03h: Pre-boot Password Violation - network boot password
		04h	04h: Other pre-boot Password Violation
		05h	05h: Out-of-band Access Password Violation
Processor	07h	00h	00h: IERR
		01h	01h: Thermal Trip
		02h	02h: FRB1/BIST failure
		03h	03h: FRB2/Hang in POST failure (used hang is believed to be due or related to a processor failure. Use System Firmware Progress sensor for other BIOS hangs.)
		04h	04h: FRB3/Processor Startup/Initialization failure (CPU didn't start)

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		05h	05h: Configuration Error
		06h	06h: SM BIOS 'Uncorrectable CPU-complex Error'
		07h	07h: Processor Presence detected
		08h	08h: Processor disabled
		09h	09h: Terminator Presence Detected
		0Ah	Processor Automatically Throttled (processor throttling triggered by a hardware-based mechanism operating independent from system software, such as automatic thermal throttling or throttling to limit power consumption.)
Power supply (also used for power converters [e.g. DC-to-DC converters] and VRMs [voltage regulator modules]).	08h	00h	00h: Presence detected
		01h	01h: Power supply failure detected
		02h	02h: Predictive failure
		03h	03h: Power supply input lost (AC/DC)[2]
		04h	04h: Power supply input lost or out-of-range
		05h	05h: Power supply input out-of-range, but present
		06h	Configuration error. The Event Data 3 field provides a more detailed definition of the error: 7:4 = Reserved for future definition, set to 0000b 3:0 = Error Type, one of 0h = Vendor mismatch, for power supplies that include this status. (Typically, the system OEM defines the vendor compatibility criteria that drives this status). 1h = Revision mismatch, for power supplies that include this status. (Typically, the system OEM defines the vendor revision compatibility that drives this status). 2h = Processor missing. For processor power supplies (typically DC-to-DC converters or VRMs), there's usually a one-to-one relationship between the supply and the CPU. This offset can indicate the situation where the power supply is present but the processor is not. This offset can be used for reporting that as an unexpected or unsupported condition. Others = Reserved for future definition
Power Unit	09h	00h	00h: Power Off / power down
		01h	01h: Power cycle
		02h	02h: 240VA power down
		03h	03h: Interlock power down
		04h	04h: AC lost
		05h	05h: Soft power Control Failure (unit did not respond to request to turn on)
		06h	06h: Power unit Failure detected
		07h	07h: Predictive Failure
Cooling Device	0Ah	—	—
Other Units-based Sensor (per units given in SDR)	0Bh	—	—
Memory	0Ch	00h	00h: Correctable ECC / other correctable memory error
		01h	01h: Uncorrectable ECC / other uncorrectable memory error

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		02h	02h: Parity
		03h	03h: Memory Scrub Failed (stuck bit)
		04h	04h: Memory Device Disabled
		05h	05h: Correctable ECC / other correctable memory error logging limit reached
		06h	06h: Presence detected. Indicates presence of entity associated with the sensor. Typically the entity will be a 'memory module' or other entity representing a physically replaceable unit of memory.
		07h	07h: Configuration error. Indicates a memory configuration error for the entity associated with the sensor. This can include when a given implementation of the entity is not supported by the system (e.g., when the particular size of the memory module is unsupported) or that the entity is part of an unsupported memory configuration (e.g. the configuration is not supported because the memory module does not match other memory modules).
		08h	08h: Spare. Indicates entity associated with the sensor represents a 'spare' unit of memory. The Event Data 3 field can be used to provide an event extension code, with the following definition: Event Data 3 [7:0] Memory module/device (e.g. DIMM/SIMM/RIMM) identification, relative to the entity that the sensor is associated with (if SDR provided for this sensor).
Drive Slot (Bay)	0Dh	—	—
POST Memory Resize	0Eh	—	—
System Firmware Progress (formerly POST Error)	0Fh	00h	System Firmware Error (POST Error) The Event Data 2 field can be used to provide an event extension code, with the following definition: 00h Unspecified. 01h No system memory is physically installed in the system. 02h No usable system memory, all installed memory has experienced an unrecoverable failure. 03h Unrecoverable hard-disk/ATAPI/IDE device failure. 04h Unrecoverable system-board failure. 05h Unrecoverable diskette subsystem failure. 06h Unrecoverable hard-disk controller failure. 07h Unrecoverable PS/2 or USB keyboard failure. 08h Removable boot media not found 09h Unrecoverable video controller failure 0Ah No video device detected 0Bh Firmware (BIOS) ROM corruption detected 0Ch CPU voltage mismatch (processors that share same supply have mismatched voltage requirements) 0Dh CPU speed matching failure 0Eh to FFh reserved

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		01h	System Firmware Hang (uses same Event Data 2 definition as following System Firmware Progress offset)
		02h	<p>System Firmware Progress</p> <p>The Event Data 2 field can be used to provide an event extension code, with the following definition:</p> <p>00h Unspecified.</p> <p>01h Memory initialization.</p> <p>02h Hard-disk initialization</p> <p>03h Secondary processor(s) initialization</p> <p>04h User authentication</p> <p>05h User-initiated system setup</p> <p>06h USB resource configuration</p> <p>07h PCI resource configuration</p> <p>08h Option ROM initialization</p> <p>09h Video initialization</p> <p>0Ah Cache initialization</p> <p>0Bh SM Bus initialization</p> <p>0Ch Keyboard controller initialization</p> <p>0Dh Embedded controller/management controller initialization</p> <p>0Eh Docking station attachment</p> <p>0Fh Enabling docking station</p> <p>10h Docking station ejection</p> <p>11h Disabling docking station</p> <p>12h Calling operating system wake-up vector</p> <p>13h Starting operating system boot process, e.g. calling Int 19h</p> <p>14h Server board initialization</p> <p>15h reserved</p> <p>16h Floppy initialization</p> <p>17h Keyboard test</p> <p>18h Pointing device test</p> <p>19h Primary processor initialization</p> <p>1Ah to FFh reserved</p>
Event Logging Disabled	10h	00h	<p>Correctable Memory Error Logging Disabled</p> <p>Event Data 2</p> <p>[7:0] Memory module/device (e.g. DIMM/SIMM/RIMM) identification, relative to the entity that the sensor is associated with (if SDR provided for this sensor).</p>

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		01h	Event 'Type' Logging Disabled. Event Logging is disabled for following event/reading type and offset has been disabled. Event Data 2 Event/Reading Type Code Event Data 3 [7:6] reserved. Write as 00b. [5] 1b = logging has been disabled for all events of given type [4] 1b = assertion event, 0b = deassertion event [3:0] Event Offset
		02h	Log Area Reset/Cleared
		03h	All Event Logging Disabled
		04h	SEL Full. If this is used to generate an event, it is recommended that this be generated so that this will be logged as the last entry in the SEL. If the SEL is very small, an implementation can elect to generate this event after the last entry has been placed in the SEL to save space. In this case, this event itself would not get logged, but could still trigger actions such as an alert via PEF. Note: An application can use the <i>Get SEL Info</i> command to determine whether the SEL is full or not. Since <i>Get SEL Info</i> is a mandatory command, this provides a cross-platform way to get that status.
		05h	SEL Almost Full. If Event Data 3 is not provided, then by default this event represents the SEL has reached a point of being 75% or more full. For example, if the SEL supports 215 entries, the 75% value would be 161.25 entries. Therefore, the event would be generated on the 162nd entry. Note: This event would be logged as the 163rd entry. Event Data 3 Contains hex value from 0 to 100 decimal (00h to 64h) representing the % of which the SEL is filled at the time the event was generated: 00h is 0% full (SEL is empty), 64h is 100% full, etc.
Watchdog 1	11h		This sensor is provided to support IPMI v0.9 to v1.0 transition. This is deprecated in IPMI v1.5. See sensor 23h for recommended definition of watchdog sensor for new v1.0 and for IPMI v1.5 implementations.
		00h	00h: BIOS Watchdog Reset
		01h	01h: OS Watchdog Reset
		02h	02h: OS Watchdog Shut Down
		03h	03h: OS Watchdog Power Down
		04h	04h: OS Watchdog Power Cycle
		05h	05h: OS Watchdog NMI / Diagnostic Interrupt
		06h	06h: OS Watchdog Expired, status only
		07h	07h: OS Watchdog pre-timeout Interrupt, non-NMI
System Event	12h	00h	System Reconfigured
		01h	01h: OEM System Boot Event

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		02h	02h: Undetermined system hardware failure This event would typically require system-specific diagnostics to determine FRU / failure type
		03h	03h: Entry added to Auxiliary Log See section Error! Reference source not found. for more information on <i>Get/Set Auxiliary Log Status</i> command support. Event Data 2 [7:4] Log Entry Action 0h = entry added 1h = entry added because event did not be map to standard IPMI event 2h = entry added along with one or more corresponding SEL entries 3h = log cleared 4h = log disabled 5h = log enabled all other = reserved [3:0] Log Type 0h = MCA Log 1h = OEM 1 2h = OEM 2 all other = reserved
		04h	PEF Action Event Data 2 The following bits reflect the PEF Actions that are about to be taken after the event filters have been matched. The event is captured before the actions are taken. [7:6] reserved [5] 1b = Diagnostic Interrupt (NMI) [4] 1b = OEM action [3] 1b = Power cycle [2] 1b = Reset [1] 1b = Power off [0] 1b = Alert
Critical Interrupt	13h	00h	00h: Front Panel NMI / Diagnostic Interrupt
		01h	01h: Bus Timeout
		02h	02h: I/O channel check NMI
		03h	03h: Software NMI
		04h	04h: PCI PERR
		05h	05h: PCI SERR
		06h	06h: EISA Fail Safe Timeout
		07h	07h: Bus Correctable Error
		08h	08h: Bus Uncorrectable Error
		09h	09h: Fatal NMI (port 61h, bit 7)
		0Ah	0Ah: Bus Fatal Error

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
Button / Switch	14h	00h	00h: Power Button pressed
		01h	01h: Sleep Button pressed
		02h	02h: Reset Button pressed
		03h	FRU latch open (Switch indicating FRU latch is in 'unlatched' position and FRU is mechanically removable)
		04h	FRU service request button (1 = pressed, service, e.g. removal/replacement, requested)
Module / Board	15h	—	—
Microcontroller / Coprocessor	16h	—	—
Add-in card	17h	—	—
Chassis	18h	—	—
Chipset	19h	00h	<p>Soft Power Control Failure (chip set did not respond to BMC request to change system power state). This offset is similar to offset 05h for a power unit, except that the power unit event is only related to a failure to power up, while this event corresponds to any system power state change directly requested via the BMC.</p> <p>Event Data 2</p> <p>The Event Data 2 field for this command can be used to provide additional information on the type of failure with the following definition:</p> <p>Requested power state</p> <p>00h = S0 / G0 "working"</p> <p>01h = S1 "sleeping with system h/w & processor context maintained"</p> <p>02h = S2 "sleeping, processor context lost"</p> <p>03h = S3 "sleeping, processor & h/w context lost, memory retained."</p> <p>04h = S4 "non-volatile sleep / suspend-to disk"</p> <p>05h = S5 / G2 "soft-off"</p> <p>06h = S4 / S5 soft-off, particular S4 / S5 state cannot be determined</p> <p>07h = G3 / Mechanical Off</p> <p>08h = Sleeping in an S1, S2, or S3 states (used when particular S1, S2, S3 state cannot be determined)</p> <p>09h = G1 sleeping (S1-S4 state cannot be determined)</p> <p>0Ah = S5 entered by override</p> <p>0Bh = Legacy ON state</p> <p>0Ch = Legacy OFF state</p> <p>0Dh = reserved</p> <p>Event Data 3</p> <p>The Event Data 3 field for this command can be used to provide additional information on the type of failure with the following definition:</p> <p>Power state at time of request</p> <p>00h = S0 / G0 "working"</p>

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
			01h = S1 "sleeping with system h/w & processor context maintained" 02h = S2 "sleeping, processor context lost" 03h = S3 "sleeping, processor & h/w context lost, memory retained." 04h = S4 "non-volatile sleep / suspend-to disk" 05h = S5 / G2 "soft-off" 06h = S4 / S5 soft-off, particular S4 / S5 state cannot be determined 07h = G3 / Mechanical Off 08h = Sleeping in an S1, S2, or S3 states (used when particular S1, S2, S3 state cannot be determined) 09h = G1 sleeping (S1-S4 state cannot be determined) 0Ah = S5 entered by override 0Bh = Legacy ON state 0Ch = Legacy OFF state 0Dh = unknown
Other FRU	1Ah	—	—
Cable / Interconnect	1Bh	—	—
Terminator	1Ch	—	—
System Boot Initiated	1Dh	00h	00h: Initiated by power up
		01h	01h: Initiated by hard reset
		02h	02h: Initiated by warm reset
		03h	03h: User requested PXE boot
		04h	04h: Automatic boot to diagnostic
Boot Error	1Eh	00h	00h: No bootable media
		01h	01h: Non-bootable diskette left in drive
		02h	02h: PXE Server not found
		03h	03h: Invalid boot sector
		04h	04h: Timeout waiting for user selection of boot source
OS Boot	1Fh	00h	00h: A: boot completed
		01h	01h: C: boot completed
		02h	02h: PXE boot completed
		03h	03h: Diagnostic boot completed
		04h	04h: CD-ROM boot completed
		05h	05h: ROM boot completed
		06h	06h: Boot completed - boot device not specified
OS Critical Stop	20h	00h	00h: Stop during OS load / initialization
		01h	01h: Run-time stop
Slot / Connector	21h	00h	00h: Fault Status asserted
		01h	01h: Identify Status asserted
		02h	02h: Slot / Connector Device installed/attached (This can include dock events)

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		03h	03h: Slot / Connector Ready for Device Installation. Typically, this means that the slot power is off. The Ready for Installation, Ready for Removal, and Slot Power states can transition together, depending on the slot implementation.
		04h	04h: Slot/Connector Ready for Device Removal
		05h	05h: Slot Power is Off
		06h	06h: Slot / Connector Device Removal Request - This is typically connected to a switch that becomes asserted to request removal of the device)
		07h	07h: Interlock asserted - This is typically connected to a switch that mechanically enables/disables power to the slot, or locks the slot in the 'Ready for Installation / Ready for Removal states' - depending on the slot implementation. The asserted state indicates that the lock-out is active.
		08h	08h: Slot is Disabled
		09h	09h: Slot holds spare device The Event Data 2 and 3 fields can be used to provide an event extension code, with the following definition: Event Data 2 7 reserved 6:0 Slot/Connector Type 0 PCI 1 Drive Array 2 External Peripheral Connector 3 Docking 4 other standard internal expansion slot 5 slot associated with entity specified by Entity ID for sensor 6 AdvancedTCA 7 DIMM/memory device 8 FAN 9 PCI Express* 10 SCSI (parallel) 11 SATA / SAS all other = reserved Event Data 3 7:0 Slot/Connector Number
		System ACPI Power State	22h
		01h	01h: S1 "sleeping with system h/w & processor context maintained"
		02h	02h: S2 "sleeping, processor context lost"
		03h	03h: S3 "sleeping, processor & h/w context lost, memory retained."
		04h	04h: S4 "non-volatile sleep / suspend-to disk"
		05h	05h: S5 / G2 "soft-off"
		06h	06h: S4 / S5 soft-off, particular S4 / S5 state cannot be determined

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		07h	07h: G3 / Mechanical Off
		08h	08h: Sleeping in an S1, S2, or S3 states (used when particular S1, S2, S3 state cannot be determined)
		09h	09h: G1 sleeping (S1-S4 state cannot be determined)
		0Ah	0Ah: S5 entered by override
		0Bh	0Bh: Legacy ON state
		0Ch	0Ch: Legacy OFF state
		0Eh	0Eh: Unknown
Watchdog 2	23h		This sensor is recommended for new IPMI v1.0 and later implementations.
		00h	00h: Timer expired, status only (no action, no interrupt)
		01h	01h: Hard Reset
		02h	02h: Power Down
		03h	03h: Power cycle
		04h-07h	04h: reserved
		08h	08h: Timer interrupt The Event Data 2 field for this command can be used to provide an event extension code, with the following definition: 7:4 Interrupt type 0h = none 1h = SMI 2h = NMI 3h = Messaging Interrupt Fh = unspecified all other = reserved 3:0 Timer use at expiration: 0h = reserved 1h = BIOS FRB2 2h = BIOS/POST 3h = OS Load 4h = SMS/OS 5h = OEM Fh = unspecified all other = reserved
Platform Alert	24h		This sensor can be used for returning the state and generating events associated with alerts that have been generated by the platform mgmt. subsystem
		00h	platform generated page
		01h	platform generated LAN alert
		02h	Platform Event Trap generated, formatted per IPMI PET specification
		03h	platform generated SNMP trap, OEM format

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
Entity Presence	25h		This sensor type provides a mechanism that allows a management controller to direct system management software to ignore a set of sensors based on detecting that presence of an entity. This sensor type is not typically used for event generation - but to just provide a present reading.
		00h	Entity Present. This indicates that the Entity identified by the Entity ID for the sensor is present.
		01h	Entity Absent. This indicates that the Entity identified by the Entity ID for the sensor is absent. If the entity is absent, system management software should consider all sensors associated with that Entity to be absent as well - and ignore those sensors.
		02h	Entity Disabled. The Entity is present, but has been disabled. A deassertion of this event indicates that the Entity has been enabled.
Monitor ASIC / IC	26h	—	—
LAN	27h	00h	00h: LAN Heartbeat Lost
		01h	01h: LAN Heartbeat
Management Subsystem Health	28h	00h	00h: Sensor access degraded or unavailable
		01h	01h: Controller access degraded or unavailable
		02h	02h: Management controller off-line
		03h	03h: Management controller unavailable
Battery	29h	00h	00h: Battery low (predictive failure)
		01h	01h: Battery failed
		02h	02h: Battery presence detected
Session Audit	2Ah	00h	Session Activated
		01h	<p>Session Deactivated</p> <p>The Event Data 2 and 3 fields can be used to provide an event extension code, with the following definition:</p> <p>Event Data 2:</p> <p>7:6 reserved</p> <p>5:0 User ID for user that activated session. 00_0000b = unspecified.</p> <p>Event Data 3:</p> <p>7:6 reserved</p> <p>5:4 Deactivation cause</p> <p>00b = Session deactivation cause unspecified. This value is also used for Session Activated events.</p> <p>01b = Session deactivated by the <i>Close Session</i> command</p> <p>10b = Session deactivated by timeout</p> <p>11b = Session deactivated by configuration change</p> <p>3:0 Channel number that session was activated/deactivated over. Use channel number that session was activated over if a session was closed for an unspecified reason, a timeout, or a configuration change.</p>
Version Change	2Bh	00h	Hardware change detected with associated Entity. Informational. This offset does not imply whether the hardware change was successful or not. Only that a change occurred.

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		01h	Firmware or software change detected with associated Entity. Informational. Success or failure not implied.
		02h	Hardware incompatibility detected with associated Entity.
		03h	Firmware or software incompatibility detected with associated Entity.
		04h	Entity is of an invalid or unsupported hardware version.
		05h	Entity contains an invalid or unsupported firmware or software version.
		06h	Hardware Change detected with associated Entity was successful. (deassertion event means 'unsuccessful').
		07h	<p>Software or F/W Change detected with associated Entity was successful. (deassertion event means 'unsuccessful')</p> <p>Event data 2 can be used for additional event information on the type of version change, with the following definition:</p> <p>Event Data 2:</p> <p>7:0 Version change type</p> <ul style="list-style-type: none"> 00h unspecified 01h management controller device ID (change in one or more fields from 'Get Device ID') 02h management controller firmware revision 03h management controller device revision 04h management controller manufacturer ID 05h management controller IPMI version 06h management controller auxiliary firmware ID 07h management controller firmware boot block 08h other management controller firmware 09h system firmware (EFI / BIOS) change 0Ah SMBIOS change 0Bh operating system change 0Ch operating system loader change 0Dh service or diagnostic partition change 0Eh management software agent change 0Fh management software application change 10h management software middleware change 11h programmable hardware change (e.g. FPGA) 12h board/FRU module change (change of a module plugged into associated entity) 13h board/FRU component change (addition or removal of a replaceable component on the board/FRU that is not tracked as a FRU) 14h board/FRU replaced with equivalent version 15h board/FRU replaced with newer version 16h board/FRU replaced with older version 17h board/FRU hardware configuration change (e.g. strap, jumper, cable change, etc.)
FRU State	2Ch	00h	FRU Not Installed

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
		01h	FRU Inactive (in standby or 'hot spare' state)
		02h	FRU Activation Requested
		03h	FRU Activation In Progress
		04h	FRU Active
		05h	FRU Deactivation Requested
		06h	FRU Deactivation In Progress
		07h	FRU Communication Lost
			<p>The Event Data 2 field for this command can be used to provide the cause of the state change and the previous state:</p> <p>7:4 Cause of state change</p> <p>0h = Normal State Change.</p> <p>1h = Change Commanded by software external to FRU.</p> <p>2h = State Change due to operator changing a Handle latch.</p> <p>3h = State Change due to operator pressing the hot swap push button.</p> <p>4h = State Change due to FRU programmatic action.</p> <p>5h = Communication Lost.</p> <p>6h = Communication Lost due to local failure.</p> <p>7h = State Change due to unexpected extraction.</p> <p>8h = State Change due to operator intervention/update.</p> <p>9h = Unable to compute IPMB address.</p> <p>Ah = Unexpected Deactivation.</p> <p>Fh = State Change, Cause Unknown.</p> <p>All other = reserved</p> <p>3:0 Previous state offset value (return offset for same state as present state if previous state is unknown)</p> <p>All other = reserved.</p>
Reserved	remaining	—	—
OEM RESERVED	C0h-FFh	—	—

6.3.4 Sensor Number

The sensor number field of the SEL corresponds to the specific sensor number that generated the event. The sensor numbers are located in the sensor data records (SDRs) of the BMC and may vary by server board and platform. The specific board or platform TPS may include a list of sensors, including the sensor numbers.

If this information is not available, the sensor numbers are alternatively available as part of the platform FRUSDR package in a file with the extension 'SDR'.

6.3.5 Event Dir/Event Type

This byte includes information on the direction of the event, either asserted or de-asserted, as well as the type of event it is.

6.3.5.1 Event Direction

Bit 7 of the Event Dir/Event Type byte is used to determine if the event is actively being asserted (i.e. the event is happening), or if the event has de-asserted (i.e. the event has stopped to happen). Not all sensors will log de-assertion events. Support for de-assertion events is determined by the event mask bytes in the SDR for the specific sensor.

6.3.5.2 Event Type

Bits 6:0 of the Event Dir/Event Type are used to determine particulars about the event. These bits include information on which table to use to translate the remaining data bytes of the SEL record. Translation data for these bits are found in table 42-1 of the IPMI 2.0 specification. This table is reproduced here as Table 38 for your convenience. Compare the Event Type bits 6:0 to the second column of Table 38 below to determine the sensor class of the event.

Table 38. Event/Reading Type Code Ranges

Event / Reading Type Code category	7-bit Event / Reading Type Code Range	Sensor Class	Description
Unspecified	00h	n/a	Event/Reading Type unspecified.
Threshold	01h	Threshold	Threshold-based. Indicates a sensor that utilizes values that represent discrete threshold states in sensor access and/or events. The Event/Reading event offsets for the different threshold states are given in Table 40.
Generic	02h-0Ch	Discrete	Generic Discrete. Indicates a sensor that utilizes an Event/Reading Type code and State bit positions / event offsets from one of the sets specified for Discrete or 'digital' Discrete Event/Reading class in Table 40.
Sensor-specific	6Fh	Discrete	Sensor-specific Discrete. Indicates that the discrete state information is specific to the sensor type. State bit positions / event offsets for a particular sensor type are specified in the 'sensor-specific offset' column in Table 37.
OEM	70h-7Fh	OEM	OEM Discrete. Indicates that the discrete state information is specific to the OEM identified by the Manufacturer ID for the IPM device that is providing access to the sensor. See section 6.4 for platform-specific discrete states.

The sensor class determines how to interpret the remaining three data bytes according to Table 29-6 of the IPMI 2.0 specification. This table is reproduced below as Table 39 for your convenience.

Table 39. Event Request Message Event Data Field Contents

Sensor Class	Event Data
Threshold	Event Data 1 [7:6] 00b = unspecified byte 2 01b = trigger reading in byte 2 10b = OEM code in byte 2 11b = sensor-specific event extension code in byte 2 [5:4] 00b = unspecified byte 3 01b = trigger threshold value in byte 3 10b = OEM code in byte 3 11b = sensor-specific event extension code in byte 3 [3:0] Offset from Event/Reading Code for threshold event.
	Event Data 2 reading that triggered event, FFh or not present if unspecified.
	Event Data 3 threshold value that triggered event, FFh or not present if unspecified. If present, byte 2 must be present.
Discrete	Event Data 1 [7:6] 00b = unspecified byte 2 01b = previous state and/or severity in byte 2 10b = OEM code in byte 2 11b = sensor-specific event extension code in byte 2 [5:4] 00b = unspecified byte 3 01b = reserved 10b = OEM code in byte 3 11b = sensor-specific event extension code in byte 3 [3:0] Offset from Event/Reading Code for discrete event state
	Event Data 2 [7:4] Optional offset from 'Severity' Event/Reading Code. (0Fh if unspecified). [3:0] Optional offset from Event/Reading Type Code for previous discrete event state. (0Fh if unspecified.)
	Event Data 3 Optional OEM code. FFh or not present if unspecified.
OEM	Event Data 1 [7:6] 00b = unspecified in byte 2 01b = previous state and/or severity in byte 2 10b = OEM code in byte 2 11b = reserved [5:4] 00b = unspecified byte 3 01b = reserved 10b = OEM code in byte 3 11b = reserved [3:0] Offset from Event/Reading Type Code

Sensor Class	Event Data
	Event Data 2 [7:4] Optional OEM code bits or offset from 'Severity' Event/Reading Type Code. (0Fh if unspecified). [3:0] Optional OEM code or offset from Event/Reading Type Code for previous event state. (0Fh if unspecified).
	Event Data 3 Optional OEM code. FFh or not present or unspecified.

Compare the sensor class indicated by the Event Type bits to the left column in Table 39. This table describes how to interpret the remaining three data bytes.

Note: Events with the 'sensor-specific' event type use the 'discrete' sensor class.

6.3.6 Event Data 1-3

The event data bytes 1-3 include specific information about the events. The type of information contained in these data bytes depend on the sensor generating the event. Table 39 describes the proper translation for the three data bytes, depending on the sensor class indicated in the event type bits of the Event Dir/Event Type byte.

6.3.6.1 Event Data 1

Decoding event data byte 1 involves dividing the byte into three parts.

- Bits 7:6 of data byte 1 describe whether valid event data is included in event data byte 2. If these bits are left unspecified (00b), then any data in event data 2 can be ignored.
- Bits 5:4 of data byte 1 describe whether valid event data is included in event data byte 3. If these bits are left unspecified (00b), then any data in event data 3 can be ignored.
- Bits 3:0 of data byte 1 describe the event offset.

For sensors that have a type reading code of threshold or generic in the left column of Table 38, use table 42-2 of the IPMI 2.0 specification. This table is reproduced here as Table 40 for your convenience.

Compare the Event Type for the event as described in section 6.3.5.2 to the left column of Table 40, below. The 'Generic Offset' (bits 3:0 of data byte 1), or 'event offset' for that sensor describes the specific state the sensor was in when the event was generated. In the case of threshold based sensors, the event offset is an indication of the specific threshold that was crossed when the event was generated. For generic discrete sensors, such as redundancy sensors, the event offset describes the state the sensor is in based on the description for the specific event offset in Table 40.

Table 40. Generic Event/Reading Type Codes

Generic Event / Reading Type Code	Event / Reading Class	Generic Offset	Description
01h	Threshold	Threshold-based States	
		00h	00h: Lower Non-critical - going low
		01h	01h: Lower Non-critical - going high
		02h	02h: Lower Critical - going low
		03h	03h: Lower Critical - going high
		04h	04h: Lower Non-recoverable - going low
		05h	05h: Lower Non-recoverable - going high
		06h	06h: Upper Non-critical - going low
		07h	07h: Upper Non-critical - going high
		08h	08h: Upper Critical - going low
		09h	09h: Upper Critical - going high
		0Ah	0Ah: Upper Non-recoverable - going low
		0Bh	0Bh: Upper Non-recoverable - going high
02h	Discrete	DMI-based "Usage State" States	
		00h	00h: Transition to Idle
		01h	01h: Transition to Active
		02h	02h: Transition to Busy
03h	'digital' Discrete	Digital / Discrete Event States	
		00h	State Deasserted
04h	'digital' Discrete	01h	State Asserted
		00h	Predictive Failure deasserted
05h	'digital' Discrete	01h	Predictive Failure asserted
		00h	Limit Not Exceeded
06h	'digital' Discrete	01h	Limit Exceeded
		00h	Performance Met
07h	Discrete	01h	Performance Lags
		Severity Event States	
07h	Discrete	00h	00h: Transition to OK
		01h	01h: Transition to Non-Critical from OK
		02h	02h: Transition to Critical from less severe
		03h	03h: Transition to Non-recoverable from less severe
		04h	04h: Transition to Non-Critical from more severe
		05h	05h: Transition to Critical from Non-recoverable
		06h	06h: Transition to Non-recoverable
		07h	07h: Monitor
		08h	08h: Informational

08h	'digital' Discrete	Availability Status States	
		00h	00h: Device Removed / Device Absent
		01h	01h: Device Inserted / Device Present
09h	'digital' Discrete	00h	00h: Device Disabled
		01h	01h: Device Enabled
0Ah	Discrete	00h	00h: Transition to Running
		01h	01h: Transition to In Test
		02h	02h: Transition to Power Off
		03h	03h: Transition to On Line
		04h	04h: Transition to Off Line
		05h	05h: Transition to Off Duty
		06h	06h: Transition to Degraded
		07h	07h: Transition to Power Save
		08h	08h: Install error
0Bh	Discrete	Other Availability Status States	
			Redundancy States
		00h	00h: Fully Redundant (formerly "Redundancy Regained") Indicates that full redundancy has been regained.
		01h	01h: Redundancy Lost Entered any non-redundant state, including Non-redundant:Insufficient Resources.
		02h	01h: Redundancy Degraded Redundancy still exists, but at a less than full level. For example, a system has four fans, and can tolerate the failure of two of them, and presently one has failed.
		03h	03h: Non-redundant:Sufficient Resources from Redundant Redundancy has been lost but unit is functioning with minimum resources needed for 'normal' operation. Entered from Redundancy Degraded or Fully Redundant.
		04h	04h: Non-redundant:Sufficient Resources from Insufficient Resources Unit has regained minimum resources needed for 'normal' operation. Entered from Non-redundant:Insufficient Resources.
		05h	05h: Non-redundant:Insufficient Resources Unit is non-redundant and has insufficient resources to maintain normal operation.
		06h	06h: Redundancy Degraded from Fully Redundant Unit has lost some redundant resource(s) but is still in a redundant state. Entered by a transition from Fully Redundant condition.
		07h	07h: Redundancy Degraded from Non-redundant Unit has regained some resource(s) and is redundant but not fully redundant. Entered from Non-redundant:Sufficient Resources or Non-redundant:Insufficient Resources.
0Ch	Discrete	ACPI Device Power States	
		00h	00h: D0 Power State
		01h	01h: D1 Power State
		02h	02h: D2 Power State
		03h	03h: D3 Power State

For sensors that have a type reading code of ‘sensor-specific’ as described by the first column in Table 38 above, use table 42-3 of the IPMI 2.0 specification. This table is duplicated as Table 37, above. Compare the Event Type for the event as described in section 6.3.5.2 to the second column of Table 37. The ‘Sensor Specific Offset’ for that sensor describes the specific state the sensor was in when the event was generated. Compare the event offset (bits 3:0) to the third column of Table 37 to determine the state the sensor is in based on the description for the specific event offset.

6.3.6.2 Event Data 2 and 3

The Intel® Management Module BMC follows the IPMI 2.0 specification for all BMC generated events.

Events generated by the BIOS are detailed in section 6.4. See that section for details on specific event data bytes.

6.4 BIOS Generated IPMI Events for the Intel® E7520 Chipset

The server board BIOS uses the Intel® Management Module BMC to log BIOS SEL events.

Table 41 details the OEM events that are initiated by the BIOS and the values that these events use to generate SEL entries. This section is specific to the system BIOS for the Intel® E7520 chipset.

For specific POST error codes, refer to the specific server board TPS, as these POST codes may be server board specific.

Table 41. BIOS Generated IPMI Events

Sensor Type	Generator ID	Sensor Type Code	Type Code	Sensor-Specific Offset	Event Data 1	Event Data 2, 3	Event
Processor	31h	07h	6Fh	03h	A3h	Data2 :port80 Data3 :port81	FRB2/POST failure
Memory	33h	0Ch	6Fh	00h	A0h	Data2: DIMM location (see note2 below) Data3: 0FFh	Correctable ECC
Memory	33h	0Ch	6Fh	01h	A1h	Data2: DIMM location (see the note below) Data3: 0FFh	Uncorrectable ECC
POST Error	31h	0Fh	6Fh	Data2[0:3]	A0h or Data2	See POST error code table7.3.3. Data2:low byte Data3:high byte	POST Error

² See the Memory Error Events table for definitions of data fields.

Sensor Type	Generator ID	Sensor Type Code	Type Code	Sensor-Specific Offset	Event Data 1	Event Data 2, 3	Event
System Event	03h	12h	6Fh	05h	05h	Data2: 00h (1st of pair) 80h (2nd of pair) Data3:0ffh	Timestamp Clock Sync
System Event	03h	12h	6Fh	01h	01h	Data2:0ffh Data3:0ffh	Boot event log
Critical Interrupt	31h	13h	6Fh	04h	A4h	Data2:bus num Data3:dev&fun	PCI SERR
Critical Interrupt	31h	13h	6Fh	05h	A5h	Data2:bus num Data3:dev&fun	PCI PERR

Table 42. Memory Error Events

Field	BIOS-Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 Format revision, Revision of the data format for OEM data bytes 2 and 3, 0 1 = ID is system software ID. As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS.
Sensor Type	0xC for memory errors
Sensor Number	See the platform SDR
Type code	0x6F
Event Data 1	Per IPMI definition. If either of the two data bytes following this do not have any data, that byte will be set to 0xff, and the appropriate filed in event data 1 should indicate that that it is unspecified. Bits 3:0 will read 0 for single bit error and 1 for multi-bit error.
Event Data 2	If this byte is specified: 7:6 Zero-based memory card number. Matches the number of type 16 entry in SMBIOS table. For example, card 0 corresponds to the first type 16 entry in SMBIOS tables. If all DIMMs are onboard, this field will always be 0. 5:0 zero-based DIMM number on the card. DIMM 0 corresponds to the first type 17 record in SMBIOS tables for that memory card.
Event Data 3	If this byte is specified, syndrome byte.

Table 43. Examples of Event Data Field Contents for Memory Errors

Error Type	Event Data 1	Event Data 2	Event Data 3
Single bit error; no information about the error is available.	00	0xFF	0xFF
Multi bit memory error, failed DIMM is the fifth DIMM on the second memory card.	0x81	0x44 (Bits 7:6 = 01 Bits 5:0 = 04)	0xFF
Single bit error. Syndrome is 0x54, DIMM location is not known.	0x20	0xFF	0x54
Multi-bit error, Syndrome byte is 0x1c. DIMMs are onboard, and the second DIMM has failed.	0xA1	0x01 (Bits 7:6 = 00 Bits 5:0 = 01)	0x1C

Table 44. PCI Error Events

Field	BIOS Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 = Format revision, Revision of the data format for OEM data bytes 2 and 3 0 1 = ID is system software ID As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS.
Sensor Type	0x13 for critical interrupt
Sensor number	See the platform SDR
Type code	0x6F
Event Data 1	Per IPMI definition. If either of the two data bytes following this do not have any data, that byte will be set to 0xff, and the appropriate field in event data 1 should indicate that that it is unspecified. Bits 3:0 will read 04 for PCI PERR and 05 for PCI SERR.
Event Data 2	If this byte is specified, it contains the PCI bus number on which the failing device resides. If the source of the PCI error cannot be determined, this byte contains 0xff and the event data 1 byte indicates that byte 2 is unspecified.
Event Data 3	If this byte is specified, it contains the PCI device/function address in the standard format: 7:3 Device number of the failing PCI device 2:0 PCI function number. Will always contain a zero if the device is not a multifunction device. If the source of the PCI error cannot be determined, this byte contains 0xff and the event data 1 byte indicates that byte 3 is unspecified.

Table 45. Examples of Event Data Field Contents for PCI Errors

Error Type	Event Data 1	Event Data 2	Event Data 3
PCI PERR, failing device is not known	04	0xFF	0xFF
PCI SERR, failing device is not known	05	0xFF	0xFF
PCI PERR, device 3, function 1 on PCI bus 5 reported the error	0xA4	0x05	0x19 (Bits 7:3 = 03 Bits 2:0 = 01)
An unknown device on PCI bus 0 reported the SERR	0x85	0x00	0xFF

Table 46. FRB2 Error Events

Field	BIOS Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 = Format revision, Revision of the data format for OEM data bytes 2 and 3 0 1 = ID is system software ID As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS.
Sensor Type	0x7 for processor related errors
Sensor number	See the platform SDR
Type code	0x6F
Event Data 1	If Event data 2 and event data 3 contain OEM codes, bits 7:6 and bits 5:4 contain 10. For platforms that do not include the POST code information with FRB-2 log, both these fields will be 0. The BIOS either should specify both bytes or should mark both bytes as unspecified. Bits 3:0 will read 03 for FRB-2 failure during POST.
Event Data 2	For format rev 0, if this byte is specified, it contains bits 7:0 of the POST code at the time FRB-2 reset occurred (port 80 code)
Event Data 3	For format rev 0, if this byte is specified, it contains bits 15:8 of the POST code at the time FRB-2 reset occurred (port 81 code). If the BIOS only uses one byte POST codes, this byte will be zero.

Table 47. Examples of Event Data Field Contents for FRB2 Errors

Error type	Event Data 1	Event Data 2	Event Data 3
FRB-2 error, failing POST code information not available	0x03	0xFF	0xFF
FRB-2 error, the BIOS uses one byte POST codes. The last POST code before FRB-2 reset was 0x60.	0xA3	0x60	0x0
FRB-2 error, the BIOS uses one byte POST codes. The last POST code before FRB-2 reset was 0x1942.	0xA3	0x42	0x19

Table 48. MCH Hub Interface Error Events

Field	BIOS-Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 = Format revision, Revision of the data format for OEM data bytes 2 and 3 0 1 = ID is system software ID As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS. 0x33 for MCH Hub Interface Error Events.
Sensor Type	0xC for memory errors
Sensor Number	See the platform SDR
Type code	0x6F
Event Data 1	Per IPMI definition. If either of the two data bytes following this do not have any data, that byte should be set to 0xff, and the appropriate field in event data 1 should indicate that that it is unspecified. Bits 3:0 will read 06 for MCH Hub Interface Error Events. 0x86 for MCH Hub Interface Error Events.
Event Data 2	Record MCH Hub Interface First or Next Error register (MCH bus0/dev0/fun1 register 50h or 52h). (See HI_FERR and HI_NERR register bit definitions)
Event Data 3	N/A (read as 0xFF)

Table 49. Examples of Event Data Field Contents for MCH Hub Interface Errors

Error Type	Event Data 1	Event Data 2	Event Data 3
HI address/command parity error detected.	0x86	0x01	0xFF
HI Internal Parity Error Detected.	0x86	0x04	0xFF

Table 50. System Bus Error Events

Field	BIOS-Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 = Format revision, Revision of the data format for OEM data bytes 2 and 3 0 1 = ID is system software ID As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS. 0x33 for System Bus Error Events.
Sensor Type	0xC for memory errors
Sensor Number	See the platform SDR
Type code	0x6F
Event Data 1	Per IPMI definition. If either of the two data bytes following this do not have any data, that byte should be set to 0xff, and the appropriate field in event data 1 should indicate that that it is unspecified. Bits 3:0 will read 07 for System Bus Error Events. 0x87 for System Bus Error Events.
Event Data 2	Record System Bus First or Next Error register (MCH bus0/dev0/fun1 register 60h or 62h). (See SYSBUS_FERR and SYSBUS_NERR bit definition in the chipset documentation.)
Event Data 3	N/A (read as 0xFF)

Table 51. Examples of Event Data Field Contents for System Bus Errors

Error Type	Event Data 1	Event Data 2	Event Data 3
System bus data strobe glitch detected	0x87	0x02	0xFF
System bus address above top of memory (TOM)	0x87	0x10	0xFF

Table 52. Memory Buffer Error Events

Field	BIOS-Specific Implementation
Generator ID	7:4 0x3 for system BIOS 3:1 0 = Format revision, Revision of the data format for OEM data bytes 2 and 3 0 1 = ID is system software ID As a result, the generator ID byte will start from 0x31 and go up to 0x3f, in increments of 2 for events logged by the BIOS. 0x33 for Memory Buffer Error Events.
Sensor Type	0xC for memory errors
Sensor Number	See the platform SDR
Type code	0x6F
Event Data 1	Follow IPMI definition. If either of the two data bytes following this do not have any data, that byte should be set to 0xff, and the appropriate field in event data 1 should indicate that that it is unspecified. Bits 3:0 will read 08 for Memory Buffer Error Events. 0x88 for System Bus Error Events.
Event Data 2	Record Memory Buffer Error First or Next register (MCH bus0/dev0/fun1 register 70h or 72h). (See BUF_FERR and BUF_NERR bit definitions in the chipset documentation.)
Event Data 3	N/A (read as 0xFF)

Table 53. Examples of Event Data Field Contents for Memory Buffer Errors

Error Type	Event Data 1	Event Data 2	Event Data 3
Internal PMWB to DRAM parity error detected.	0x88	0x01	0xFF
Internal system bus or I/O to OMWB parity error detected	0x88	0x04	0xFF

6.5 SEL Translation Examples

Included below are some examples of SEL translation.

6.5.1 Example 1

This example uses the following SEL entry.

```
00 E0 02 40 46 00 3D 20 00 04 01 98 01 57 4B 4B
```

The SEL record fields include the following:

- 00 E0 = Record ID
- 02 = Record Type
- 40 46 00 3D = Timestamp
- 20 00 = Generator ID
- 04 = EvM Rev
- 01 = Sensor Type
- 98 = Sensor Number
- 01 = Event Dir/Type
- 57 = Event Data 1
- 4B = Event Data 2
- 4B = Event Data 3

Understanding the SEL data will typically start with the generator ID, but in this example, we will start at the beginning of the SEL.

- 00 E0 = Record ID: The Record ID is the internal reference number in the BMC to indicate the unique identifying for the SEL. It has no bearing on the actual SEL translation and can be ignored.
- 02 = Record Type: Per Table 35, a record type of 02h indicates a standard system event record.
- 40 46 00 3D = Timestamp: Per section 6.2, the timestamp is calculated as the number of seconds from 00:00:00, January 1, 1970. In this example, this timestamp equals 1,078,329,405 seconds since the base date, or Wednesday, March 3rd 2004, 15:56:45 (GMT). This calculation is intended to be done programmatically, but online tools are available to convert a UNIX-style timestamp to a real date.
- 20 00 = Generator ID: Per Table 35, the first byte of this pair is the slave address of the generator of the event. In this case, 20 indicates that the generator is a device with a slave address of 20, which is the BMC as described in Table 36, and was not generated by system software. The second byte of the pair indicates that the event was received either on the primary IPMB or was internally generated by the BMC. Since the slave address matches the BMC, this indicates that the event was internally generated.
- 04 = EvM Rev: This byte indicates what SEL version to use for decoding of the SEL entry. In this case, 04h refers to the IPMI 2.0 specification.

- 01 = Sensor Type: Per Table 37, a sensor type of 01h is a Temperature Sensor.
- 98 = Sensor Number: Per the platform SDR, sensor number 98 is the processor 1 temp sensor.

Note: This sensor number for the processor 1 temp sensor may be different based on the specific platform. Always refer to the platform SDR file for current sensor number information.

- 01 = Event Dir/Type: Per Table 35, the event direction is read from bit 7 of this byte. Bit 7 is 0b, which indicates the event is being asserted. The event type is obtained from the remaining bits 6:0 and are compared to Table 38. An event type of 01h (for the remaining 7 bits) indicates a threshold based sensor.
- 57 = Event Data 1: Event data 1 must be split into two main parts for proper translation. Bits 7:4 are 5h and bits 3:0 are 7h. Convert bits 7:4 to binary and compare these bits to the first row in Table 39, which is used for threshold based sensors (as determined by the event type in the previous byte). Converting 5h to binary gives a result of 0101b. Table 39 indicates that event data 2 contains the trigger reading and event data 3 contains the trigger threshold. Bits 3:0 (7h) are compared to Table 40 (for threshold based sensors). An offset of 07h indicates that the sensor has crossed an upper non-critical threshold, and the temperature is increasing (going high).
- 4B = Event Data 2: Event data 1 indicates that this byte contains the trigger reading. The trigger reading is the reading of the sensor at the time it crossed the specified threshold from event data 1 (bits 3:0). This byte must be converted in the same way as a sensor reading obtained using a *Get Sensor Reading* command, using conversion factors from the Sensor Data Records as described in section 36.3 of the IPMI 2.0 specification. In the case of this sensor, a trigger reading of 4Bh indicates a temperature of 75C.
- 4B = Event Data 3: Event data 1 indicates that this byte contains the trigger threshold. The trigger threshold is the threshold configured for the specific sensor based on the Sensor Data Records. The specific threshold can be determined from event data 1 (bits 3:0). This byte must be converted in the same way as a sensor reading obtained using a *Get Sensor Reading* command, using conversion factors from the Sensor Data Records as described in section 36.3 of the IPMI 2.0 specification. In the case of this sensor, a trigger threshold of 4Bh indicates a temperature of 75C.

Translation: Processor temperature 1 has crossed an upper non-critical threshold (75C) and had a reading of 75C at the time the event was generated. The temperature was increasing.

7. Command Support

This chapter defines the requests (commands) that the BMC accepts, and the corresponding functionality and request/response data for these commands. These commands direct the BMC to perform specific actions. The commands are sent to the BMC via the IPMB, LPC, LAN, or EMP interfaces. For information on which interface supports which commands, see Table 89 through Table 94.

For the base specification and descriptions of BMC commands other than those specified in this chapter, see the *Intelligent Platform Management Interface Specification v2.0*. The BMC implements the event receiver, SEL, SDR, FRU, and sensor devices, as described in the *Intelligent Platform Management Interface Specification*.

7.1 Command Queuing

The BMC implements two command handlers, one for the SMM interface and the other for all other interfaces. Non-SMM sourced commands are placed on a common IPMI command queue for processing. They are executed in a first-come first-served, single threaded fashion. Commands that may take a while to execute (e.g., FRU access) will delay other, perhaps quicker to execute commands until they complete. Commands arriving on the SMM interface are executed immediately, potentially in the middle of command execution by the other command handler.

7.2 Power On/Off Issues Related to Commands

At the command interface level, the system power on/off status does not affect the validity of commands. The effectiveness of the command or the interpretation of its results, however, sometimes must be interpreted within the context of the system's power state.

For example: When the system is powered off, most scanning of sensors is disabled. Only a few sensors are actively scanned. Therefore, while any of the sensors can be read, most of the reported values are from the last sensor reading taken before system power was shut off. As stated above, any threshold or event message enable changes are lost when power is applied, but changing them with the power off could still result in event messages being generated based on the "frozen" sensor reading.

When the system power is off, all of the sensors that are enabled but not being scanned have the "sensor update in progress" bit set in the response to *Get Sensor Reading*, *Get Sensor Event Status*, or *Get Sensor Event Enable* IPMI commands.

7.3 BMC Command Tables

The tables on the following pages present the commands, requests, and responses that the BMC accepts via the IPMB, LPC, LAN, EMP and PCI SMBus interfaces. Responses always go out on the interface on which the associated request was received.

Table 54. BMC Intelligent Platform Management Interface (IPMI) Commands

Net Function = Application, LUN = 00			
Code	Command	Request, Response Data	Description
01h	Get Device ID	Per the <i>IPMI 2.0</i> specification	Platform-specific response fields: Byte 2 (device ID) – 20h Byte 3 (device revision) Byte 4 () – Firmware Revision 1 Byte 5 () – Firmware Revision 2 Byte 6 (IPMI version) – 02h Byte 7 (Additional Device Support) bit 7 – Chassis device bit 6 – Bridge bit 5 – IPMB Event Generator bit 4 – IPMB Event Receiver bit 3 – FRU Inventory Device bit 2 – SEL Device bit 1 – SDR Repository Device bit 0 – Sensor Device Bytes 8:10 (manufacturer ID) – 343 (57h, 01h, 00h) Bytes 11:12 (product ID) Bytes 13:16 (Optional Auxiliary Firmware Revision Information) Byte 13 – Boot firmware revision, Major (binary) Byte 14 – Boot firmware revision, Minor (BCD) Byte 15 – PIA format revision, Major Byte 16 – PIA instance version, Minor
01h	Broadcast get device ID	Per the <i>IPMI 2.0</i> specification	
02h	Cold reset	Not implemented	
03h	Warm reset	Not implemented	
04h	Get Self Test results	Per the <i>IPMI 2.0</i> specification	See Section 4.23 for more information.
05h	Manufacturing Test On	Not implemented	
06h	Set ACPI Power State	Per the <i>IPMI 2.0</i> specification	
07h	Get ACPI Power State	Per the <i>IPMI 2.0</i> specification	
08h	Get Device GUID	Not implemented	

Table 55. BMC Watchdog Timer Commands

Net Function = Application, LUN = 00			
Code	Command	Request, Response Data	Description
22h	Reset watchdog timer	Per the <i>IPMI 2.0</i> specification	
24h	Set watchdog timer	Per the <i>IPMI 2.0</i> specification	The only supported interrupts are SMI (IA-32 platforms only) and NMI / Diagnostic Interrupt.
25h	Get watchdog timer	Per the <i>IPMI 2.0</i> specification	

Table 56. BMC IPMI Messaging Support Commands

Net Function = Application, LUN = 00			
Code	Command	Request, Response Data	Description
2Eh	Set BMC global enables	Per the <i>IPMI 2.0</i> specification	The OEM 2 Enable bit enables the SMI interrupt for Event Message Buffer full. Supported on IA-32 platforms only. Note: The Event Message Buffer is shared between the SMS and SMM interfaces. This bit enables the interrupt for the SMM interface. Bit 1 "Enable Event Message Buffer Full Interrupt" enables the interrupt for the SMS interface. The OEM 1 Enable bit is not supported. The OEM 0 Enable bit is not supported.
2Fh	Get BMC global enables	Per the <i>IPMI 2.0</i> specification	The OEM enable bits are supported as in the <i>Set BMC Global Enables</i> command above.
30h	Clear message flags	Per the <i>IPMI 2.0</i> specification	The OEM 0 flag is not supported. The OEM 1 flag is used to indicate Diagnostic Interrupt (Front Panel NMI) button press. The OEM 2 flag is not supported.
31h	Get message flags	Per the <i>IPMI 2.0</i> specification	The OEM message flags are supported as in the <i>Clear Message Flags</i> command above
32h	Enable message channel receive	Not implemented	Message channels are always enabled.
33h	Get message	Per the <i>IPMI 2.0</i> specification	See section 4.27.1 for supported channel IDs
34h	Send message	Per the <i>IPMI 2.0</i> specification	See section 4.27.1 for supported channel IDs
35h	Read event message buffer	Per the <i>IPMI 2.0</i> specification	
36h	Get BT Interface Capabilities	Not implemented	
37h	Get System GUID	Per the <i>IPMI 2.0</i> specification	

Net Function = Application, LUN = 00			
Code	Command	Request, Response Data	Description
38h	Get Channel Authentication Capabilities	Per the <i>IPMI 2.0</i> specification	The following authentication types are supported over the Serial/Modem and LAN interfaces: Straight Password/Key MD5 MD2
39h	Get Session Challenge	Per the <i>IPMI 2.0</i> specification	This command is only accepted over the LAN and EMP/PPP transports.
3Ah	Activate Session	Per the <i>IPMI 2.0</i> specification	This command is only accepted over the LAN and EMP/Basic, EMP/PPP transports.
3Bh	Set Session Privilege Level	Per the <i>IPMI 2.0</i> specification	This command is only accepted over the LAN and PPP transports.
3Ch	Close Session	Per the <i>IPMI 2.0</i> specification	
3Dh	Get Session Info	Per the <i>IPMI 2.0</i> specification	
3Fh	Get AuthCode	Per the <i>IPMI 2.0</i> specification	This command is only accepted over the host interface.
40h	Set Channel Access	Per the <i>IPMI 2.0</i> specification	
41h	Get Channel Access	Per the <i>IPMI 2.0</i> specification	
42h	Get Channel Info	Per the <i>IPMI 2.0</i> specification	
43h	Set User Access	Per the <i>IPMI 2.0</i> specification	
44h	Get User Access	Per the <i>IPMI 2.0</i> specification	
45h	Set User Name	Per the <i>IPMI 2.0</i> specification	
46h	Get User Name	Per the <i>IPMI 2.0</i> specification	
47h	Set User Password	Per the <i>IPMI 2.0</i> specification	
48h	Activate Payload	Per the <i>IPMI 2.0</i> specification	
49h	Deactivate Payload	Per the <i>IPMI 2.0</i> specification	
4Ah	Get Payload Activation Status	Per the <i>IPMI 2.0</i> specification	
4Bh	Get Payload Instance Info	Per the <i>IPMI 2.0</i> specification	
4Ch	Set User Payload Access	Per the <i>IPMI 2.0</i> specification	
4Dh	Get User Payload Access	Per the <i>IPMI 2.0</i> specification	
4Eh	Get Channel Payload Support	Per the <i>IPMI 2.0</i> specification	

Net Function = Application, LUN = 00			
Code	Command	Request, Response Data	Description
4Fh	Get Channel Payload Version	Per the <i>IPMI 2.0</i> specification	
50h	Get Channel OEM Payload Info	Per the <i>IPMI 2.0</i> specification	
52h	Master write-read I2C	Per the <i>IPMI 2.0</i> specification	The supported buses are public bus 0 (IPMB).
54h	Get Channel Cipher Suites	Per the <i>IPMI 2.0</i> specification	
55h	Suspend/Resume Payload Encryption	Per the <i>IPMI 2.0</i> specification	
56h	Set Channel Security Keys	Per the <i>IPMI 2.0</i> specification	

Table 57. BMC Chassis Commands

Net Function = Chassis, LUN = 00			
Code	Command	Request, Response Data	Description
00h	Get Chassis Capabilities	Per the <i>IPMI 2.0</i> specification Request: N/A Response: Byte 1 Completion code Byte 2 Capabilities flags 7:4: reserved 3: provides Power Interlock status 2: provides Diagnostic Interrupt (FP NMI) control 1: provides Secure Mode 0: provides Intrusion status Byte 3 Chassis FRU info IPMB address Byte 4 Chassis SDR IPMB address Byte 5 Chassis SEL IPMB address Byte 6 Chassis SM IPMB address Byte 7 Chassis Bridge IPMB address	
01h	Get Chassis Status	Per the <i>IPMI 2.0</i> specification	

Net Function = Chassis, LUN = 00			
Code	Command	Request, Response Data	Description
02h	Chassis Control	Per the <i>IPMI 2.0</i> specification	Supported control actions are: 0h Power Down, forced S4/S5 1h Power Up 2h Power Cycle, forced/immediate 3h Hard Reset, forced/immediate 4h Pulse Diag Int 5h Soft shutdown, uses OS agent based solution
04h	Chassis Identify	Per the <i>IPMI 2.0</i> specification	The BMC implements the optional duration parameter.
06h	Set Power Restore Policy	Per the <i>IPMI 2.0</i> specification	
0Fh	Get POH Counter	Per the <i>IPMI 2.0</i> specification	

Table 58. Chassis Control Command Results

Command	Current Power State	Result
Power down	Up	Powers the system down
	Down	No change to system power state
Power up	Up	No change to system power state
	Down	Powers the system up
Power cycle	Up	System power goes down , then back up
	Down	No change to system power state
Hard reset	Up	BMC performs system reset
	Down	No action
Pulse Diagnostic Interrupt	Up	BMC performs the Diagnostic Interrupt operation
	Down	No action

Table 59. Boot Control Commands

Net Function = Chassis, LUN = 00		
Code	Command	Request, Response Data
07h	Get System Restart Cause	Per the <i>IPMI 2.0</i> specification
08h	Set System Boot Options	Per the <i>IPMI 2.0</i> specification
09h	Get System Boot Options	Per the <i>IPMI 2.0</i> specification

Table 60. Supported Boot Option Parameters

Parameter	#	Description
Set In Progress	0	Per the IPMI 2.0 specification
Service partition selector	1	Per the IPMI 2.0 specification
Service partition scan	2	Per the IPMI 2.0 specification
BMC boot flag clearing	3	Per the IPMI 2.0 specification
Boot info acknowledge	4	Per the IPMI 2.0 specification
Boot flags	5	Per the IPMI 2.0 specification
Boot initiator info	6	Per the IPMI 2.0 specification
Boot initiator mailbox	7	Per the IPMI 2.0 specification

Table 61. BMC Event Receiver Device Commands

Net Function = Sensor/Event, LUN = 00			
Code	Command	Request, Response Data	Description
00h	Set Event Receiver	Not implemented	BMC does not function as IPMB event generator.
01h	Get Event Receiver	Per the <i>IPMI 2.0</i> specification	Returns the BMC's IPMB address and LUN.
02h	Platform Event Message	Per the <i>IPMI 2.0</i> specification	

Table 62. PEF Commands

Net Function = Sensor/Event, LUN = 00		
Code	Command	Request, Response Data
10h	Get PEF Capabilities	Per the IPMI 2.0 specification
11h	Arm PEF Postpone Timer	Per the IPMI 2.0 specification
12h	Set PEF Configuration Parameters	Per the IPMI 2.0 specification
13h	Get PEF Configuration Parameters	Per the IPMI 2.0 specification
14h	Set Last Processed Event ID	Per the IPMI 2.0 specification
15h	Get Last Processed Event ID	Per the IPMI 2.0 specification
16h	Alert Immediate	Per the IPMI 2.0 specification
17h	PET Acknowledge	Per the IPMI 2.0 specification

Table 63. Supported PEF Configuration Parameters

Parameter	#	Description
Set in progress	0	Per the IPMI 2.0 specification
PEF control – non-volatile	1	Per the IPMI 2.0 specification PEF startup delay disable (bit 2) is not supported.
PEF Action global control – non-volatile	2	Per the IPMI 2.0 specification
PEF Startup Delay – non-volatile	3	Per the IPMI 2.0 specification
PEF Alert Startup Delay – non-volatile	4	Per the IPMI 2.0 specification
Number of Event Filters	5	Per the IPMI 2.0 specification
Event Filter Table – non-volatile	6	Per the IPMI 2.0 specification
Event Filter Table Data 1 – non-volatile	7	Per the IPMI 2.0 specification
Number of Alert Policy Entries	8	Per the IPMI 2.0 specification
Alert Policy Table	9	Per the IPMI 2.0 specification
System GUID	10	Per the IPMI 2.0 specification
Number of Alert Strings	11	Per the IPMI 2.0 specification
Alert String Keys	12	Per the IPMI 2.0 specification
Alert Strings	13	Per the IPMI 2.0 specification
OEM Alert String	96	This parameter is a 28-byte field formatted according to the IPMI 1.5 OEM Custom Fields and Text Alert Strings for IPMI 1.5 PET, section 15.16.1, Table 15-8 in the IPMI 2.0 specification. This parameter is only accepted over LAN, EMP/Basic or KCS transports.
Alert Retry Algorithm	97	0 IPMI 1.5 (default) 1 reserved
UTC Offset	98	This parameter has the following format Byte 1 [7:1] Reserved [0] Valid Bytes 2:3 UTC Offset (LSB first)
PET Trap Format Options	99	Data 1 Set Selector [7:2] Reserved [1] Select SNMP Manager compatible PET trap [0] 1= Append SDR name to trap The larger, SNMP-manager compatible format splits the single variable binding fields into many separate variable bindings fields (i.e OID/value pairs) so as to provide visibility into each element of the trap, rather than one large hexadecimal string.
OEM Event Filter Table, (non-volatile)	100	Data 1 Set Selector = filter number [7] Reserved [6:0] Filter number. 1-based. 00h = reserved Data 2:3 Filter data

Table 64. BMC Sensor Device Commands

Net Function = Sensor/Event, LUN = 00			
Code	Command	Request, Response Data	Description
20h	Get device SDR info	Not implemented	
21h	Get device SDR	Not implemented	
22h	Reserve device SDR repository	Not implemented	
23h	Get sensor reading factors	Not implemented	
24h	Set sensor hysteresis	Per <i>IPMI 2.0</i> specification	
25h	Get sensor hysteresis	Per <i>IPMI 2.0</i> specification	
26h	Set sensor threshold	Per <i>IPMI 2.0</i> specification	
27h	Get sensor threshold	Per <i>IPMI 2.0</i> specification	The readable threshold mask returned by the BMC for the <i>Get Sensor Thresholds</i> command only contains set bits for thresholds that have been given values. Only those thresholds are actively processed by the BMC. Thresholds that are marked as Readable in the SDR but haven't been set for that sensor have no valid value.
28h	Set sensor event enable	Per <i>IPMI 2.0</i> specification	
29h	Get sensor event enable	Per <i>IPMI 2.0</i> specification	Most enabled but not scanned sensors will indicate 'sensor update in progress' when system power is off. See section 7.2.
2Ah	Re-arm sensor events	Per <i>IPMI 2.0</i> specification	
2Bh	Get sensor event status	Per <i>IPMI 2.0</i> specification	Most enabled but not scanned sensors will indicate 'sensor update in progress' when system power is off. See section 7.2.
2Dh	Get sensor reading	Per <i>IPMI 2.0</i> specification	Most sensors will indicate 'sensor update in progress' when system power is off. See section 7.2.
2Eh	Set Sensor Type	Not implemented	
2Fh	Get Sensor Type	Not implemented	

Table 65. BMC FRU Inventory Device Commands

Net Function = Storage, LUN = 00		
Code	Command	Request, Response Data
10h	Get FRU inventory area info	Per the IPMI 2.0 specification. The FRU device is accessed by bytes.
11h	Read FRU inventory data	Per the IPMI 2.0 specification. The FRU device is accessed by bytes.
12h	Write FRU inventory data	Per the IPMI 2.0 specification. The FRU device is accessed by bytes.

Table 66. BMC SDR Repository Device Commands

Net Function = Storage, LUN = 00		
Code	Command	Request, Response Data
20h	Get SDR repository info	Per the IPMI 2.0 specification
21h	Get SDR repository allocation info	Per the IPMI 2.0 specification
22h	Reserve SDR repository	Per the IPMI 2.0 specification
23h	Get SDR	Per the IPMI 2.0 specification
24h	Add SDR	Not implemented (Per IPMI 2.0, this command is optional if partial-add SDR is implemented.)
25h	Partial add SDR	Per the <i>IPMI 2.0</i> specification
26h	Delete SDR	Per the <i>IPMI 2.0</i> specification
27h	Clear SDR repository	Per the <i>IPMI 2.0</i> specification
28h	Get SDR repository time	Per the <i>IPMI 2.0</i> specification
29h	Set SDR repository time	Not implemented
2Ah	Enter SDR repository update mode	Not implemented
2Bh	Exit SDR repository mode	Not implemented
2Ch	Run Initialization Agent	Per the <i>IPMI 2.0</i> specification

Table 67. BMC SEL Device Commands

Net Function = Storage, LUN = 00		
Code	Command	Request, Response Data
40h	Get SEL info	Per the IPMI 2.0 specification
41h	Get SEL allocation info	Per the IPMI 2.0 specification
42h	Reserve SEL	Per the IPMI 2.0 specification
43h	Get SEL entry	Per the IPMI 2.0 specification
44h	Add SEL entry	Per the IPMI 2.0 specification
45h	Partial Add SEL entry	Per the IPMI 2.0 specification
46h	Delete SEL entry	Per the IPMI 2.0 specification
47h	Clear SEL	Per the IPMI 2.0 specification
48h	Get SEL time	Per the IPMI 2.0 specification
49h	Set SEL time	Per the IPMI 2.0 specification
5Ah	Get Auxiliary Log Status	Not implimented
5Bh	Set Auxiliary Log Status	Not implimented

Table 68. LAN Commands

Net Function = Transport, LUN = 00		
Code	Command	Request, Response Data
01h	Set LAN Configuration	Per the IPMI 2.0 specification
02h	Get LAN Configuration	Per the IPMI 2.0 specification
03h	Suspend BMC ARPs	Per the IPMI 2.0 specification
04h	Get IP/UDP/RMCP Statistics	Per the IPMI 2.0 specification

Table 69. Supported LAN Configuration Parameters

Parameter	#	Description
Set In Progress	0	Per the IPMI 2.0 specification
Authentication Type Support	1	Per the IPMI 2.0 specification
Authentication Type Enables	2	Per the IPMI 2.0 specification
IP Address	3	Per the IPMI 2.0 specification
IP Address Source	4	Per the IPMI 2.0 specification
MAC Address	5	Per the IPMI 2.0 specification
Subnet Mask	6	Per the IPMI 2.0 specification
Ipv4 Header Parameters	7	Per the IPMI 2.0 specification
Primary RMCP Port Number (optional)	8	Per the IPMI 2.0 specification
Secondary RMCP Port Number (optional)	9	Per the IPMI 2.0 specification
Gratuitous ARP control	10	Per the IPMI 2.0 specification
Gratuitous ARP interval	11	Per the IPMI 2.0 specification
Default Gateway Address	12	Per the IPMI 2.0 specification
Default Gateway MAC Address	13	Per the IPMI 2.0 specification
Backup Gateway Address	14	Per the IPMI 2.0 specification
Backup Gateway MAC Address	15	Per the IPMI 2.0 specification
Community String	16	Per the IPMI 2.0 specification
Number of Destinations (Read Only)	17	
Destination Type	18	Per the IPMI 2.0 specification Additionally, destination type OEM1 (110b) is mapped to SMTP Alerting (if supported – see parameter 195 Feature Configuration Support)
Destination Addresses	19	Per the IPMI 2.0 specification
802.1q VLAN ID (12-bit)	20	Not implemented
802.1q VLAN Priority	21	Not implemented
RCMP+ Messaging Cipher Suite Entry Support (Read Only)	22	Per the IPMI 2.0 specification
RMCP+ Messaging Cipher Suite Entries	23	Per the IPMI 2.0 specification

Parameter	#	Description
RMCP+ Messaging Cipher Suite Privilege Levels	24	Per the IPMI 2.0 specification
Support (read-only)	195 (OEM)	<p>This parameter provides the set of OEM alerting and messaging features that are supported over this channel.</p> <p>Data 1 Telnet Support [7:1] Reserved [0] 1 = Telnet supported, 0 = Telnet not supported</p> <p>Data 2 HTTP Support [7:1] Reserved [0] 1 = HTTP supported, 0 = HTTP not supported</p> <p>Data 3 HTTPS Support [7:1] Reserved [0] 1 = HTTPS supported, 0 = HTTPS not supported</p> <p>Data 4 SNMP Support [7:1] Reserved [0] 1 = SNMP supported, 0 = SNMP not supported</p> <p>Data 5 SMTP Alerting Support [7:1] Reserved [0] 1 = SMTP Alerting supported, 0 = SMTP Alerting not supported</p> <p>Data 6 DHCP/ARP Support [7:2] Reserved [1] 1 = ARP supported, 0 = ARP not supported [0] 1 = DHCP supported, 0 = DHCP not supported</p>
SMTP Alert Configuration	196 (OEM)	<p>This parameter maps LAN Alert Destinations to SMTP Alert Configurations (as configured via the <i>Get/Set</i> SMTP Alert Configuration Parameter commands).</p> <p>Data 1 Set Selector = Destination Selector [7:4] Reserved [3:0] Destination Selector, same selector as for IPMI LAN parameter 18.</p> <p>Data 2 SMTP Alert Configuration index This value maps the selected Alert Destination to an SMTP Alert configuration. Any LAN alert destination, including the volatile destination 0, can be mapped to one of the four SMTP configurations. This value is used as the selector in SMTP Alert Configuration parameters.</p>
Primary DNS server IP address	197 (OEM)	The DNS server IP addresses are only used by the advanced feature set when DHCP is disabled. If DHCP is enabled the DNS server IP addresses will be set via DHCP.
Secondary DNS server IP address	198 (OEM)	The DNS server IP addresses are only used by the advanced feature set when DHCP is disabled. If DHCP is enabled the DNS server IP addresses will be set via DHCP.

Table 70. EMP Commands

Net Function = Transport, LUN = 00		
Code	Command	Request, Response Data
10h	Set Serial/Modem Configuration	Per the IPMI 2.0 specification
11h	Get Serial/Modem Configuration	Per the IPMI 2.0 specification
12h	Set Serial/Modem Mux	Per the IPMI 2.0 specification
14h	Set PPP UDP Proxy Transmit Data	Per the IPMI 2.0 specification
15h	Get PPP UDP Proxy Transmit Data	Per the IPMI 2.0 specification
16h	Send PPP UDP Proxy Packet	Per the IPMI 2.0 specification
17h	Get PPP UDP Proxy Receive Data	Per the IPMI 2.0 specification
18h	Serial/Modem Connection Active	Per the IPMI 2.0 specification
19h	Callback	Per the IPMI 2.0 specification
1Ah	Set User Callback Options	Per the IPMI 2.0 specification
1Bh	Get User Callback Options	Per the IPMI 2.0 specification

Table 71. Supported Serial/Modem Configuration Parameters

Note: The default parameter values are as specified in the IPMI 2.0 specification if they are not specified below. Platform specific BMC EPS definition of defaults and string lengths override those specified here.

Parameter	#	Description
Set in progress	0	Per the IPMI 2.0 specification
Authentication Type Support	1	
Authentication Type Enables	2	
Connection Mode	3	Per the IPMI 2.0 specification Terminal Mode, PPP Mode, and Basic Mode supported
Session Inactivity Timeout	4	
Channel Callback Control	5	
Session Termination	6	Per the IPMI 2.0 specification
IPMI Messaging Comm Setting	7	Per the IPMI 2.0 specification
Mux switch Control	8	Per the IPMI 2.0 specification
Modem Ring Time	9	Per the IPMI 2.0 specification
Modem Init String	10	Per the IPMI 2.0 specification Size: 48 bytes maximum
Modem Escape Sequence	11	Per the IPMI 2.0 specification Size: 5 bytes maximum
Modem Hangup Sequence	12	Per the IPMI 2.0 specification Size: 8 bytes maximum
Modem Dial Command	13	Per the IPMI 2.0 specification
Page Blackout Interval	14	Per the IPMI 2.0 specification

Parameter	#	Description
Community String	15	Per the IPMI 2.0 specification Size: 18 bytes maximum
Number of Alert Destinations (Read Only)	16	Per the IPMI 2.0 specification
Destination Info (volatile and non-volatile)	17	Per the IPMI 2.0 specification
Call Retry Interval	18	Per the IPMI 2.0 specification
Destination Comm Settings (volatile and non-volatile)	19	Per the IPMI 2.0 specification
Number of Dial Strings (Read Only)	20	Per the IPMI 2.0 specification
Destination Dial Strings (volatile and non-volatile)	21	Per the IPMI 2.0 specification Size per string: 32 bytes maximum
Number of Alert Destination IP Addresses	22	Per the IPMI 2.0 specification
Destination IP Addresses	23	Per the IPMI 2.0 specification
Terminal Mode Configuration	29	Per the IPMI 2.0 specification Defaults: Line Editing enabled Delete Control Sequence <BKSP><SP><BKSP> Echo enabled Handshake enabled
PPP Protocol Options	30	Per the IPMI 2.0 specification
PPP Primary RMCP Port Number	31	Per the IPMI 2.0 specification
PPP Secondary RMCP Port Number (optional)	32	Per the IPMI 2.0 specification
PPP Link Authentication	33	Per the IPMI 2.0 specification
CHAP Name	34	Per the IPMI 2.0 specification
PPP ACCM (optional)	35	Per the IPMI 2.0 specification
PPP Snoop ACCM	36	Per the IPMI 2.0 specification
Number of PPP Accounts	37	Per the IPMI 2.0 specification
PPP Account Dial String Selector	38	Per the IPMI 2.0 specification
PPP Account IP Addresses	39	Per the IPMI 2.0 specification
PPP Account User Names	40	Per the IPMI 2.0 specification
PPP Account User Domains	41	Per the IPMI 2.0 specification
PPP Account User Passwords	42	Per the IPMI 2.0 specification
PPP Account Authentication Settings	43	Per the IPMI 2.0 specification
PPP Account Connection Hold Times	44	Per the IPMI 2.0 specification
PPP UDP Proxy IP Header data	45	Per the IPMI 2.0 specification
PPP UDP Proxy Transmit Buffer Size (Read Only)	46	Per the IPMI 2.0 specification
PPP UDP Proxy Receive Buffer Size (Read Only)	47	Per the IPMI 2.0 specification
PPP Remote Console IP Address	48	Per the IPMI 2.0 specification

Parameter	#	Description
System Phone Number	192	This parameter stores a 32 byte string holding (by convention) the system phone number. The BMC does not interpret or otherwise use this information. Data 1 – block selector, 16 byte block to set, 1 based (2 blocks supported) Data 2-17 – data for selected block
CLI Syntax	193	Command Line Interface Syntax selection [7:1] reserved [0] 0b = TMCLI (Terminal Mode Text Commands) (default) 1b = NCLI (Native CLI Text Commands)

Table 72. Terminal Mode Commands

Command	Request, Response Data	Privilege	Description
SYS PWD	As defined by the IPMI 2.0 specification	Callback	All IPMI defined variations of this command supported
SYS TMODE	As defined by the IPMI 2.0 specification	Callback	
SYS SET BOOT	As defined by the IPMI 2.0 specification	Admin	
SYS SET BOOTOPT	As defined by the IPMI 2.0 specification	Admin	
SYS GET BOOTOPT	As defined by the IPMI 2.0 specification	Operator	
SYS SET TCFG	As defined by the IPMI 2.0 specification	Admin	All IPMI defined variations of this command supported
SYS RESET	As defined by the IPMI 2.0 specification	Operator	
SYS POWER ON	As defined by the IPMI 2.0 specification	Operator	
SYS POWER OFF	As defined by the IPMI 2.0 specification	Operator	
SYS HEALTH QUERY	As defined by the IPMI 2.0 specification	User	All IPMI defined variations of this command supported
XX XX ...	Hex ASCII encoded IPMI commands – Request/Response Data as defined in other command tables.	Varies	Privilege as defined for specific encoded IPMI command.

Table 73. Intel OEM Terminal Mode Commands

Command	Request, Response Data	Privilege	Description
SYS 000157 ACTIVATE		User	See Table 86
SYS BOOT		Operator	See Table 86
SYS DIAGINT		Operator	See Table 86
SYS HELP		Callback	See Table 86
SYS ID		Callback	See Table 86
SYS IDENTIFY		Operator	See Table 86
SYS NETWORK		Operator	See Table 86
SYS SHUTDOWN		Operator	See Table 86

Table 74. Intel General Application Commands

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
16h	Read Self Test	Request: Byte 1 Action 0 Get First 1 Get Next Response: Byte 1 Completion code Byte 2 First byte of test result FFh No more results Byte 3 Second byte of test result	This command allows all stored self-test results to be retrieved. Normal use is to first call with action of 'Get First' then use 'Get Next' for subsequent calls. If byte 2 of the response is ever 0xFF, all the results have been retrieved. Unlike <i>Get Self Test Results</i> command, this command will never return code 55h (No Error) for the first response byte.
20h	Get Power Supply State	Request: N/A Response: Byte 1 Completion code Byte 2 Number of power supplies Byte 3 Power supply 0 state 00h On 01h Power supply is operational and off 10h Power supply is malfunctioning and commanded on 20h Power supply is not present Byte 4 Power supply 1 state Byte 5 Power supply 2 state	This command returns the current state of the power subsystem. Byte 2 indicates how many supplies are monitored by the BMC. Only the power supply state bytes up to that number are valid.
21h	Get DIMM State	Request: N/A Byte 1 DIMM Group Id Response: Byte 1 Completion code Byte 2 DIMM Group Presence [7:1]Reserved [0] Presence (1 = group present) Byte 3 Bitmap of DIMM slot existence Byte 4 Bitmap of DIMM failure state Byte 5 Bitmap of DIMM disabled state Byte 6 Bitmap of DIMM sparing state Byte 7 Bitmap of DIMM presence state	This command allows the <i>presence</i> , <i>disabled</i> , <i>failure</i> , and <i>spared</i> state of a set of DIMMs to be obtained without having to know the BMC IPMI sensor numbers of the associated DIMM sensors. The state is returned as bitmaps. A bit being set in a bitmap indicates assertion of the state. The DIMM that a bit offset refers to is platform dependent. The DIMM Group Selector value is platform dependent. Platforms with 8 or fewer DIMMs should always use 0 as the Group Selector value. On platforms with memory boards, the Group Selector maps to board number, which is 1 based.
23h	ReArm DIMMs	Request: N/A Response: Byte 1 Completion code	This command causes the DIMM failure and disabled state of all DIMM sensors to be reset.
29h	Get Processor State	Request: Byte 1 processor ID Valid values are 0:N-1 where N is the number of processors supported by the platform Response: Byte 1 Completion code	This command returns the current Processor Status sensor event assertion status for the requested processor. This command allows the caller to get processor state without having to know the IPMI sensor numbers of the Processor Status sensors.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		Byte 2:3 Processor state These bytes contain the associated Processor Status sensor's 2 byte event assertion status as defined in the IPMI 1.5 specification.	
2Ah	ReArm Processors	Request: N/A Response: Byte 1 Completion code	This command clears all error and disabled state for all processors. Processor/terminator presence is not affected. Disabled processors are not actually run until the next system reset.
36h	Chassis Identify LED	Request: Byte 1 7:3 Reserved 2:0 LED State 00h No change (used to read current LED State) 01h LED Off 02h LED Blink 03h-07h Reserved Response: Byte 1 Completion code Byte 2 7:3 Reserved 2:0 Present LED State 01h LED Off 02h LED Blink 03h LED On 04h-07h Reserved	This command changes the chassis ID LED to the new state specified in the LED state parameter. See section 4.7.1 for more information.
57h	Set Fault Indication	Request: Byte 1 Source Supported values: 0 – 3 Byte 2 Fault Type. Supported fault types: Fan = 0 Temp = 1 Power = 2 Drive Slot = 3 Software = 4 Byte 3 State to set. Supported states: OK = 0 Degraded = 1 NonCritical = 2 Critical = 3 NonRecoverable = 4 Response: Byte 1 Completion code	Satellite controllers on the IPMB and/or software can indicate to the BMC that they have detected a fault assertion or deassertion using this command. Source values should be specified in the platform specific BMC EPS. The BMC generates a composite fault state from this information, as well as its own internal fault state, and uses this to drive front panel indicators.
64h	Get Platform Information	Request: Byte 1 Parameter selector 7 Reserved	This command is used to retrieve various platform specific information, per Table 75. This command can be executed before a session being activated.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		<p>6:0 Parameter selector</p> <p>Byte 2 Set selector; Selects a particular block or set of parameters under the given parameter selector.</p> <p>Write as 00h if parameter does not use a Set Selector</p> <p>Byte 3 Block selector; Selects a particular block <i>within a set</i> of parameters.</p> <p>Write as 00h if parameter does not use a Block Selector.</p> <p>Response:</p> <p>Byte 1 Completion code</p> <p>Byte 2 Parameter version</p> <p>7:4 Reserved</p> <p>3:0 Parameter version. 1h for this specification unless otherwise specified.</p> <p>Byte 3 Parameter valid. This indicates that the selected parameter has been marked invalid or locked. Currently this only applies to the "Service Partition Presence" parameter, which can be marked invalid by the "Set System Boot Options" IPMI command.</p> <p>7</p> <p>1b = parameter marked invalid / locked 0b = parameter marked valid / unlocked</p> <p>6:0 Parameter selector</p> <p>Byte 4:N</p> <p>Platform information parameter data, per Table 75.</p>	
70h	Graceful OS Shutdown	<p>Request:</p> <p>Byte 1</p> <p>7:3 Reserved</p> <p>2:0 Shutdown Operation</p> <p>00h No change (used to read shutdown command status)</p> <p>01h Power off using the OS Agent</p> <p>02h Reset using the OS Agent</p> <p>03h-07h Reserved</p> <p>Response:</p> <p>Byte 1 Completion code</p> <p>Byte 2 (only for command status)</p> <p>7:2 Reserved</p> <p>1:0 Shutdown command status</p> <p>00h Operation Successful</p> <p>01h Operation Failed (no OS agent)</p> <p>02h Operation in progress</p> <p>03h Reserved</p>	This command performs graceful shutdown using OS agent. See section 4.28.2 for more information.
8Ch	Get Memory RAS	Request: N/A	This command returns the Memory RAS domain configuration. See the description of

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
	Configuration	Response: Byte 1 Completion code Byte 2 Sparing Domain Enable Mask [7:0] Bit set indicates associated domain enabled Byte 3 Mirroring Domain Enable Mask [7:0] Bit set indicates associated domain enabled Byte 4 RAID Domain Enable Mask [7:0] Bit set indicates associated domain enabled	Set Memory RAS Configuration command and section 4.19 for more information.
90h	Get Chassis Name	Request: N/A Response: Byte 1 Completion code Byte 2 Length of chassis name = 10h Bytes 3:18 Chassis name	This command retrieves a chassis name from nonvolatile storage. The chassis name can be 0-16 characters. This implementation uses 16-byte name. The Intel® Local Control Panel uses the Chassis Name field to display the server name.
91h	Set Chassis Name	Request: Byte 1 Number of characters to store Bytes 2 to N Chassis name Response: Byte 1 Completion code	This command stores a chassis name to nonvolatile storage. The chassis name can be 0-16 non-null characters. The Intel® Local Control Panel uses the Chassis Name field to display the server name.
B0h	Get User Feature Configuration Parameters	Request: Byte 1 Flags 7 Get type 0b = Get parameter 1b = Get parameter revision only 6:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Byte 2 Parameter revision Format: MSN = present revision LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification. Bytes 3-N Parameter data	This command gets the per user feature enable state, indicating whether a particular user is enabled for a feature This is controlled on a per channel basis. See Table 83 for the definition of the User Feature Parameters. Note: Parameters that require a set and/or block selector will require additional request bytes.
B1h	Set User Feature Configuration Parameters	Request: Byte 1 Flags 7:4 reserved 3:0 Channel Number	This command sets the per user feature enable state, allowing use of a feature to be controlled on a per-user basis. This is controlled on a per channel basis. See Table 83 for the definition of the User Feature

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		Byte 2 Parameter selector Bytes 3-N Parameter data Response: Byte 1 Completion code	Parameters.
B2h	Get SMTP Configuration Parameters	Request: Byte 1 Flags 7 Get type 0b = get parameter 1b = get parameter revision only 6:0 Reserved Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Byte 2 Parameter revision Format: MSN = present revision. LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification. Bytes 3-N Parameter data	See Table 77 for the definition of the supported parameters.
B3h	Set SMTP Configuration Parameters	Request: Byte 1 Parameter selector Bytes 2-N Parameter data Response: Byte 1 Completion code	See Table 77 for the definition of the supported parameters.
B4h	Get HTTP Configuration Parameters	Request: Byte 1 Flags 7 Get type 0b = get parameter 1b = get parameter revision only 6:0 reserved Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Byte 2 Parameter revision Format:	See Table 78 for the definition of the supported parameters.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		<p>MSN = present revision. LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification. Bytes 3-N Parameter data</p>	
B5h	Set HTTP Configuration Parameters	<p>Request: Byte 1 Parameter selector Bytes 2-N Parameter data Response: Byte 1 Completion code</p>	See Table 78 for the definition of the supported parameters.
B6h	Get HTTP Channel Configuration Parameters	<p>Request: Byte 1 Flags 7 Get type 0b = get parameter 1b = get parameter revision only 6:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Generic codes, plus the following command-specific code: 80h = Parameter not supported Byte 2 Parameter revision Format: MSN = present revision LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification. Bytes 3-N Parameter data</p>	See Table 79 for the definition of the supported parameters
B7h	Set HTTP Channel Configuration Parameters	<p>Request: Byte 1 Flags 7:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Bytes 3-N Parameter data Response: Byte 1 Completion code</p>	See table Table 79 for the definition of the supported parameters
B8h	Get Telnet Configuration Parameters	<p>Request: Byte 1 Flags 7 Get type</p>	See Table 81 for the definition of the supported parameters.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		0b = get parameter 1b = get parameter revision only 6:4 Reserved 3:0 -Channel Number Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Byte 2 Parameter revision Format: MSN = present revision LSN = oldest revision parameter is backward compatible with 11h for parameters in this specification. Bytes 3-N Parameter data	
B9h	Set Telnet Configuration Parameters	Request: Byte 1 Flags 7:4 Rreserved 3:0 Channel Number Byte 2 Parameter selector Bytes 3-N Parameter data Response: Byte 1 Completion code	See Table 81 for the definition of the supported parameters.
BAh	Get SNMP Configuration Parameters	Request: Byte 1 Flags 7 Get type 0b = get parameter 1b = get parameter revision only 6:4 Reserved 3:0 -Channel Number Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector Response: Byte 1 Completion code Byte 2 Parameter revision Format:	See Table 80 for the definition of the supported parameters.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		<p>MSN = present revision LSN = oldest revision parameter is backward compatible with 11h for parameters in this specification.</p> <p>Bytes 3-N Parameter data</p>	
BBh	Set SNMP Configuration Parameters	<p>Request: Byte 1 Flags 7:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Bytes 3-N Parameter data</p> <p>Response: Byte 1 — Completion code</p>	See Table 80 for the definition of the supported parameters.
BCh	Get KVM Configuration Parameters	<p>Request: Byte 1 Flags 7 Get type 0b = get parameter 1b = get parameter revision only 6:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Byte 3 Set selector Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector. Byte 4 Block Selector 00h if parameter does not require a block selector</p> <p>Response: Byte 1 Completion code Generic codes, plus following command-specific completion code(s): 80h = Parameter not supported Byte 2 Parameter revision Format: MSN = present revision LSN = oldest revision parameter is backward compatible with 11h for parameters in this specification. Bytes 3-N Parameter data. These bytes are not returned when the 'get parameter revision only' bit is 1b.</p>	See Table 82 for the definition of the supported parameters.
BDh	Set KVM Configuration Parameters	<p>Request: Byte 1 Flags 7:4 Reserved 3:0 Channel Number Byte 2 Parameter selector Bytes 3-N Parameter data</p>	See Table 82 for the definition of the supported parameters.

Net Function = Intel General Application (0x30), LUN = 00			
Code	Command	Request, Response Data	Description
		Response: Byte 1 Completion code 80h = Parameter not supported 81h = Attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters) 82h = Attempt to write read-only parameter	
EEh	Get All SEL Entry	Same request and response parameters as the IPMI Event/Sensor command <i>Get SEL Entry</i> .	This command allows the caller to traverse the entire SEL including the "deleted" SEL entries.
F9h	Get POST Progress	Request: Byte 1 Starting index Byte 2 Count of progress codes and status to get Response: Byte 1 Completion code Bytes 2:N Progress code/status pairs	This command allows the ASF progress codes captured at the last boot to be retrieved. If more progress codes are requested than are queued starting at the provided index, the command data will be truncated to the existing data.

Table 75. Intel® Platform-specific Commands

Net Function = Intel® Platform Specific (0x32), LUN = 00			
Code	Command	Request, Response Data	Description
40h	Get LED Status	Request: None Response: Byte 1 Completion code Byte 2 LED Status bit 1:0 Cooling Fault bit 3:2 System Status bit 5:4 System Ready bit 7:6 System Identify	This command is used to retrieve the LED status. Status definitions: 00b = Off 01b = Blink 10b = On 11b = Invalid (not installed)

Note:

0x32 is the Intel® platform-specific NetFn command.

The *Get LED Status* command is supported on the following server boards when used in conjunction with an Intel® Management Module:

SE7520JR2

SE7520AF2

SE7520BD2

Table 76. Platform Information Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Firmware revision (read-only)	1	<p>Data 1</p> <p>Set selector = Firmware area</p> <p>[7] Reserved</p> <p>[6:0] Firmware area</p> <p>01h = Operational Firmware</p> <p>02h = Boot block</p> <p>03h = PIA (Platform Information Area)</p> <p>00h, 04h – 7Fh is reserved</p> <p>Data 2</p> <p>For Operational or Boot block, this byte returns the major Firmware Revision (binary). If PIA is selected, then PIA format revision is returned (binary).</p> <p>Data 3</p> <p>For Operational or Boot block, this byte returns the minor Firmware Revision (BCD). If PIA is selected, then PIA instance version is returned (binary).</p>
Service partition presence (read-only)	2	<p>This parameter data is the same as the “service partition discovered” bit of parameter 2 (service partition scan) returned by the IPMI command “Get System Boot Options” command.</p> <p>Data 1</p> <p>[7:1] Reserved</p> <p>[0] Service Partition discovered</p> <p>0 = Service partition not discovered</p> <p>1 = Service partition discovered</p>
Chassis status (read-only)	3	<p>This parameter data is the same as what is returned by the IPMI command “Get Chassis Status”.</p>
BIOS version (non-volatile)	4	<p>BIOS version string information.</p> <p>Data 1 Set Selector = 01h.</p> <p>[7:1] reserved.</p> <p>[0] string selector.</p> <p>01h = only one string is supported</p> <p>Data 2- Block Selector = string block number to set, 1 based. Blocks are 16 bytes.</p> <p>[7:2] reserved.</p> <p>[1:0] block selector.</p> <p>only two blocks are supported.</p> <p>Data 3:N String data. Null terminated 8-bit ASCII string. 16-bytes maximum. per block.</p>
Service Partition version (non-volatile)	5	<p>Data 1 Byte1 of SP version</p> <p>Data 2 Byte2 of SP version</p>
Product ID (read-only)	6	<p>Data 1 Byte1 of Product ID</p> <p>Data 2 Byte2 of Product ID</p>

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Advanced Feature Support (read-only)	7	Data 1 Feature support mask [7] Reserved [6] KVM 1 = Supported [5] BMC DHCP/ARP 1 = Supported [4] SMTP 1 = Supported [3] SNMP 1 = Supported [2] HTTPS 1 = Supported [1] HTTP 1 = Supported [0] Telnet 1 = Supported

Table 77. Supported SMTP Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress (volatile)	0	
Sender Machine Name	1	This is the name used by all SMTP configurations. This string identifies the managed server to the SMTP server, it is used with the SMTP "HELO" command. This string is limited to 64 bytes. Data 1 Set Selector = 0 Data 2 Block selector (1-based, blocks are 16 bytes) Data 3:N String data
Number of supported SMTP configurations – read-only	2	This read-only parameter indicates the number of SMTP configurations that are supported on this BMC. Response Data: Byte 3 Maximum number of configurations supported Byte 4 Number of currently valid configurations. Validity is determined by legitimate configuration of 3 other parameters: Sender Machine Name, From Address and To Address.
email From: Address	3	This parameter provides the "From:" line for an SMTP alert. The Set Selector indicates which SMTP configuration to reference. Each "From:" string is up to 64 bytes. Data 1 Set Selector = configuration selector Data 2 Block selector (1 based, blocks are 16 bytes) Data 3:N String data
email To: address	4	This parameter provides the "To:" line for an SMTP alert. The Set Selector indicates which SMTP configuration to reference. Each "To:" string is up to 64 bytes. Data 1 Set Selector = configuration selector Data 2 Block selector (1 based, blocks are 16 bytes) Data 3:N String data

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
email Subject: line	5	This parameter provides the "Subject:" line for an SMTP alert. The Set Selector indicates which SMTP configuration to reference. Each "Subject:" string is up to 64 bytes. Data 1 Set Selector = configuration selector Data 2 Block selector (1 based, blocks are 16 bytes) Data 3:N String data
DNS Override	6	This parameter determines whether to use the IP address defined in the destinations section of LAN configuration data or the SMTP server name defined in the following parameter instead. A non-zero value indicates that the SMTP Server Name should be used.
SMTP Server Name	7	When DNS Override is non-zero, this parameter provides an SMTP server name to use when sending eMail alerts to the address specified in this SMTP configuration. In this case, the IP address specified in the correlated LAN destination is ignored. Each Server Name is up to 64 bytes. Data 1 Set Selector = configuration selector Data 2 Block selector (1 based, blocks are 16 bytes) Data 3:N String data

Table 78. Supported HTTP Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress (volatile)	0	
Health Icon Update Rate	1	This parameter is used to control the behavior of the Health Icon. Data 1:2 Healthy System Update Rate (in seconds) 0 = Updates only done on user request, default = 120 sec Data 3:4 Warning System Update Rate (in seconds) 0 = Updates only done on user request, default = 30 sec Data 5:6 Critical System Update Rate (in seconds) 0 = Updates only done on user request, default = 120 sec
Web Update of EFS	2	Data 1 Set Selector 01h EFS Update pathname (default = !/@efsUpdate) 02h EFS Info pathname (default = !/@efsInfo) Data 2:17 Pathname These strings must be NULL terminated
Useful Constants	3	Data 1 Set Selector 01h %%usrUrl1%% value (96 bytes maximum) 02h %%usrUrl2%% value (96 bytes maximum) 03h %%usrVal1%% value (32 bytes maximum) 04h %%usrVal2%% value (32 bytes maximum) Data 2 Block Selector Long strings are composed of multiple 16 byte data blocks Data 3:18 Data bytes These strings must be NULL terminated if shorter than full length

Table 79. Supported HTTP Channel Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress (volatile)	0	
Useful Constants	1	<p>This parameter allows various web page macro expansions to be defined.</p> <p>Data 1 Set Selector</p> <p>01h %%hostname%% value (16 bytes maximum)</p> <p>02h %%domain%% value (32 bytes maximum)</p> <p>03h %%telnetUrl%% value (96 bytes maximum)</p> <p>04h %%kvmUrl%% value (96 bytes maximum)</p> <p>Data 2 Block Selector</p> <p>Long strings are composed of multiple 16 byte data blocks</p> <p>Data 3:18 Data bytes</p> <p>These strings must be NULL terminated</p>
HTTP Port Enable	2	<p>This parameter is used to enable or disable the HTTP port on a channel.</p> <p>Data 1 Flags</p> <p>[7:2] Reserved</p> <p>[1:0] HTTP Enable</p> <p>00b = Port Disabled</p> <p>01b = Port Enabled</p> <p>10b = Port not supported on this channel</p> <p>11b = Reserved</p>
HTTPS Port Enable	3	<p>This parameter is used to enable or disable the HTTPS port on a channel.</p> <p>Data 1 Flags</p> <p>[7:2] Reserved</p> <p>[1:0] HTTPS Enable</p> <p>00b = Port Disabled</p> <p>01b = Port Enabled</p> <p>10b = Port not supported on this channel</p> <p>11b = Reserved</p>
HTTP Port Number	4	<p>This is the TCP port number used by the HTTP server on this channel.</p> <p>Changing the port number while using the HTTP feature is not recommended.</p> <p>Attempting to use a previously used port number requires a BMC reset.</p> <p>Data 1-2 Port number (Default is 80)</p>
HTTPS Port Number	5	<p>This is the TCP port number used by the HTTPS server on this channel.</p> <p>Changing the port number while using the HTTPS feature is not recommended.</p> <p>Attempting to use a previously used port number requires a BMC reset.</p> <p>Data 1-2 Port number (Default is 443)</p>

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Use DNS Host Domain	6	<p>Control where the host name and domain name come from.</p> <p>Data 1 Host name source</p> <p>[7:1] Reserved</p> <p>[0] Source</p> <p>0b = Prefer DNS for hostname lookup. (default)</p> <p>1b = Ignore DNS and use values from parameter 1 (above).</p> <p>By default, the system will attempt to look up the hostname using DNS. If there is a DNS server available, and the IP address resolves to hostname then that value will be used instead of the value managed by parameter 1 of this command.</p> <p>If this value is set to 1, then the value set by parameter 1 will be used and DNS will not be used to resolve the host name.</p>

Table 80. Supported SNMP Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress (volatile)	0	
SNMP Enable	1	<p>This parameter is used to enable or disable the SNMP feature on a channel.</p> <p>Data 1 Flags</p> <p>[7:2] Reserved</p> <p>[1] Community Sets Enabled, 1 = Enabled, 0 = Disabled (default)</p> <p>[0] SNMP Enable</p> <p>1 = Enabled</p> <p>0 = Disabled (default)</p>
Port Number	2	<p>This is the UDP port number used by the SNMP server on this channel.</p> <p>Changing the port number while using the SNMP feature is not recommended.</p> <p>Attempting to use a previously used port number requires a BMC reset.</p> <p>Data 1-2 Port number (Default is 161)</p>

Table 81. Supported Telnet Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress (volatile)	0	
Telnet Enable	1	This parameter is used to enable or disable the SNMP feature on a channel. Data 1 Flags [7:1] Reserved [0] Telnet Enable, 1 = Enabled, 0 = Disabled (default)
Port Number	2	This is the TCP port number used by the Telnet server on this channel. Changing the port number while using the telnet feature is not recommended. Attempting to use a previously used port number requires a BMC reset. Data 1-2 Port number (Default is 23)
Newline Sequences	3	Data 1 [7:3] Reserved [2:0] Output newline sequence (BMC to console). Selects what characters the BMC uses as a <newline> sequence when the BMC writes a line to the console in Telnet Mode. 0h = Reserved 1h = <CR-LF> 2h = <NULL> 3h = <CR> 4h = <LF-CR> 5h = <LF> (default) all other = reserved
Prompt String	4	Data 1:16 Prompt String Default = 'CLI'

Table 82. Supported KVM Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
Set In Progress (volatile)	0	<p>Data 1 This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software than some other software or utility is in the process of making changes to the data.</p> <p>An implementation can also elect to provide a 'rollback' feature that uses this information to decide whether to 'roll back' to the previous configuration information, or to accept the configuration change.</p> <p>If used, the roll back shall restore all parameters to their previous state. Otherwise, the change shall take effect when the write occurs.</p> <p>[7:2] reserved</p> <p>[1:0] Set Status/Control</p> <p>00b = set complete. If a system reset or transition to powered down state occurs while 'set in progress' is active, the BMC will go to the 'set complete' state. If rollback is implemented, going directly to 'set complete' without first doing a 'commit write' will cause any pending write data to be discarded.</p> <p>01b = set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The BMC does not provide any interlock mechanism that would prevent other software from writing parameter data while.</p> <p>10b = commit write (optional). This is only used if a rollback is implemented. The BMC will save the data that has been written since the last time the 'set in progress' and then go to the 'set in progress' state. An error completion code will be returned if this option is not supported.</p> <p>11b = reserved</p>
KVM Enable	1	<p>This parameter is used to enable or disable the KVM feature on a channel.</p> <p>Data 1</p> <p>[7:1] Reserved</p> <p>[0] KVM Enable for new sessions. Does not affect current active sessions.</p> <p>1 = Enabled</p> <p>0 = Disabled (default)</p>
Max Request Data Structure support (Read Only)	2	<p>Data 1 The number of "Request Data Structures" it can handle in a single Video Data Request. This determines how many discontinuous sets of video data can be requested in a Video Data Request packet.</p> <p>Note: All request data structures must be for the same frame.</p>
Video Packet Throttle	3	<p>Data 1 Minimum delay, in milliseconds, between video payload packets issued by the BMC. 1-based. Default = 00h (no delay).</p>
Video Frame Throttle	4	<p>Data 1 Minimum delay, in milliseconds, between frames issued by the BMC. 1-based. Default = 00h (no delay). The minimum delay between video frames will be the greater of the Video Packet Throttle or Video Frame Throttle values.</p>

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
Local Keyboard Control Policy	5	Data 1 Policy Flags [7:4] reserved [3:0] Local Keyboard Control Policy 0h = remote console determines policy 1h = local keyboard activity terminates remote KVM 2h = local keyboard activity terminates remote K/M
Local Keyboard Timeout	6	Data 1 Number of 10s of seconds of inactivity on local keyboard before remote KVM is automatically re-enabled. 1-based. 00h = remote console can attempt to restore remote KVM control 'immediately'.

Table 83. Supported User Feature Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set in progress	0	
Telnet Enables	1	This parameter provides per-channel enables for the Telnet feature. Data 1 Set Selector = User ID Number Data 2 Telnet Enables [7:1] Reserved [0] Telnet Enable
HTTP Enables	2	This parameter provides per-channel enables for the HTTP feature. Data 1 Set Selector = User ID Number Data 2 HTTP Enables [7:1] Reserved [0] HTTP Enable Data 3 HTTP CGI Capabilities [7] efsUpload [6] efsInfo [5] cliXml [4] ipmiXml [3] cliPage [2] ipmiPage [1] cliTxt [0] ipmiHex
HTTPS Enables	3	This parameter provides per-channel enables for the HTTPS feature. Data 1 Set Selector = User ID Number Data 2 HTTPS Enables [7:1] Reserved [0] HTTPS Enable

Table 84. SOL 2.0 Commands

Net Function = Transport, LUN = 00		
Code	Command	Request, Response Data
20h	SOL 2.0 Activating	Per the <i>IPMI 2.0 specification</i>
21h	Set SOL 2.0 Configuration Parameters	Per the <i>IPMI 2.0 specification</i>
22h	Get SOL 2.0 Configuration Parameters	Per the <i>IPMI 2.0 specification</i>

Table 85. Supported SOL 2.0 Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
Set In Progress (volatile)	0	Per the IPMI 2.0 specification
SOL Enable	1	Per the IPMI 2.0 specification
SOL Authentication	2	Per the IPMI 2.0 specification
Character Accumulate Interval and Character Send Threshold	3	Per the IPMI 2.0 specification
SOL Retry	4	Per the IPMI 2.0 specification
SOL non-volatile bit rate (non-volatile)	5	Per the IPMI 2.0 specification
SOL volatile bit rate (volatile)	6	Per the IPMI 2.0 specification

Table 86. Intel OEM Terminal Mode Text Commands

Command Text	Description
SYS 000157 ACTIVATE	This command activates the Intel OEM Terminal Mode extensions defined here. This allows Intel OEM text commands to be entered without the “000157” text prefix. To enable the OEM commands this command must be executed every time a session is activated.
SYS BOOT [-F] SYS BOOT [-F] NORMAL SYS BOOT [-F] SERVICE1	The boot command sets the IPMI boot options and resets the system. This command is different than the IPMI “SET BOOT” command. BOOT: This command attempts a graceful shutdown of the operating system before resetting. With no parameters this is the same as “SHUTDOWN -R”. BOOT -F: Forces a hard reset (no graceful shutdown). With no additional parameters this is the equivalent of “SHUTDOWN -R -F”. BOOT (NORMAL SERVICE): This command can also have the -f F option. Sets the needed boot flags for use on the next boot. This command only sets the required IPMI boot flags options for hard disk (normal) and diagnostics (service). If the BIOS does not support the required boot flags it will have no effect on the boot sequence.
SYS DIAGINT	Command will cause the BMC to generate an IPMI diagnostic interrupt (NMI for IA-32 systems).

Command Text	Description
SYS HELP {{command}}	<p>Displays the syntax and usage information for the command specified. If a command is not specified the current IPMI revision, FW version numbers and all supported commands are displayed. The BMC returns a string of the following format:</p> <p>For a specific command: Command: xx<output termination sequence></p> <p>If command is not specified: IPMI Rev: x.x<output termination sequence> FW Rev: xx.xx<output termination sequence> [Commands <output termination sequence> ..]</p>
SYS ID	<p>Displays the 16-byte system GUID of the managed server. Output formatted similar to what is displayed by the DPCCLI command. The BMC returns a string of the following format:</p> <p>xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</p>
SYS IDENTIFY {{(ON {# of seconds} OFF) {-S}}	<p>Commands the server to signal its location (e.g. with a blinking led or beep).</p> <p>IDENTIFY = enables the server to identify itself for 15 seconds. IDENTIFY ON = enables the server to identify itself for 15 seconds. IDENTIFY ON XX = Causes the BMC to signal the system's location for a specific amount of time. XX is a hex-ASCII byte representing the number of seconds the BMC is to cause the system to identify itself. If XX is 00 the BMC will signal the system's location until a request to stop (via the IDENTIFY OFF command, other software commands or the ID Button) is received. IDENTIFY OFF = Causes the BMC to stop signaling the system's location. This has no effect if the system is not currently identifying itself. IDENTIFY -S = Displays the current identify state and source of request. (ON (Application) ON (Button) OFF) <output termination sequence></p>
SYS NETWORK {{(IP MAC SUBNET GATEWAY)}}	<p>Displays the network configuration of the BMC. Using the IP, MAC, SUBNET or GATEWAY parameters only display the requested information. Output formatted similar to what is displayed by the DPCCLI command. This only displays information for the primary OOB NIC. The BMC returns the response string in the following format:</p> <p>IP Address : x.x.x.x <output termination sequence> IP Address Source : (Static DHCP BIOS Other) <output termination sequence> MAC Address : xx:xx:xx:xx:xx:xx <output termination sequence> Subnet Mask : x.x.x.x <output termination sequence> Gateway : x.x.x.x <output termination sequence></p>
SYS SHUTDOWN {[-R] [-F]}	<p>Shutdown SHUTDOWN = Graceful operating system shutdown and power off Shutdown -f SHUTDOWN -F = Forced system power off Shutdown -r SHUTDOWN -R = Graceful operating system shutdown then a system hard reset Shutdown -r -f SHUTDOWN -R -F = Forced system hard reset.</p>

Table 87. ICMB Bridge Device Commands

Code	Net Function	Command	Allowed Over ICMB?	Allowed Over IPMB?	Notes
00h	Bridge	Get Bridge State	X	X	1
01h	Bridge	Set Bridge State	X	X	1
02h	Bridge	Get ICMB Address	X	X	
03h	Bridge	Set ICMB Address	X	X	1
04h	Bridge	Set Bridge Proxy Address	X	X	1
05h	Bridge	Get Bridge Statistics	X	X	
06h	Bridge	Get ICMB Capabilities	X	X	1
08h	Bridge	Clear Bridge Statistics	X	X	
09h	Bridge	Get Bridge Proxy Address	X	X	1
0Ah	Bridge	Get ICMB Connector Info	X	X	1, 3
0Bh	Bridge	Get ICMB Connection ID	X	X	1, 2, 3
0Ch	Bridge	Send ICMB Connection ID	X	X	1, 2, 3
10h	Bridge	Prepare For Discovery	X	X	2
11h	Bridge	Get Addresses	—	X	2
12h	Bridge	Set Discovered	X	X	
13h	Bridge	Get Chassis Device ID	X	X	1
14h	Bridge	Set Chassis Device ID	X	X	1
20h	Bridge	Bridge Request	X	X	1, 2
21h	Bridge	Bridge Message	X	X	1, 2
30h	Bridge	Get Event Count	X	X	1
31h	Bridge	Set Event Destination	X	X	1
32h	Bridge	Set Event Reception State	X	X	1
33h	Bridge	Send ICMB Event Message	X	X	1, 2
00h	Application	Get Device ID	—	X	
00h	Application	Broadcast Get Device ID	—	X	
04h	Application	Get Self Test Results	—	X	
06h	Application	Set ACPI Power State	—	X	
07h	Application	Get ACPI Power State	—	X	

Notes:

1. Only allowed over ICMB once the chassis has been discovered at least once (i.e., it has received a SetDiscovered message).
2. Only allowed over IPMB if the ICMB bridge state is "Enabled". These commands typically require generation of ICMB traffic.
3. *Connector ID* commands are only supported on platforms that implement the *Connector ID* signals on the ICMB cable.

7.4 Completion Codes

For convenience, the following table defines standard IPMI 2.0 completion codes. Some commands may implement other, command specific completion codes. These are defined in the IPMI 2.0 specification, or if Intel® Management Module BMC specific, in the command tables above.

Table 88. Completion Codes

Completion Code	Meaning
00h	Command successfully completed.
C0h	Node Busy. Command could not be processed because command processing resources are temporarily unavailable.
C1h	Invalid Command. Used to indicate an unrecognized or unsupported command.
C2h	Command invalid for given LUN.
C3h	Timeout while processing command. Response unavailable.
C4h	Out of space. Command could not be completed because of a lack of storage space required to execute the given command operation.
C5h	Action not performed due to a canceled or invalid reservation number.
C7h	Command has wrong number of data bytes.
C8h	Too many data bytes for requested operation.
C9h	One or more of the data bytes in a command is outside the range of valid values.
CAh	Cannot return number of requested data bytes.
CBh	The request is for a nonexistent sensor in the BMC
CCh	One or more of the data bytes is an invalid value.
CDh	The requested operation is not supported for the given sensor.
CEh	Command not completed; a BMC peripheral did not respond.
D3h	Cannot deliver request to selected destination
D4h	Cannot execute command. Insufficient privilege level.
FFh	Unspecified error

7.5 Command Support

Table 89 through Table 94 specify the privilege level associated with each command as well as whether the command is supported in Operational and Firmware Transfer modes. Superscripts refer to the notes on special treatment below.

- Only allowed over the System Interface
- Only available over Local Interface (System Interface, IPMB or PCI SMBus)

This section does not list the commands available on the SMM Interface.

Note: As stated in the IPMI 2.0 Specification, unless otherwise specified, unauthenticated, session-less interfaces, such as the System Interface and IPMB, can support any IPMI command.

Table 89. Application Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Application – 06h	Get Device ID	01h	User	X
	Broadcast Get Device ID	01h	User (note 2)	
	Get Self Test Results	04h	User	X
	Set ACPI Power State	06h	Admin	
	Get ACPI Power State	07h	User	
	Reset Watchdog Timer	22h	Operator	
	Set Watchdog Timer	24h	Operator	
	Get Watchdog Timer	25h	User	
	Set BMC Global Enables	2Eh	Admin (note 1)	
	Get BMC Global Enables	2Fh	User	
	Clear Message Flags	30h	Admin (note 1)	
	Get Message Flags	31h	Admin (note 1)	
	Get Message	33h	Admin (note 1)	
	Send Message	34h	User	
	Read Event Message Buffer	35h	Admin (note 1)	
	Get System GUID	37h	Any	
	Get Channel Authentication Capabilities	38h	Any	
	Get Session Challenge	39h	Any	
	Activate Session	3Ah	Any	
	Set Session Privilege Level	3Bh	User	
	Close Session	3Ch	Any	
	Get Session Info	3Dh	User	
	Get AuthCode	3Fh	Operator	
	Set Channel Access	40h	Admin	
	Get Channel Access	41h	User	
	Get Channel Info	42h	User	
	Set User Access	43h	Admin	
	Get User Access	44h	Operator	
	Set User Name	45h	Admin	
	Get User Name	46h	Operator	
	Set User Password	47h	Admin	
	Activate Payload	48h	See IPMI 2.0 spec	
	Deactivate Payload	49h	See IPMI 2.0 spec	
	Get Payload Activation Status	4Ah	User	
	Get Payload Instance Info	4Bh	User	
	Set User Payload Access	4Ch	Admin	
	Get User Payload Access	4Dh	User	
	Get Channel Payload Support	4Eh	User	
	Get Channel Payload Version	4Fh	User	
	Get Channel OEM Payload Info	50h	User	
Master Write-Read I2C	52h	Operator		
Get Channel Cipher Suites	54h	Any		
Suspend/Resume Payload Encryption	55h	User		
Set Channel Security Keys	56h	Admin		

¹ Only available via the system (SMS) interface.

² Only available via the IPMB interface

Table 90. Chassis Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Chassis – 00h	Get Chassis Capabilities	00h	User	
	Get Chassis Status	01h	User	
	Chassis Control	02h	Operator (note 1)	
	Chassis Identify	04h	Operator	
	Set Power Restore Policy	06h	Operator	
	Get System Restart Cause	07h	User	
	Set System Boot Options	08h	Operator	
	Get System Boot Options	09h	Operator	
	Get POH Counter	0Fh	User	

1 The system can be powered up using the Chassis Control command at User privilege level.

Table 91. Sensor/Event Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Sensor/Event – 04h	Get Event Receiver	01h	User	
	Platform Event	02h	Operator	
	Get PEF Capabilities	10h	User	
	Arm PEF Postpone Timer	11h	Admin	
	Set PEF Configuration Parameters	12h	Admin	
	Get PEF Configuration Parameters	13h	Operator	
	Set Last Processed Event ID	14h	Admin	
	Get Last Processed Event ID	15h	Admin	
	Alert Immediate	16h	Admin	
	PET Acknowledge	17h	Any	
	Set Sensor Hysteresis	24h	Operator	
	Get Sensor Hysteresis	25h	User	
	Set Sensor Threshold	26h	Operator	
	Get Sensor Threshold	27h	User	
	Set Sensor Event Enable	28h	Operator	
	Get Sensor Event Enable	29h	User	
	Re-Arm Sensor Events	2Ah	Operator	
	Get Sensor Event Status	2Bh	User	
	Get Sensor Reading	2Dh	User	

Table 92. Storage Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Storage – 0Ah	Get FRU Inventory Area Info	10h	User	
	Read FRU Data	11h	User	
	Write FRU Data	12h	Operator	
	Get SDR Repository Info	20h	User	
	Get SDR Repository Allocation Info	21h	User	
	Reserve SDR Repository	22h	User	
	Get SDR	23h	User	
	Partial Add SDR	25h	Operator	
	Delete SDR	26h	Operator	
	Clear SDR Repository	27h	Operator	
	Get SDR Repository Time	28h	User	
	Run Initialization Agent	2Ch	Operator	
	Get SEL Info	40h	User	
	Get SEL Allocation Info	41h	User	
	Reserve SEL	42h	User	
	Get SEL Entry	43h	User	
	Add SEL Entry	44h	Operator	
	Partial Add SEL Entry	45h	Operator	
	Delete SEL Entry	46h	Operator	
	Clear SEL	47h	Operator	
Get SEL Time	48h	User		
Set SEL Time	49h	Operator		

Table 93. Transport Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Transport – 0Ch	Set LAN Configuration	01h	Admin	
	Get LAN Configuration	02h	Operator	
	Suspend BMC ARPs	03h	Admin	
	Get IP/UDP/RMCP Statistics	04h	User	
	Set Serial/Modem Configuration	10h	Admin	
	Get Serial/Modem Configuration	11h	Operator	
	Set Serial/Modem Mux	12h	Operator	
	Set PPP UDP Proxy Transmit Data	14h	SMS only	
	Get PPP UDP Proxy Transmit Data	15h	SMS only	
	Send PPP UDP Proxy Packet	16h	SMS only	
	Get PPP UDP Proxy Receive Data	17h	SMS only	
	Serial/Modem Connection Active	18h	N/A	
	Callback	19h	Callback	
	Set User Callback Options	1Ah	Admin	
	Get User Callback Options	1Bh	User	
	SOL 2.0 Activating	20h	N/A	
	Set SOL 2.0 Configuration Parameters	21h	Admin	
Get SOL 2.0 Configuration Parameters	22h	Operator		

Table 94. Intel General Application Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Intel General Application – 30h	Read Self Test	16h	User	
	Get Secure Mode Options	1Fh	User	
	Get Power Supply State	20h	User	
	Get DIMM State	21h	User	
	ReArm DIMMs	23h	Operator	
	Get Processor State	29h	User	
	Re-Arm Processors	2Ah	Operator	
	Chassis Identify LED	36h	Operator	
	CMOS Clear Options	52h	Operator	
	Set Fault Indication	57h	Operator	
	Get Platform Information	64h	Any	
	Get Memory RAS Redundancy State	8Ah	Operator	
	Get Memory RAS Configuration	8Ch	Operator	
	Get Chassis Name	90h	User	
	Set Chassis Name	91h	Operator	
	Get All SEL Entry	Eeh	User	
Get POST Progress	F9h	User		

Table 95. Intel Advanced Feature Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Intel General Application – 30h	Get User Feature Configuration Parameters	B0h	User	
	Set User Feature Configuration Parameters	B1h	Admin	
	Get SMTP Configuration Parameters	B2h	User	
	Set SMTP Configuration Parameters	B3h	Admin	
	Get HTTP Configuration Parameters	B4h	User	
	Set HTTP Configuration Parameters	B5h	Operator	
	Get HTTP Channel Configuration Parameters	B6h	User	
	Set HTTP Channel Configuration Parameters	B7h	Operator	
	Get Telnet Configuration Parameters	B8h	User	
	Set Telnet Configuration Parameters	B9h	Admin	
	Get SNMP Configuration Parameters	BAh	User	
	Set SNMP Configuration Parameters	BBh	Admin	
	Get KVM Configuration Parameters	BCh	User	
	Set KVM Configuration Parameters	BDh	Admin	

Table 96. Intel Platform Specific Command Support Matrix

Network Function	Command Name	Cmd #	Operational Mode Privilege	Firmware Update
Intel® Platform Specific – 32h	Get LED Status	40h	Operator	

7.6 RMCP+ Command Interface

Table 97. BMC RSSP/RAKP Messages

Session Setup Payload Types			
6-bit Number	Command	Request, Response Data	Description
10h	RSSP Open Session Request	As defined by the IPMI 2.0 specification	The remote console sends this RSSP message to the managed system to open a protected session.
11h	RSSP Open Session Response	As defined by the IPMI 2.0 specification	This command is sent from the BMC to the client. It is not accepted by the BMC
12h	RAKP Message 1	As defined by the IPMI 2.0 specification	A remote console sends this RAKP message to the managed system to begin the session authentication process.
13h	RAKP Message 2	As defined by the IPMI 2.0 specification	This command is sent from the BMC to the client. It is not accepted by the BMC
14h	RAKP Message 3	As defined by the IPMI 2.0 specification	A remote console sends this RAKP message to a managed system in response to the receipt of an RAKP Message 2.
15h	RAKP Message 4	As defined by the IPMI 2.0 specification	This command is sent from the BMC to the client. It is not accepted by the BMC

Table 98. OEM Payload Types

Session Setup Payload Types			
6-bit Number	Command	Request, Response Data	Description
20h	KVM Video Payload	As defined by the IPMI 2.0 specification	All KVM Video payload packets use this payload type.
21h	KVM Keyboard/Mouse Payload	As defined by the IPMI 2.0 specification	All KVM Keyboard/Mouse payload packets use this payload type.

8. Environmental Specifications

8.1 Environmental Ratings

8.1.1 Maximum Ratings

Table 99 contains absolute maximum ratings for the Intel® Management Module BMC component. Functional operation at the absolute maximum and minimum is neither implied nor guaranteed. The BMC should not receive a clock while subjected to these conditions. Extended exposure to the maximum ratings may affect device reliability. Although the BMC contains protective circuitry to resist damage from static discharge, always take precautions to avoid high static voltages or electric fields.

Table 99. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
Top	Operating temperature under bias	0	70	°C
Tstorage	Storage temperature	-65	150	°C
Vdd	Core and IO power supply voltage (3.3V +/- 10%)	3.0	3.6	V
Vdd5	5V tolerant IO supply voltage	3.6	5.5	
Vin3.3	Input voltage range for 3.3V IO	-0.5	5.0	V
Vin5	Input voltage 5V tolerant IO	-0.5	7.0	V

8.1.2 Maximum Current Requirements

Below are maximum current and wattage requirements for the three board configurations. BMC flash and SDRAM access do not occur simultaneously, so FLASH current requirements are reduced from 18mA to 5mA. FPGA requirement of 500mA is based on initial configuration current requirements. NIC 82551QM is based on D3 (hot) wakeup enable since full PCI interface is not utilized.

IMM Power	P3V3_STBY	3.3VAUX	Max mW
	mA	mA	
Sahalee	136		
BMC flash	5		
BMC sram	90		
FRU EEPROM	3		
buffers	20		
misc res	30		
PRO SUM	284		938.4
DVI rcvr	0	170	
video sram	0	160	
FPGA	500		
FPGA config	15		
misc res	30		

IMM Power	P3V3_STBY	3.3VAUX	Max mW
ADV SUM	829	330	3825.9
NIC 82551	125		
FRU EEPROM	3		
SPI EEPROM	3		
Misc res	30		
GCM SUM	161		531.3

8.1.3 BMC Thermal

Symbol	Parameter	Min	Typ	Max	Unit
Tamb	Ambient operating temperature	0		80	°C
Tcase	Case operating temperature	0		98	°C
TJ	Junction operating temperature	0		100	°C
Pd	Power dissipation 45 Mhz core clock			0.45	W
θ_{jc}	Junction to case temperature change per watt dissipated		2.7		°C/W
θ_{ja} (0 LFM)	Junction to Air temperature change per watt dissipated with no air flow over the package.		40.0		°C/W

8.1.4 DC Specifications

All pins on the Sahalee component are 5 volt tolerant with the exception of the LPC signals, which are PCI compatible. The following sections describe the DC specifications of the signals.

8.1.4.1 5 Volt Tolerant Signals

The 5V tolerant voltage levels described below are compatible with both 3.3 and 5 volt CMOS and TTL signaling. This enables them to be interfaced to devices of either technology.

Note: That there are two varieties of 5V tolerant bidirects: one with Schmitt input characteristics, and one without. Schmitt bidirects are used on the I2C signals. Schmitt input-only cells are used on the TIC2 signals.

All 5V tolerant bi-directional signals have a weak internal pull-up in them. The effective resistance of the pull-up changes with pin voltage, and is very high at low pin voltage. The Schmitt input only cells (TIC2 ports) have repeaters, not pull-ups. These pull-ups should not be relied on for pulling up a heavily loaded bus; they are included to ensure that a floating input will *eventually* get pulled to a logic '1'.

While being 5V tolerant, all Sahalee outputs are driven to 3.3V levels.

Table 100. DC Specifications for 5V Tolerant Signals

Symbol	Parameter	Min	Max	Unit
Vil 5Vtol	Input low voltage, non-schmitt	-0.5	0.3Vdd	V
Vil Schmitt	Input low voltage, Schmitt	-0.5	0.8	V
Vih 5Vtol	Input high voltage, non-schmitt	0.5Vdd	Vdd5+0.3	V
Vih Schmitt	Input high voltage, Schmitt	2.0	Vdd5+0.3	V
Vol 5Vtol	Output low voltage (Iol = 2 to 12 mA)		0.4	V
Voh 5Vtol	Output high voltage (Ioh = -2 to -12 mA)	2.4		V
Iil 5Vtol	Input leakage current		+/-10	uA
	Pullup resistance @ 2V, non-schmitt bidirect	24	100	KΩ
	Pullup resistance @ 2V, Schmitt bidirect	6	100	KΩ
	Schmitt hysteresis (peak-to-peak)	0.5		V

If an SMBus device is connected to an I2C port on Sahalee, note that the maximum leakage current is greater than the SMBus specification. This does not make it impossible to put an SMBus device off of Sahalee's I2C ports, but it must be considered when determining the loading on the bus.

8.1.5 LPC Signals (3.3V PCI)

The DC specifications for the LAD[3:0], LFRAME#, LDRQ# and SYSIRQ are set the same as defined for bussed signals in Section 4.2.2.1 of the *PCI Local Bus Specification, Revision 2.1*, which contains the specifications for the 3.3V signaling environment. The I/O drivers on the LPC signals are PCI 2.1 compliant drivers.

Table 101. DC Specifications for LPC Signals (3.3 V)

Symbol	Parameter	Min	Max	Unit
Vil LPC	Input low voltage	-0.5	0.3Vdd	V
Vih LPC	Input high voltage	0.5Vdd	Vdd+0.5	V
Vol LPC	Output low voltage (Iol = 1.5 mA)		0.1Vdd	V
Voh LPC	Output high voltage (Ioh = -.5 mA)	0.9Vdd		V
Iil LPC	Input leakage current		+/-10	uA

8.1.6 FMLBus DC Specifications

This table below summarizes the DC specifications of the bus, which applies for both master and slave:

Table 102. FML Bus DC Specifications

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
Vil	Data, Clock input low voltage	-	0.8	V	
Vih	Data, Clock input high voltage	2.0	-	V	
Vol	Data, clock output low voltage	-	0.4	V	
Voh	Data, clock output high voltage	2.4	-	V	
Vdd	Nominal bus voltage	3.0	3.6	V	3.3V typical
Iih	Input high current	-	15	uA	
Iil	Input low current	15	-	uA	

8.2 AC Specifications

8.2.1 I²C Interface

Where calculations are provided, the I2C_CLK_DIV register must be set to meet these constraints based on the Sahalee internal clock frequency. Tcyc is the internal Sahalee clock period.

Symbol	Parameter	Min	Max	Unit	Notes
Freq	Operating frequency		400	KHz	
Tbuf	Bus free time between Stop and Start condition (= Tcyc * (I2C_CLK_DIV+16))	4.7		us	
thd:sta	Hold time after (repeated) start condition. After this period, the first clock is generated (= Tcyc * (I2C_CLK_DIV-8))	4.0		us	
tsu:sta	Repeated Start condition setup time (= Tcyc * (I2C_CLK_DIV+15))	4.0		us	
tsu:sto	Stop condition setup time (= Tcyc * (I2C_CLK_DIV+15))	4.0		us	
thd:data	Data hold time from SCL	300		ns	
tsu:data	Data setup time to SCL	250		ns	
Tf	Clock/Data fall time into 100 pF capacitance and 4.7K ohm pullup.		300	ns	1

8.2.2 LPC Interface

The AC specifications for the LAD[3:0], LFRAME#, LDRQ# and SYSIRQ meet the requirements for bussed signals spelled out in Section 4.2.3.2 of the *PCI Local Bus Specification*, Revision 2.1, with the exception noted below (see Note 1) on the following page.

The following figures define the conditions under which input and outputs timing measurements are made. As per the PCI spec, V_{test} is set to $0.4 \cdot V_{cc}$, V_{trise} is $0.285 \cdot V_{cc}$, and V_{fall} is $0.615 \cdot V_{cc}$.

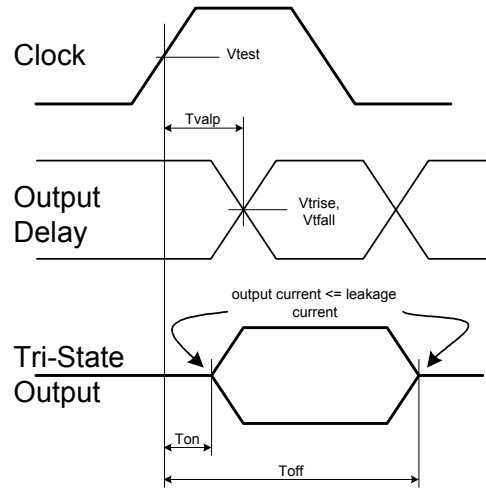


Figure 19. Output Timing Measurement Conditions

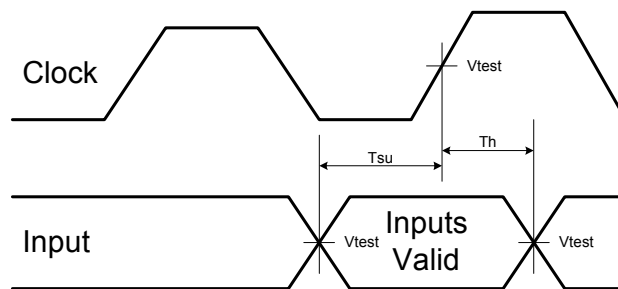


Figure 20. Input Timing Measurement Conditions

Symbol	Parameter	Min	Max	Unit	Notes
Tvalp	LCLK to signal valid delay	2	11	ns	
Tsu	Input setup time to LCLK	2		ns	
Th	Input hold time from LCLK	1		ns	1
Ton	Float to active delay	2		ns	
Toff	Active to float delay		11	ns	

Note 1: The hold time for the LPC inputs is 1ns with respect to LCLK, not 0ns as is defined in the PCI spec. Since the minimum clock to output of any LPC device is 2 ns, care must be taken by the system designer to ensure that Sahalee does not see an LCLK more than 1 ns slower than the LPC host agent's LCLK.

8.2.2.1 LPC Clock

The following clock waveform must be delivered to the Sahalee LPC clock pin, LCLK. This waveform is compatible with section 4.2.3 of the *PCI Local Bus Specification, Revision 2.1*.

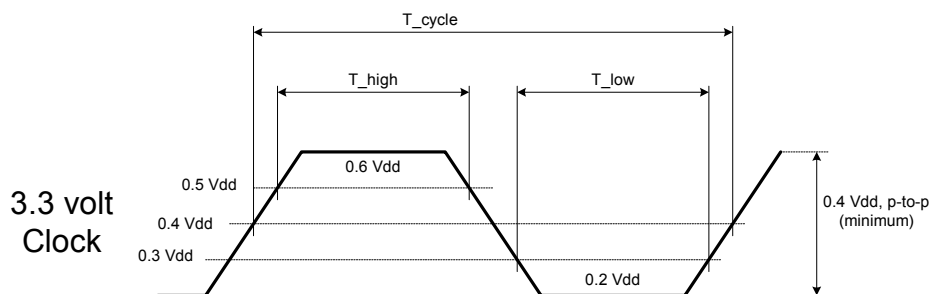


Figure 21. LPC Clock Waveform

Table 103. LPC Clock Specifications

Symbol	Parameter	Min	Max	Unit	Notes
Freq	Clock frequency		33	Mhz	1
T _{high}	Clock high time	5		ns	
T _{low}	Clock low time	5		ns	
T _{slew}	Clock slew rate	1	4	V/ns	

Note 1: Internally, Sahalee's LPC interface can run significantly faster (>66 MHz), however I/O timings would start to limit maximum frequency.

8.2.3 UART interface

Table 104. UART Specifications

Symbol	Parameter	Min	Max	Unit	Notes
	Baud rate error tolerance		6	%	

Appendix A: OEM SDR Record Formats

This section describes the format for OEM SDR records supporting various BMC features. OEM SDR records are used in cases where a platform needs to be configured depending on what type of chassis or environment it is operating in.

OEM SDR records are of type C0h. They contain a Manufacturer ID and OEM data in the record body. Intel OEM SDR records also have a sub-type field in them as the first byte of the OEM data that indicates the type of record following. The following table provides the record subtype number assignments.

Table 105. Intel OEM SDR Subtypes

Value	Description
02h	Power Unit Map SDR Record
05h	Fan Redundancy Map SDR Record
06h	System Information SDR Record
07h	Tcontrol Fan Speed Control SDR Record
09h	PC87427 SIO3 Fan Measurement Configuration Record
0Ah	Web Server Map Table Size
0Bh	Web Server Map Table Entry
53h	SDR Version Record (first letter of 'SDR')
54h	OEM SDR Tag Record
00h, 01h, 03h, 04h, 0Ch-52h, 55h-FFh	Reserved

Power Unit Map Record

A Power Unit Map SDR record is an OEM SDR record (type C0h) with the Intel manufacturers ID (343 = 157h). The record contains a subtype identifier (02h).

A Power Unit Map contains information about the power supply sensor association and redundancy definition for the associated power unit sensor. The sensor numbers for the power unit redundancy sensor and associated power supply sensors are provided. The *Get Power Supply State* command also uses this information to determine what to return to the requestor.

Table 106. Power Unit Map SDR Record Format

	Byte	Name	Description
	0:2	OEM ID	Intel manufacturers ID – 157h, little endian
	3	Record Subtype	Value 02h, See Table 105.
Power Unit Map Record	4	Power Unit Redundancy Sensor Number	Sensor number of the power unit redundancy sensor that this record configures.
	5	Flags	Bits [7:1] Reserved Bit [0] 1 = No monitorable supplies present
	6	Power Supplies Required	The number of present and operational (not failed) power supplies required for the power unit to be considered operational (n in n+m).
	7:N	Power Supply Sensor Numbers	Array of associated power supply sensor numbers.

Table 107. Example Power Unit Map SDR Record for a 2+1 Power Subsystem

	Byte	Value	Description
	0:2	000157h	Intel manufacturers Id – 157h
	3	02h	Value 02h, See Table 105.
Power Unit Map Record	4	41h	Sensor number of the power unit redundancy sensor that this record configures.
	5	00h	Bits [7:1] Reserved Bit [0] 1 = No monitorable supplies present
	6	02h	Two supplies required for normal operation
	7	3Dh	Power supply 1 sensor number
	8	3Eh	Power supply 2 sensor number
	9	3Fh	Power supply 3 sensor number

System Information Record

This System Information OEM SDR record provides the system information including platform and chassis. This record is for upper level software information and does not affect firmware operation.

Table 108. System Information SDR Record Format

Byte		Name	Description
0:2		OEM ID	Intel manufacturers ID – 157h, little endian
3		Record Subtype	Value 06h, See Table 105.
System Information Record	4	Capabilities	Provides capabilities of the system including server board/chassis. Bits 7:1 Reserved Bit 0 0b = System has no ID LED 1b = System has ID LED

TControl Fan Speed Control Record

This record provides the following information to control the fans based on Temperature:

- Control values to be used for normal, boost and sleep states for each Fan Control Domain.
- Ramp step and Scan rate at which a new contribution is added to the fan control domain.
- Which temperature sensor and algorithm that contributes for the fan domain.
- A table that maps from raw sensor temperature readings to fan control PWM value (in %) for Stepwise Linear type Temperature Sensor.
- PWM value (in %) and scan rate for a clamp type Temperature Sensor.

Note: The Normal, Sleep, and Boost control values in the record do not configure fan speeds, but the PWM value that the BMC will provide to the hardware to control power to the fans.

Multiple instances of this SDR record can exist that refer to the same Fan Control Domain. In that case, all the values in the headers (Normal, Sleep, Boost, Ramp Step, Scan Rate, and Flags) should be the same from record to record. The BMC makes no guarantees as to which order the SDR records will be processed.

Note: The default values provided in the platform SDR files should not be modified under normal circumstances. This description is provided to allow customer to modify these values if required for a specific chassis or environment. If thermal problems occur after modification of this SDR, return to the standard values for the SDR record as provided in the platform SDR.

Table 109. Temperature Fan Speed Control SDR Record Format

Byte	Name	Description	
0:2	OEM ID	Intel manufacturers ID – 157h, little endian	
3	Record Subtype	Value 07h, See Table 105.	
Temperature Fan Control Record	4	Fan Control Domain	Fan Control Domain number defined in the platform PIA. 0 means no domain is defined.
	5	Normal control value	If No Temperature Sub-records present: This is the PWM value (% value – 0-100) to use when fans are commanded to be at nominal speed. If Temperature Sub-records present: Unused.
	6	Sleep control value	PWM value (% value – 0-100) to use when the system is in S1 sleep state and controlling fans on sleep is enabled (via <i>Set ACPI Configuration Mode</i> command).
	7	Boost control value	PWM value (% value – 0-100) to use when a fan has failed or a critical temperature has been detected.
	8	Ramp Step	Max step up/down ramp (in %) when applying contribution.
	9	Scan Rate	Number of scan cycles before applying next Ramp step.
	10	Flags	Reserved for later use. (Reserved bits are initialized to 0)
	Temperature Sensor sub-records		

Table 110. Stepwise Linear Type Temperature Sensor Sub-record

Byte	Name	Description	
Additive type Temperature Sensor Sub-record	0	Temperature Sensor Number	Sensor number of the temperature sensor associated with the fan control domain.
	1	Control Type	01 – Stepwise Linear
	2	Temperature Info 7 – Reserved 6 – Domain Maximum 5:3 – Negative Hysteresis 2:0 – Positive Hysteresis	Reserved bits are initialized to 0. If the Domain Maximum bit is set, then this sub-record provides the maximum nominal value that the Fan Control Domain will be set to. In this case, multiple control pairs may be provided and are interpreted as a table of maximum PWM values for the associated temperature. Behavior is unpredictable if multiple sub-records contributing to the same Fan Control Domain are configured as Domain Maximums. This hysteresis is used only in Temperature Fan Control module.
	3	Control pair count (N)	Count of Temperature/Control Value Pairs
	4	Temperature 1	The temperature at which a new contribution required.
	5	PWM % Value 1	PWM value (in %) to use for the Fan Control domain.
	6-M	Temp/PWM 2-N	Up to N pairs of temperature/PWM values. These entries should be in order of increasing temperature, from low to high.

Table 111. Clamp Type Temperature Sensor Sub-record

Byte	Name	Description	
Clamp type Temperature Sensor Sub-record	0	Temperature Sensor Number	Sensor number of the temperature sensor associated with the fan control domain.
	1	Control Type	02 – Clamp
	2	Ramp Contribution	Ramp contribution (in %) up/down over time to reduce temperature to under threshold.
	3	Scan Rate	Number of scan cycles when the ramp contribution is re-assessed.
	4	Temperature	The temperature at which clamp contribution required or temperature offset for TControl value.
	5	Clamp Flags [7:4] CPU number [3:1] Reserved [0] Temp Source 1 = Use TControl 0 = Use Fixed Temp	This field configures the clamp temperature source. If <i>fixed temp</i> is selected, then the above <i>Temperature</i> field is used as the clamp value. If <i>TControl</i> is selected, then the value provided via the <i>Set CPU TControl</i> command for the indicated CPU is used. The actual temperature control value is the TControl value plus the <i>Temperature</i> field. Reserved bits should be set to 0.

Fan Redundancy Map Record

A Fan Redundancy Map record contains information about the fan sensor association and redundancy definition for a fan redundancy sensor. The sensor number of the fan redundancy sensor is provided, as are all the sensor numbers for the associated fans. The record includes the number of required fans (N), the total number of fans (N+M), and the fan speed event states that indicate a fault.

Table 112. Fan Redundancy Map SDR Record Format

Byte	Name	Description	
0:2	OEM ID	Intel manufacturers Id – 157h, little endian	
3	Record Subtype	Value 05h, See Table 105.	
Fan Redundancy Map Record	4	Fan Redundancy Sensor Number	Sensor number of the fan redundancy sensor that this record configures.
	5	Flags 7:1 reserved 0 persistently stored event state	If bit 0 of the flags is set to a 1, the associated fan redundancy sensor will store its event assertion state persistently.
	6:7	Failed Event State Mask	This mask defines the Fan Speed Sensor event offsets that indicate that the associated fan has failed.
	8	Redundant Fan Count	The number of fans (M) that can fail and the fan set still provide sufficient cooling. The total number of fans in the set (N+M) is determined by the number of fan subrecords below and which of those fans are actually enabled.
	9:10	Fan Speed Sensor #	This sub record defines the sensors associated with a single fan. If the fan is not a hot swap fan, the <i>Fan Presence Sensor</i> field will be set to 0. One of these subrecords should exist for each fan in the fan redundancy set.
		Fan Presence Sensor #	
11:N	Other fan subrecords	One pair of subrecords for each fan in the set	

PC87427 SIO-3 Fan Measurement Configuration Record Format

This record controls how the BMC interprets, scales and decodes the SIO-3 fan tachometer measurements.

The return format bit determines what type of 8-bit value is returned: a scaled RPM value or a scaled period value, on a per-fan basis. The fan monitoring circuit in the SIO-3 is initialized to process fan tachometer pulses according to the pulse-per-revolution setting, also on a per-fan basis. The maximum RPM values are used to scale the return values such that an 8-bit RPM value of 0xFF equals the maximum RPM for the corresponding fan. When scaled period values are being returned rather than RPM, a value of 1 represents the shortest period (i.e. maximum RPM) while larger values represent slower fan speeds. Lack of resolution in the scaled period value prevents fan speeds slower than about 300 RPM.

Note: The default values provided in the platform SDR files should not be modified under normal circumstances. This description is provided to allow customer to modify these values if required for a specific chassis or environment. If thermal problems occur after modification of this SDR, return to the standard values for the SDR record as provided in the platform SDR.

Table 113. PC87427 SIO-3 Configuration SDR Record Format

Byte	Name	Description
0:2	OEM ID	Intel manufacturers Id – 157h, little endian
3	Record Subtype	Value 09h, See Table 105.
SIO-3 Configuration Record	4	Return format Bit 0 (LSB) = Fan 0, Bit 1 = Fan 1, etc. 0 = RPM, 1 = Period
	5	Number of tach pulses per revolution Bit 0 (LSB) = Fan 0, Bit 1 = Fan 1, etc. 0 = 2 Pulses/Rev, 1 = 1 Pulse/Rev
	6:7	Fan 0 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	8:9	Fan 1 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	10:11	Fan 2 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	12:13	Fan 3 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	14:15	Fan 4 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	16:17	Fan 5 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	18:19	Fan 6 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM
	20:21	Fan 7 Max RPM (in LSB,MSB order) 0000h = 0 RPM, FFFFh = 65,535 RPM

Web Server Map Table Size Record Format

An Intel web page includes a map of the server showing approximate physical device locations. The map is implemented as an HTML table. This record defines the height and width of the table, as well as the width of the individual cells.

Table 114. Map Table Size Record Format

Byte	Name	Description	
0:2	OEM ID	Intel manufacturers Id – 157h, little endian	
3	Record Subtype	Value 0Ah, See Table 105.	
Map Table Size Record	4	Table Rows	Number of rows in the table
	5..n	Width of column 'n'	Each byte represents the width of one table column in terms of display percentage, 1% - 100%. Columns are represented from left to right. The number of cells is inferred from the size of the record, 25 maximum.

Web Server Map Table Entry Record Format

An Intel web page includes a map of the server showing device locations. The map is implemented as an HTML table. This record defines the data required for one device in the map.

Table 115. Map Table Entry Record Format

Byte	Name	Description	
0:2	OEM ID	Intel manufacturers Id – 157h, little endian	
3	Record Subtype	Value 0Bh, See Table 105.	
Map Table Entry Record	4	Sensor Number	The sensor number associated with this map item. FFh = no sensor associated.
	5	Top Row	Row number of the top table row for this item. Rows are numbered beginning with 1.
	6	Bottom Row	Row number of the bottom table row for this item. If only 1 row is occupied by the item, this will be the same value as Top Row.
	7	Left Column	Column number of the leftmost table column for this item. Columns are numbered beginning with 1.
	8	Right Column	Column number of the rightmost table column for this item. If only one column is occupied by the item, this will be the same value as Left Column.
	9	Display Name Format	Format of the Display Name text, if included. 0 = ASCII 1 = 1 byte Unicode 2 = 2 byte Unicode 3 = 4 byte Unicode 4-FF = Reserved
	10:n	Display Name	The name to be displayed for the item. If no text is supplied, the name from the sensor's SDR record will be used. The length of the supplied string is inferred from the size of the record.

OEM SDR Tag Record Format

An OEM SDR Tag record is an IPMI OEM SDR that includes one or more SDR tag strings. The format is as follows:

Table 116. OEMSDR Tag Record Format

Byte	Name	Description	
0:2	OEM ID	Intel manufacturer Id – 157h, little endian	
3	Record Subtype	Value 54h	
SDR Tag Record	SDR Tag 0		
	4	Type/Length Byte	Encoding and length of tag string. IPMI standard format. See section 37.14 of the IPMI 1.5 specification for details.
	5:N	Tag string	Up to 30 bytes of string data. NULL termination allowed but not required.
	SDR Tags 1 - M		

Glossary

This appendix contains important terms used in the preceding chapters. For ease of use, numeric entries are listed first (e.g., “82460GX”) with alpha entries following (e.g., “AGP 4x”). Acronyms are then entered in their respective place, with non-acronyms following.

Term	Definition
ACPI	Advanced Configuration and Power Interface
BIOS	Basic input/output system
BMC	Baseboard management controller.
Bridge	Circuitry connecting one computer bus to another, allowing an agent on one to access the other.
byte	8-bit quantity.
CHAP	Challenge Handshake Authentication Protocol
CMOS	In terms of this specification, this describes the PC-AT compatible region of battery-backed 128 bytes of memory, which normally resides on the server board.
DPC	Direct Platform Control
DVI	Digital Visual Interface
EEPROM	Electrically erasable programmable read-only memory
EMP	Emergency management port.
EPS	External Product Specification
FRB	Fault resilient booting
FRU	Field replaceable unit
GB	1024 MB.
GCM	Generic communication module
GPIO	General purpose I/O
HSC	Hot-swap controller
Hz	Hertz (1 cycle/second)
I2C	Inter-integrated circuit bus
IA	Intel® architecture
IBF	Input buffer
ICH	I/O controller hub
ICMB	Intelligent Chassis Management Bus
IERR	Internal error
IFB	I/O and firmware bridge
IP	Internet Protocol
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface
IR	Infrared
ITP	In-target probe
KB	1024 bytes.
KCS	Keyboard controller style
LAN	Local area network
LCD	Liquid crystal display
LCP	Intel® Local Control Panel
LPC	Low pin count
LUN	Logical unit number

MAC	Media Access Control
MB	1024 KB
MD2	Message Digest 2 – Hashing Algorithm
MD5	Message Digest 5 – Hashing Algorithm – Higher Security
Ms	milliseconds
Mux	multiplexor
NIC	Network interface card
NMI	Nonmaskable interrupt
OBF	Output buffer
OEM	Original equipment manufacturer
Ohm	Unit of electrical resistance
PEF	Platform event filtering
PEP	Platform event paging
PLD	programmable logic device
PMI	Platform management interrupt
POST	Power-on Self Test
RAM	Random Access Memory
RISC	Reduced instruction set computing
ROM	Read Only Memory
RTC	Real-time clock. Component of ICH peripheral chip on the server board.
SDR	Sensor data record
SECC	Single edge connector cartridge
EEPROM	Serial electrically erasable programmable read-only memory
SEL	System event log
SMI	Server management interrupt. SMI is the highest priority nonmaskable interrupt.
SMM	Server management mode.
SMS	Server management software
SNMP	Simple Network Management Protocol.
UART	universal asynchronous receiver/transmitter
UART	Universal asynchronous receiver and transmitter
UDP	User Datagram Protocol.
UTC	
Word	16-bit quantity